Thesis

Towards Robust Large-scale Audio/Visual Learning

Juncheng Billy Li

CMU-LTI-23-008

(8/2/2023)

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Thanks to NCSA Delta, PSC Bridges, DARPA GAILA Project, Bosch Center for Artificial Intelligence.

Keywords: Audio Event detection; Audio Visual Event detection; Noise Robustness; Adversarial Robustness;

List of Figures

List of Tables

List of Abbreviations

AED	Audio Event Detection
AP	Average Precision
ASR	Automatic Speech Recognition
AST/ViT	Audio Spectrogram Transformer/ Vision Transformer (AST and ViT shares the same architecture)
AUC	Area Under the Curve (ROC)
BN	Batch Normalization
CNN	Convolutional Neural Network
CRNN	convolutional and recurrent neural network
CSN	Convolutional SelfAttention Network
CV	cross validation
DL	Deep Learning
DNN	Deep Neural Network
ECE	Expected Calibration Error
FC	Fully Connected Layer
KL	Kullback-Leibler (divergence)
LLM	Large Language Model
mAP	mean Average Precision
MAUC	Mean Area Under the Curve (ROC)

MFCC	Mel-frequency cepstral coefficients
MIL	multiple instance learning
NAS	Network Architecture Search
PGD	Projected Gradient Descent
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SL	Supervised Learning
SMI	standard multiple instance assumption
SoTA	state-of-the-art
SSL	Self-Supervised Learning
VTT	Video-to-text Retrieval

Abstract

Audio Visual Event Detection has benefited greatly from the advancement of deep learning in the past few years. Various model architectures have been applied to the task in multiple modalities, pushing the performance benchmark and enabling the deployment of such models in many critical tasks such as surveillance and malicious content filtering. However, the research community still lacks: 1) a **systematic understanding** of the different machine learning models' behavior given the *unique nature* of audio signals compared to the image or text counterparts. 2) The **robustness** of different models used for audio-visual learning also remains to be an under-studied area.

The first goal of this thesis is to investigate best practices for building an audio-only and audio-visual learning system that performs well. Specifically, we analyze the features, compare different architectures, and understand the difference in training techniques to provide a comprehensive and thorough understanding. Our investigation traces the evolution of models from the convolutional family to the Transformer family, and the transition in learning paradigms from supervised learning to self-supervised learning. (This part is elaborated in Chapters 2,3,4,5,6,7)

The second goal is to study the robustness of each model by gauging their behavior under noise and adversarial perturbation. We first demonstrate the existence of real-world threats caused by adversarial perturbation in both the visual and audio domains. Following this, we broaden our adversarial robustness analysis beyond the scope of unimodal audio input to include a myriad of modalities such as audio, video, imagery, and text. (This part is covered in Chapters 8,9,10, 11) Further, we extend our research purview to include a comparative study between adversarial robustness and noise robustness (Chapter 12). Aiming at fulfilling the promise of both generalization and robustness in audio-visual learning, we present our audio-journey diffusion system. We utilize the diffusion model as an effective data augmentation instrument, adding semantically diverse samples to enhance performance, demonstrating the potential for generalization. Additionally, we take advantage of the diffusion model's innate denoising capabilities, suggesting that it could readily enhance the robustness of existing audio classification systems. (Chapter 13)

Acknowledgements

In Fall 2019, I officially began my PhD-level study after having a few years of working experience at Bosch and 2 master's degree (MLT 2019, ETIM 2014). After proposing my PhD thesis in 2022, now in 2023, I am finally wrapping up my long PhD journey. To begin, I'd like to extend my deepest gratitude to all those who've lent their assistance throughout the years and to everyone I've had the privilege of working alongside. This includes my advisor, my collaborators, my family, and my cherished friends. I want to express my deepest gratitude to my life partner Ying and my parents, whose unwavering support has been invaluable.

Throughout the course of my 5.5-year journey through MLT+PhD, I've experienced a rollercoaster of highs and lows. There were momentous occasions, such as when we were honored with the best paper awards at ICMR 2018, and the best student paper award at ICASSP 2022. Contrasting these highs, my PhD journey was also marked by the global COVID pandemic, a period when it felt like "the world had collapsed." This chapter inevitably cast a shadow, making it a particularly challenging year for me.

During those difficult times, I faced significant struggles, but I was fortunate to receive support and guidance from my advisor and several others that helped me find my way back to my "mojo". I am particularly indebted to Angie Lusk, who introduced me to a fresh perspective on self-management and the art of writing.

Initially, my perception of a PhD was anchored around the idea that it was primarily concerned with the execution of outstanding research, mostly involving the exploration of the external world. Certainly, our role as PhD students at CMU LTI encapsulates conducting ground-breaking research into pertinent AI-related problems, enhancing systems, or developing novel algorithms to amplify the cognitive abilities of computers. However, what I found absent in this understanding was the recognition that a PhD journey is as much an inward journey as it is an outward exploration.

We ought to invest time in understanding ourselves on a deeper level, discovering our authentic interests, our habits, and most critically, our rhythm - understanding when to fully immerse ourselves in work with a laser-like focus, and when it's necessary to disengage and relax. While it may seem straightforward, this introspective exploration holds just as much, if not more, importance as our outward investigations.

The inward exploration doesn't only equip us to manage stress, but it also aids us in inching closer to understanding our true calling and identifying what genuinely holds significance in our lives. In my opinion, this revelation stands as the most crucial takeaway from my PhD journey.

Delving into this in more depth would necessitate an entirely separate book, so I'll reserve that discussion for my blog or a potential future book. Now, let's pivot and delve into the specifics of my technical work.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The ability of machines to interpret sound, much like humans, is increasingly prevalent across diverse settings, from smartphones and security systems to autonomous robots. This advancement in sound understanding has the potential to enable a vast array of applications, including intelligent machine state monitoring through acoustic data, enhanced visual surveillance with acoustic assistance, cataloging, and information retrieval uses such as audio archive search and audio-assisted multimedia content search. To imbue an AI with the ability to comprehend audio, the initial hurdle is addressing the Audio/Visual Event Detection (AED) task. This entails characterizing the acoustic event within an audio stream (which may be augmented by visual data) by assigning a semantic label to it. This is the primary focus of the research conducted in this thesis.

In the past decade, many neural network models have been applied to the AED task and have shown great performance. [Dai+17; WLM19; Kon+19b; For+19] showcased the effectiveness of using the convolutional family architecture, which has become the mainstream architecture for audio classification. However, a shift is being observed recently with the advent of pure attention-based neural architectures like DeiT [Tou+21] which are now topping audio tagging leaderboards [GCG21a], challenging the previously uncontested dominance of CNN family models in audio classification. Moreover, the evaluation performance of AudioSet [Gem+17b] - the largest available weakly-labeled sound event dataset - has seen significant improvement. The mean average precision (mAP) has risen from an initial 31.2 when the task was first released in 2017 to the current 47.1 mAP.

However, a detailed examination reveals significant variations in the research published in this area. For instance, some past comparisons were drawn between audio-visual systems and audio-only systems, and the performance of models initialized with pre-trained

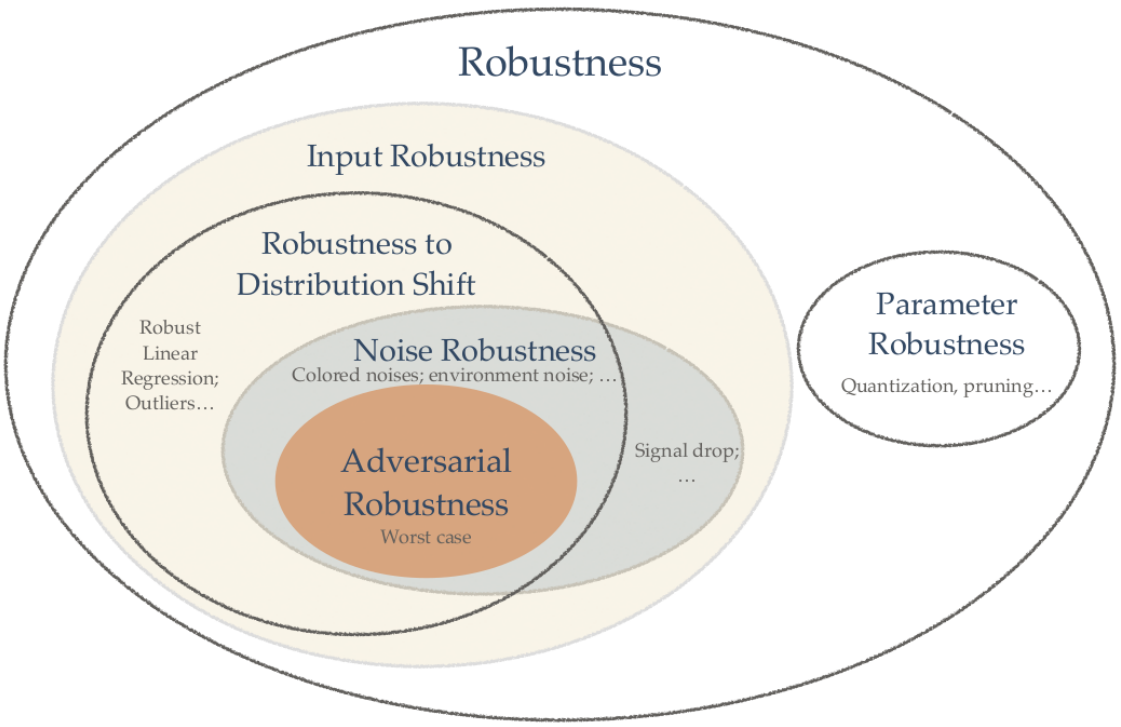


Figure 1.1: Categories of Robustness

weights greatly differs from those without pre-training [GCG21a]. Training durations for models can span from mere hours to several weeks, and ensemble systems often secure unwarranted advantages over single-checkpoint models. Frequently, the essence of these studies is concealed in seemingly minor details, leaving key information unexplained.

In this thesis, **our primary aim** is to articulate the most effective approach to constructing a large-scale audio-visual learning system. To this end, we conduct thorough experiments on AudioSet [Gem+17b]. We evaluate state-of-the-art baselines on the AED task and delve into the performance and efficiency of two principal categories of neural architectures: CNN variants and attention-based variants. We examine these models in both audio-only and audio-visual contexts, ensuring reproducible specifications. We also scrutinize their optimization procedures and carry out an analysis focused on data quality and efficiency. In addition, we also conducted a study on their efficiency-accuracy trade-off across GPU versus CPU platforms.

Despite accuracy, since audio-visual systems are widely applied in safety-critical tasks such as content filtering and surveillance, this calls for a thorough understanding of their robustness. Robustness is an overloaded term as is shown in Fig. 1.1. This definition is a summary of Robustness defined in[BBV04; Bis07; Li18]. In this thesis, we are mainly focusing on input robustness, noise robustness, and adversarial noise. [1]

The robustness study of this thesis begins by acknowledging and validating the threats, showcasing the real-world presence of unimodal adversarial perturbations against visual-only and audio-only classification systems. In order to tackle such realistic threats, in this thesis, we focus on *how audio-only and audio-visual event classification systems can be built so that they are robust to noise and adversarial attacks.* We study how to fuse audio with other modalities, and what is the best loss functions for multimodal input (e.g. audio+visual). Conversely, we utilize adversarial perturbation as a precise tool to dissect and comprehend the robustness of each element within a multimodal system, specifically examining early fusion versus late fusion and time versus frequency. We then aim to identify a practical metric that can aid in quantifying robustness on a large scale and elucidate the relationship between adversarial robustness and noise robustness. Ultimately, our research investigates methods capable of enhancing system robustness while maintaining reasonable computational costs.

1.2 Thesis Outline

A detailed outline of our contributions in this thesis is presented below.

Part I: Understanding what's the best practice of building a large-scale Audio Event Detection (AED) system, Chapter 2,3,4,6,7 are **audio-only** systems, and Chapter 5 is our first attempt to build an **audio-visual** system.

Chapter 2 and Chapter 3 list our efforts studying CNN family neural architecture on audio classification tasks on smaller datasets. In Chapter 2, we present a comparison of several Deep Learning models on the IEEE challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) 2016 challenge acoustic scene classification task and data. We perform experiments on six sets of features, including standard Mel-frequency cepstral coefficients (MFCC), Binaural MFCC, log Mel-spectrum, and two different large-scale temporal pooling features extracted using OpenSMILE. On these features, we apply five models: Gaussian Mixture Model (GMM), Deep Neural Network (DNN), Recurrent Neural Network (RNN), Convolutional Deep Neural Network (CNN), and i-vector. With large feature sets, deep neural network models outperform traditional methods and achieve the best performance among all the studied methods. The best-performing single model is the non-temporal DNN model, which we take as evidence that environmental sounds do not exhibit strong temporal dynamics.

[1]Input Robustness: the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions; Noise Robustness: a subset of input robustness, the robustness against noise in nature; Adversarial robustness: worst case scenario, model's ability to withstand adversarial perturbation without changing output.

Chapter 3 describes a detailed case study on 1D CNN with raw audio as input on the UrbanSound8k dataset. We showed that CNNs with up to 34 weight layers are efficient to optimize over very long sequences (e.g., vector of size 32000), necessary for processing acoustic waveforms. This is achieved through batch normalization, residual learning, and a careful design of receptive fields to down-sample. Our networks are fully convolutional, without the use of fully connected layers and dropout. We demonstrate the performance gains with the deeper models. Our evaluation shows that the CNN with 18 weight layers outperforms the CNN with 3 weight layers by over 15% in absolute accuracy for an environmental sound recognition task and is competitive with the performance of models using log-mel features. **The work described in the above 2 Chapters were both published at ICASSP 2017.** [Li+17; Dai+17]

Chapter 4 presents multiple instance learning (MIL) frameworks in order to tackle weakly labeled large-scale audio datasets, and presents an analysis of various pooling functions. We find the linear softmax pooling function to perform the best among the five. We build CNN+Pooling and CRNN+Pooling network architectures. It was the first system to reach state-of-the-art audio tagging performance on AudioSet [Gem+17b], while exhibiting strong localization performance on the DCASE 2017 challenge at the same time. **The initial small-scale efforts on DCASE 2017 subset were published at InterSpeech 2018 [Tse+18; WLM18] and expanded to a large-scale setting at ICASSP 2019 [WLM19].**

Chapter 5 describes an attempt to add visual cues to audio classification networks. We conducted experiments on the audio tagging task of the DCASE 2017 challenge showing that the incorporation of video information improves a strong baseline audio tagging system by 5.3% in terms of F_1 score. However, not all audio classes benefited from the fusion. **This work was completed and published at ICASSP 2018. [Li+18]**

Chapter 6 is a comprehensive study of state-of-the-art baselines on the AED task. We study the performance and efficiency of 2 major categories of neural architectures: CNN variants and attention-based variants. We perform extensive experiments on AudioSet [Gem+17b], where we closely examine their optimization procedures. We also did an analysis based on the data quality and efficiency in Section 6.4. Our open-sourced experimental results[2] provide insights into the trade-off between performance, efficiency, and optimization process, for both practitioners and researchers. **This work was completed and published at InterSpeech 2022 [LQM+22], the efficiency study is accepted to ICML 2023 ESFomo workshop [Ber+23]**

As self-supervision methods showing success in recent years, we study a simple extension of image-based Masked Autoencoders (MAE) [He+21a] in Chapter 7 to self-supervised representation learning from audio spectrograms. Following the Transformer encoder-decoder design in MAE, our Audio-MAE first encodes audio spectrogram patches

[2] https://github.com/lijuncheng16/AudioTaggingDoneRight

with a high masking ratio, feeding only the non-masked tokens through encoder layers. The decoder then re-orders and decodes the encoded context padded with mask tokens, in order to reconstruct the input spectrogram. We find it beneficial to incorporate local window attention in the decoder, as audio spectrograms are highly correlated in local time and frequency bands. Empirically, Audio-MAE sets new state-of-the-art performance on six audio and speech classification tasks, outperforming other recent models that use externally supervised pre-training. **This work was completed and presented at NeurIPS 2022 [Poy+22]**

Part II: Understanding the robustness of multi-modal systems in both unimodal and multimodal settings.

In Chapter 8, we verified that threats from adversarial attacks exist in both the vision and audio domains in the real world. In the visual domain, we demonstrated that it is possible to adversarially fool deep image classifiers in the real world, not by modifying an object of interest itself, but by modifying the camera observing the objects using a piece of sticker for instance. The optics involved with such modifications greatly limits the types of attacks that can be physically realized, but by developing a reasonable threat model and then fitting the parameters of this model to data, we can accurately capture the allowable set of perturbations. In the audio domain, we showed we could jam an emulated Alexa model with some inconspicuous background music to deactivate the VAs while our audio adversary is present. We implemented an emulated wake-word detection system of Amazon Alexa based on recent publications. Then we computed our audio adversaries with consideration of expectation over transform and we implemented our audio adversary with a differentiable synthesizer. Our experiments show that we can effectively reduce the recognition F1 score of our emulated model from 93.4% to 11.0%. Finally, we tested our audio adversary over the air, and verified it works effectively against Alexa, reducing its F1 score from 92.5% to 11.0%. **These 2 works were completed and published at ICML 2019[LSK19] and NeurIPS 2019[Li+19].**

In Chapter 9, we study how to effectively utilize available multimodal cues from videos for the cross-modal video-text retrieval task. We propose a framework that simultaneously utilizes multi-modal features (different visual characteristics, audio inputs, and text) by an ensemble approach for efficient retrieval. We conduct extensive experiments to verify that our model combination is able to boost the performance of a video-to-text retrieval task compared to the state-of-the-art. In addition, we propose a new pairwise ranking loss function in training the embedding and studying the effect of various loss functions. The results on MSVD and MSR-VTT datasets show that our method yields significant gains compared to the state-of-the-art. **This work was completed and published at ICMR 2018[Mit+18a] (Best Paper Award) and IJMIR 2019 [Mit+19] and extend to Trecvid 2022 Workshop [YLM22]**.

In Chapter 10 We aim to study several key questions related to multimodal learning through the lens of adversarial noises: 1) The trade-off between early/middle/late fusion affecting its robustness and accuracy 2) How does different frequency/time domain features contribute to the robustness? 3) How do different neural modules contribute against the adversarial noise? In our experiment, we construct adversarial examples to attack state-of-the-art neural models trained on Google AudioSet.[Gem+17b][3] We compare how much attack potency in terms of adversarial perturbation of size ϵ using different L_p norms we would need to "deactivate" the victim model. Using adversarial noise to ablate multimodal models, we are able to provide insights into what is the best potential fusion strategy to balance the model parameters/accuracy and robustness trade-off and distinguish the robust features versus the non-robust features that various neural networks model tend to learn. **This work was completed and published at ICASSP 2021[Li+21]**.

In chapter 11, we study several key questions related to multi-modal learning through the lens of robustness: 1) Are multi-modal models necessarily more robust than uni-modal models? 2) How to efficiently measure the robustness of multi-modal learning? 3) How to fuse different modalities to achieve a more robust multi-modal model? To understand the robustness of the multi-modal model in a large-scale setting, we propose a density-based metric, and a convexity metric to efficiently measure the distribution of each modality in high-dimensional latent space. Our work provides a theoretical intuition together with empirical evidence showing how multi-modal fusion affects adversarial robustness through these metrics. We further devise a mix-up strategy based on our metrics to improve the robustness of the trained model. Our experiments on AudioSet [Gem+17b] and Kinetics-Sounds [AZ17a] verify our hypothesis that multi-modal models are not necessarily more robust than their uni-modal counterparts in the face of adversarial examples. We also observe our mix-up trained method could achieve as much protection as traditional adversarial training, offering a computationally cheap alternative. [4] **This work was completed and published at ICASSP 2022[Li+22b] (Best student paper)**.

Chapter 12 is inspired by the insights derived from Chapter10 and Chapter 11, which highlight that all existing defenses come with certain compromises. Given the increasing sizes of datasets and models, standard-trained models are poised to remain prevalent in the field. Hence, gaining a deeper understanding of the robustness of various AED model architectures, without any defense mechanisms, is crucial. Equally important is the exploration of alternative metrics that could potentially signal robustness prior to empirical robustness measurement. Sharpness, providing insight into the model's loss landscape, emerges as a promising candidate in this context. We also study adversarial robustness *versus* noise robustness, where we find the two types of robustness do not necessarily translate to each other. Our findings suggest that the architectural difference contributes to their difference in robustness. Transformer architectures are more robust overall *versus* convolutional families. Sharpness-wise, we found it difficult to establish fair

[3]Audioset is the largest available weakly-labeled audio dataset.
[4]Implementation: https://github.com/lijuncheng16/AudioSetDoneRight

comparisons across different architectures following existing definitions of sharpness metrics, though we observe that robust-aware trained models display flatness. Acknowledging the aforementioned gap, as well as the limitations of classical robustness research methods such as adversarial training, certified robustness, and smoothing, we discern that the most pragmatic way forward involves capitalizing on the development of foundational models trained on large datasets. This approach can simultaneously enhance both generalization and robustness. **The preprint of this work is at [LQM22], and will be submitted to incoming conferences in 2023.**

Part III: Future pathway to generalization and robustness through foundational models. Therefore, in Chapter 13, we leverage state-of-the-art (SoTA) Large Language Models (LLMs) to augment the existing weak labels of the audio dataset to enrich captions; we adopt SoTA video-captioning model to automatically generate video caption, and we again use LLMs to merge the audio-visual captions to form a rich dataset of large-scale. Using this dataset, we train diffusion models on this audio captioning resource. In our experiment, we first verified that our Audio+Visual Caption is of high quality against baselines and ground truth (12.5% gain in semantic score against baselines). To show diffusion help with generalization, we use the trained diffusion model to generate more diverse audio data of the same format by using textual inversion [Gal+22]. We demonstrate that we could train a classifier from scratch using the diffusion-generated data, or use diffusion to enhance classification models on the AudioSet test set, working in conjunction with mixup or other augmentation methods for impressive performance gains. To show diffusion ultimately aids robustness, we use the trained diffusion model to denoise noise tested in the previous chapter, and it showed impressive robustness gains, and also came with a certifiable guarantee. Our approach exemplifies a promising method for augmenting low-resource audio datasets, making them more generalizable, and a promising approach to improve robustness.

This work is accepted at ICML2023 ESFoMo workshop[Li+23b], and is also in submission to NeurIPS 2023, and is open-sourced. [5]

[5]https://audiojourney.github.io/

Part I

Building an Audio Visual Event Recognition System that performs well

Part I Overview:

In this part, we mainly focus our discussion on what is the best practice for building a large-scale audio-visual learning system, the readers will also see the progress in this field in chronicle order towards its maturity.

We start with comparative studies on CNN families (Chapter 2, Chapter 3), where we compare and study the latent spaces and performances of CNN models VS. other common deep neural network models under a supervised learning framework using strong labels. To scale up to a larger scale, we discuss the challenges and methods of dealing with weak labels in Chapter 4. To scale out to multimodality, we analyze the effect of introducing visual cues in Chapter 5.

In reviewing the state-of-the-art in Audio Event Detection (AED), we explore Transformer family models and their training methodologies in Chapter 6, comparing them with the CNN family models discussed in the preceding chapters.

Acknowledging the constraints of supervised learning, we introduce self-supervised techniques in Chapter 7. Here, we discuss the optimal model architecture and self-supervised tasks for effective AED, and we study how SSL would enable us to learn an optimal audio representation that potentially captures the distribution of general audio data.

Chapter 2

Deep Learning Methods for Environmental Sound Detection

2.1 Introduction

Compared with speech, environmental sounds are more diverse and span a wide range of frequencies. Moreover, they are often less well-defined. Existing works for this task largely use conventional classifiers such as GMM and SVM, which do not have the feature abstraction capability found in deeper models. Furthermore, conventional models do not model temporal dynamics. For example, the winning solutions by [Rom+13][EZ+16] for DCASE challenge 2013 and 2016, extracts MFCC and i-vectors, and they both used other deeper models for temporal relation analysis. In this chapter, we focus on the task of acoustic scene identification, which aims to characterize the acoustic environment of an audio stream by selecting a semantic label for it. We apply deep learning (DL) architectures to various feature representations generated from signal processing methods. Specifically, we use the following architectures: (1) Deep Neural Network (DNN) (2) Recurrent Neural Network (RNN); (3) Convolutional Deep Neural Network (CNN). Additionally, we explore the combination of these models: (DNN, RNN, and CNN). We also compare DL models with Gaussian mixture model (GMM), and i-vectors. We also use several feature representations based on signal processing methods: Mel-frequency cepstral coefficients (MFCC), log Mel-Spectrum, spectrogram, and other conventional features such as pitch, energy, zero-crossing rate, mean-crossing rate, etc.

This chapter covers a *comprehensive study* of a diverse set of deep architectures on the acoustic scene recognition task, borrowing ideas from signal processing as well as recent advancements in automatic speech recognition. We use the dataset from the DCASE challenge. The dataset contains 15 diverse indoor and outdoor locations (classes), such as buses, cafes, cars, city center, forest paths, libraries, trains, totaling 13 hours of audio recording (see Section 3.1 for detail). In this chapter, we present a comparison of the most successful and complementary approaches to sound event detection on DCASE, which we implemented on top of our evaluation system [Dai+16] in a systematic and consistent way.

2.2 Experiments

2.2.1 Dataset

We use the dataset from the IEEE challenge on Detection and Classification of Acoustic Scenes and Events [HMV16], and we also use the evaluation setup from the contest. The training dataset contains 15 diverse indoor and outdoor locations (labels), totaling 9.75 hours of recording (1170 files) and 8.7GB in WAV format (Dual Channel, Sample Rate: 44100 Hz, Precision: 24 bit, Duration: 30 sec each). We do a 4-fold CV for model selection and parameter tuning. The evaluation dataset (390 files) contains the same classes of audio as the training set, totaling 3.25 hours of recording and 2.5GB in the same WAV format.

2.2.2 Audio Features

We create six sets of features using audio signal processing methods:

1. *Monaural and Binaural MFCC*: Same as the winning solution in the DCASE challenge 2016 [EZ+16]. We take 23 Mel-frequency (excluding the 0th) cepstral coefficients over a window length 20 ms. We augment the feature with first and second-order differences using 60 ms window, resulting in a 61-dimension vector. We also computed the MFCC on the right, left, and channel difference (BiMFCC).

2. *Smile983 & Smile6k*: We use OpenSmile [EWS10] to generate MFCC, Fourier transforms, zero crossing rate, energy, and pitch, among others. We also compute first and second-order features resulting in 6573 features. We select 983 features recommended by domain experts to create the 983-dim feature. Note that this is a much larger feature set than the MFCC features and each feature represents a longer time window of 100 ms.

3. *LogMel*: We use LibROSA [McF+15] to compute the log Mel-Spectrum, and we use the same parameters as the MFCC setup. This is the mel log powers before the discrete cosine transform step during the MFCC computation. We take 60 mel frequencies and 200 mel frequencies resulting in 60-dim and 200-dim LogMel features.

All features are standardized to have zero mean and unit variance on the training set. The same standardization is applied at the validation and test time.

2.2.3 Models and Hyperparameter Tuning

Guassian Mixture Models (GMMs)

We use the GMMs provided by the DCASE challenge committee [HMV16] as the baseline system for acoustic scene recognition. Each audio clip is represented as a bag of acoustic features extracted from audio segments, and for each class label, a GMM is trained on this bag of acoustic features using only audio clips from that class.

i-vector Pipeline

We replicate the i-vector [KBD05] pipeline from [EZ+16]. The universal background model (UBM) is a GMM with 256 components trained on the development dataset using BiMFCC feature. The mean super vector M of the GMM can be decomposed as $M = m + T{\cdot}y$, where m is an audio scene independent vector and $T{\cdot}y$ is an offset. The low-dimensional (400-dim) subspace vector y is an audio scene-dependent vector, and it is a latent variable with the normal prior. The i-vector w is a maximum a posteriori probability (MAP) estimate of y. We use the Kaldi Toolkit [Pov+11b] to compute T matrix and perform Linear Discriminant Analysis (LDA).

Deep Neural Networks (DNNs)

Multi-layer perception has recently been successfully applied to speech recognition and audio analysis and shows superior performance compared to GMMs [GMH13]. Here we tried various sets of hyperparameters including depth (2-10 layers), number of hidden units (256-1024), dropout rate (0-0.4), regularizer (L1, L2), and various optimization algorithms(stochastic gradient descent, Adam [KB14c], RMSProp [DV15], Adagrad [DHS11]), batch normalization [IS15b], etc. All the deep models we tried in the next two sections are tuned via cross-validation (CV) to achieve their best performance.

Recurrent Neural Networks (RNNs)

Bidirectional architectures generally perform better than uni-directional counterparts. We tried both LSTM [HS97] and GRU [Chu+14b] bidirectional layers. Our network only has 2 layers (one direction a layer) due to convergence time and limited improvement from deeper RNN models [IC14].

Convolutional Neural Networks (CNNs)

Lately, CNNs have been applied to speech recognition using spectrogram features [Han+14b] and achieve state-of-the-art speech recognition performance. We employ architectures similar to the VGG net [SZ14d] to keep the number of model parameters small. The input

DNN	**RNN**	**CNN**
Input depending on feature		
Dense 256	GRU 256	32×3×3-BN-ReLu
BN + Dropout0.2	GRU 256	32×3×3-BN-ReLu
Dense 256	Dropout0.4	MaxPool2×2+Dropout0.3
BN + Dropout0.2	BN	64×3×3-BN-ReLu
Dense 256		64×3×3-BN-ReLu
BN + Dropout0.2		MaxPool2×2+Dropout0.3
Dense 256		128×3×3-BN-ReLu
BN + Dropout0.2		128×3×3-BN-ReLu
		MaxPool2×2+Dropout0.3
15-way Softmax		

Table 2.1: Specifications of different deep neural network models in comparison.
BN:Batch Normalization; ReLu: Rectified Linear Activation Function

we use is the popular rectified linear units (Relu) to model non-linearity. We also found that dense layers in the bottom do not help but only slow down computation, so we do not include them in most experiments. Dropout layers significantly improve performance, which is consistent with the CNN behaviors on natural images. Table 2.1 shows an example of the architectures of all the DL models described above.

2.2.4 Pipeline & System Configuration

For each audio clip (train and test), our processing pipeline consists of the following: 1) Apply the various transforms (Section 2.2.2) to each audio clip to extract the feature representations; 2) For non-temporal models such as GMMs, we treat each feature as a training example. For temporal models such as RNNs, we consider a sequence of features as one training example; 3) At test time, we apply the same pipeline as training and break the audio clip as multiple instances, and the likelihood of a class label for a test audio clip is the sum of the predicted class likelihood for each segment. The class with the highest predicted likelihood is the predicted label for the test audio clip.
We train our deep learning models with the Keras library [Cak15] built on TensorFlow, using 4 Titan X GPUs on a 128GB memory, Intel Core i7 node.

2.2.5 Model Ensemble

As a side experiment, in pursuit of the best performance, we ensemble all the models mentioned above. In total, we have thirty models for the problem and five different

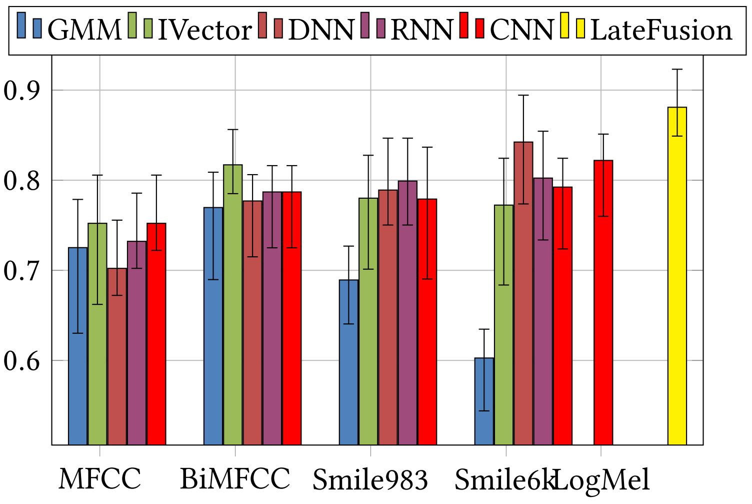


Figure 2.1: 4-fold Cross Validation avg. accuracy of 5 different models

architectures. We rank the models by performance, only best-performing models which pass a predefined accuracy threshold are included in fusion. To further stabilize the model, we construct ensembles of the ensembles. For example, the baseline GMM is excluded due to its poor performance. We test with random forest, extremely randomized trees, Adaboost, gradient tree boosting, weighted average probabilities, and other model selection methods in the late fusion [Car04]. The ML community has seen renewed interest in Model Ensemble and uncertainty quantification recently, [RT21] has surveyed the effects of using ensemble, mixup, and temperature scaling. Since ensembling is not the focus of this thesis, interested readers can refer to [Man22] for a complete list of literature on conformal predictions and ECE (Expected Calibration Error).

2.3 Results

Figure 2.1 shows the cross-validation (CV) accuracy for 5 classifiers over 6 features. 60-dim and 200-dim LogMel are listed in a single column. GMM with MFCC feature is the official baseline provided in the DCASE challenge, which achieves a mean CV accuracy of 72.5%, while our best performing model (DNN with the Smile6k features) achieves a mean CV accuracy of 84.2% and test accuracy of 84.1%. The best late fusion model has an 88.1% mean CV accuracy and 88.2% test accuracy, which is competitive with the winning solution in the DCASE 2016 challenge [EZ+16].

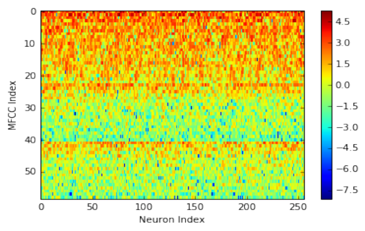
	GMM	I-Vector	DNN	RNN	CNN	Fusion
Beach	69.3	80.7	89.8	80.3	78.7	92.3
Bus	79.6	82.4	95.3	88.6	72.1	95.3
Cafe/Rest	83.2	70.0	69.9	64.7	66.4	79.9
Car	87.2	96.1	87.2	88.8	99.1	97.2
City	85.5	90.0	97.3	96.2	93.5	89.2
Forest	81.0	92.0	96.4	95.0	99.8	99.8
Grocery	65.0	93.8	79.3	75.5	85.3	96.2
Home	82.1	65.2	84.8	75.7	82.9	88.2
Library	50.4	76.1	81.2	81.6	72.7	86.2
Metro	94.7	83.5	97.3	93.7	98.7	92.3
Office	98.6	93.1	99.7	79.6	97.6	99.7
Park	13.9	78.6	49.4	45.8	45.7	71.2
Resident	77.7	66.5	76.9	68.7	81.6	77.0
Train	33.6	72.4	51.1	61.2	59.2	65.2
Tram	85.4	84.6	97.0	90.7	91.7	92.2
Average	72.5	81.7	84.2	80.2	82.2	88.1

Table 2.2: Class-wise accuracy (%) of the best CV average models.
Colored rows correspond to the most challenging classes in the confusion matrix from [Dai+16]

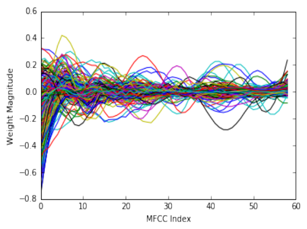
2.4 Discussion

Figure 2.1 shows that feature representation is critical for classifier performance. For neural network models (RNNs, DNNs), a larger set of features extracted from the signal processing pipeline improves performance. Among the neural network models, it is interesting to note that RNNs and CNNs outperform DNNs using MFCC, BiMFCC and Smile983 features, but DNNs outperform RNNs and CNNs on Smile6k feature. It is possible that with limited feature representation (e.g., MFCC and BiMFCC), modeling temporally adjacent pieces enhances the local feature representation and thus improves the performance of temporal models like RNNs. However, with a sufficiently expressive feature (e.g., Smile6k), temporal modeling becomes less important, and it becomes more effective to model local dynamics rather than long-range dependencies. Unlike speech, which has a long-range dependency (a sentence utterance could span 6-20 seconds), environment sounds generally lack a coherent context, as events in the environment occur more or less randomly from the listener's perspective. A human listener of environmental noise is unlikely able to predict what sound will occur next in an environment, in contrast to speech.

Table 2.2 shows that most locations are relatively easy to identify except for a few difficult classes to distinguish, such as park and residential area, or train and tram. We can also see that various models have varying performances in different classes, and



(a) Weight after FFT



(b) Weight after Smoothing

Figure 2.2: DNN's 1st layer after input

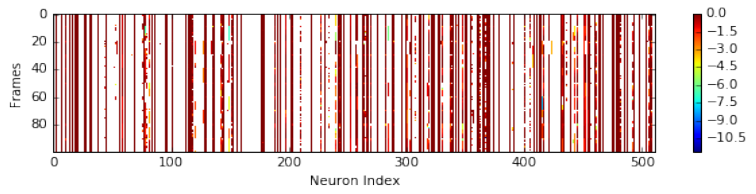
thus performing late fusion tends to compensate for individual model's error, leading to improved overall performance.

2.4.1 GMM & I-Vectors

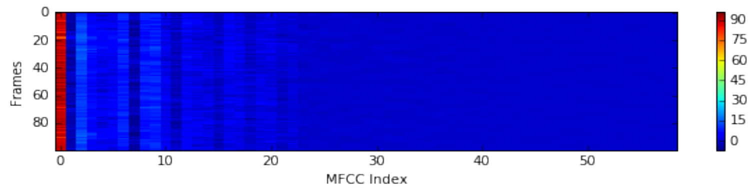
The performance of non-neural network models, particularly, the GMMs suffer from the *curse of dimensionality*. That is, in the high dimensional space, the volume grows exponentially while the number of available data stays constant, leading to a highly sparse sample. In spite of being GMM-based, this issue is less prominent for the i-vector approach since its Factorial Analysis procedure always keeps the dimension low. The performance of i-vector pipeline[KBD05] is the best among all the models using BiMFCC feature, we observe i-vector pipeline outperforms DL models with low-dimension features. We also observe i-vector pipeline tends to do better in more noisy classes such as train and tram, while suffering in relatively quiet classes such as home and residential areas.

2.4.2 DNN

Deep classifiers are able to learn more abstract representations of the feature. Figure 2.2 shows the 1st layer of a fully connected DNN model. Here, our feature is BiMFCC (61-dim). The DNN model has 5 dense layers, each with 256 hidden units. Figure 2.2(a) shows the FFT of the weight of the first layer, and indicates the responsiveness of the 256 corresponding hidden units. We note that DNN's neurons are more active in the MFCC range (0-23) and are less active in the delta of MFCC (24-41) and double delta dimension (42-61). If we apply a Savitzky-Golay smoothing function [SG64] which acts like a low-pass filter on each neuron's vector (61-dim). We get Figure 2.2(b) which is the de-noised weight of the layer (each colored line corresponds with one neuron vector), which looks like a filter bank. The chaotic responses of DNN neurons also demonstrate that DNN is *not capable of* capturing temporal information in the feature.

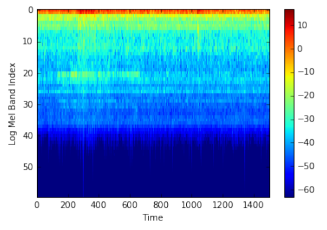


(a): RNN Neuron (512-dim) Activation

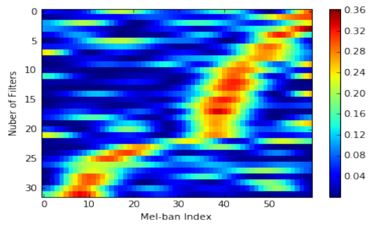


(b): BiMFCC 61 over 100 frames

Figure 2.3: RNN activation



(a)Log Mel-spectrum



(b)Weight after FFT

Figure 2.4: CNN 1st Convolutional2D layer

2.4.3 RNN

Our RNN model consists of 2 bidirectional GRU layers, and they both have 512 hidden units. Figure 2.3(a) shows the neuron activation of the forward layer of the bidirectional GRU network over 100 frames. Figure 2.3(b) shows the corresponding input feature (MFCC). With a train audio, it shows that RNN neurons are stable across the time domain as long as there is no variation of features over time. This shed light on why our RNN performs better on relatively more monotonous audio scenes such as train and tram rather than event-rich audio scenes like park and residential areas. Meanwhile, there could be a potential gain from incorporating attention-based RNN [Cho+14] here to tackle those event-rich audio scenes based on audio events.

2.4.4 2D-CNN

Figure 2.4(a) shows the input to the 2D-CNN which is a log Mel energy spectrum (60-dim). Figure 2.4(b) is the weight of the 1st convolutional layer (32 convolutional filters) after FFT. This highly resembles a filter bank of bandpass filters. We notice there is a sharp transition in filters at around the 40th Mel band. This is due to the weak energy beyond the 40th Mel band shown in Figure 2.4(a). Our finding is consistent with prior work on speech data [Gol+15a]. The filter bank we learned is relatively wider compared with that learned in speech.

2.5 Conclusion

We found that deep learning models compare favorably with traditional pipelines (GMM and i-vector). Specifically, GMMs with the MFCC feature, the baseline model provided by DCASE contest, achieves 77.2% test accuracy, while the best-performing model (hierarchical DNN with Smile6k feature) reaches 88.2% test accuracy. RNN and CNN generally have performance in the range of 73-82%. Fusing the temporal specialized models (e.g. CNNs, RNNs) with resolution specialized models (DNNs, i-vector) improves the overall performance significantly. We train the classifiers independently first to maximize model diversity and fuse these models for the best performance. We find that no single model outperforms all other models across all feature sets, showing that model performance can vary significantly with feature representation. The fact that the best-performing model is the non-temporal DNN model is evidence that *environmental (or "scene") sounds do not necessarily exhibit strong temporal dynamics.* This is consistent with our day-to-day experience that environmental sounds tend to be random and unpredictable. This chapter sets the foundation for the ensuing discussions within this thesis, offering readers a concise overview of the appearance of a typical audio classification system.

Chapter 3

Deep Dive into CNNs: 1D CNN + Raw Audio

3.1 Introduction

Acoustic modeling is traditionally divided into two parts: (1) designing a feature representation of the audio data, and (2) building a suitable predictive model based on the representation. However, it is often challenging and time-intensive to find the right representation in the so-called "feature-engineering" process, and the often heuristically designed features might not be optimal for the predictive task. Deep neural networks, which have achieved state-of-the-art performances in acoustic scene recognition as we have seen in Chapter 2, have increasingly blurred the line between representation learning and predictive modeling. Instead of using the hand-tuned Gaussian Mixture Model features and Mel-frequency cepstrum coefficients, neural network models can directly take as input features such as spectrograms [Han+14a] and even raw waveforms [HWW15]. By using simpler features, deep neural networks can be viewed as extracting feature representation *jointly* with classification, rather than separately [Sai+15]. This joint optimization is highly effective in speech recognition [Han+14a] and image classification [KSH12a], among others.

A fundamental building block of these models is the convolutional neural networks (CNNs), which can learn spatially or temporally invariant features from pixels or time-domain waveforms. CNNs have famously achieved performance competitively or even surpass human-level performance in the visual domains, such as object recognition [He+15a] and face recognition [Tai+14; SKP15a]. A common theme among these powerful CNN models is that they are usually very deep, with the number of layers ranging from tens to even over a hundred. Nonetheless, designing and training a deep network suitable for a new application domain remains challenging.

Recent works have applied CNNs to audio tasks such as environmental sound recognition and speech recognition and found that CNNs perform well with just the raw waveforms [Tüs+14; Sai+15; Gol+15b]. In one case, CNNs with time-domain waveforms can match the performance of models using conventional features like log-mel features as mentioned in Chapter 2. These works, however, have mostly considered only less deep networks, such as two convolutional layers [Sai+15; Pic15a].

In this chapter, we study very deep architectures with up to 34 weight layers, directly using time-series waveforms as the input. Our deep networks are efficient to optimize over long sequences (e.g., vector of length 32000), necessary for processing raw audio waveforms. Our architectures use a very small receptive field in the convolutional layers, but a large receptive field in the first layer chosen based on the audio sampling rate to mimic the bandpass filter. Our models are fully convolutional, without fully connected layers and dropout, in order to maximize the representation learning in the convolutional layers and can be applied to audio of varying lengths. By applying batch normalization [IS15a], residual learning [He+15a], and a careful design of down-sampling layers, we overcome the difficulties in training very deep models while keeping the computation cost low.

On an environmental sound recognition task [SJB14a], we show that deep networks improve the performance of networks with 2 convolutional layers by over 15% in absolute accuracy. We further demonstrate that the performance of deep models using just the raw signal is competitive with models using log-mel features [Pic15a].

3.2 Very Deep Convolutional Networks

Table 3.1 outlines the 5 architectures we consider. Our architectures take as input time-series waveforms, represented as a long 1D vector, instead of hand-tuned features or specially designed spectrograms. Key design elements are:

Deep architectures. To build very deep networks, we use very small receptive field 3 for all but the first 1D convolutional layers[1]. This reduces the number of parameters in each layer and controls the model sizes and computation cost as we go deeper. Furthermore, we aggressively reduce the temporal resolution in the first two layers by 16x with large convolutional and max pooling strides to limit the computation cost in the rest of the network [Sze+15]. After the first two layers, the reduction of resolution is complemented by a doubling in the number of feature maps[2]. We use rectified linear units (ReLU) for lower computation cost, following [Zei+13; SZ14b].

[1]Small receptive fields were first popularized by [SZ14b] for 2D images.

[2]In the visual domain this change in resolution and the number of features maps leads to more specialized filters at the higher layers (e.g., feature maps responding to faces) and more basic filters at the bottom (e.g., feature maps responding diagonal lines).

M3 (0.2M)	M5 (0.5M)	M11 (1.8M)	M18 (3.7M)	M34-res (4M)
Input: 32000x1 time-domain waveform				
[80/4, 256]	[80/4, 128]	[80/4, 64]	[80/4, 64]	[80/4, 48]
Maxpool: 4x1 (output: 2000 $\times$ n)				
[3, 256]	[3, 128]	[3, 64] $\times$ 2	[3, 64] $\times$ 4	$\begin{bmatrix} 3, 48 \\ 3, 48 \end{bmatrix} \times 3$
Maxpool: 4x1 (output: 500$\times$n)				
	[3, 256]	[3, 128] $\times$ 2	[3, 128] $\times$ 4	$\begin{bmatrix} 3, 96 \\ 3, 96 \end{bmatrix} \times 4$
	Maxpool: 4x1 (output: 125 $\times$ n)			
	[3, 512]	[3, 256] $\times$ 3	[3, 256] $\times$ 4	$\begin{bmatrix} 3, 192 \\ 3, 192 \end{bmatrix} \times 6$
	Maxpool: 4x1 (output: 32 $\times$ n)			
		[3, 512] $\times$ 2	[3, 512] $\times$ 4	$\begin{bmatrix} 3, 384 \\ 3, 384 \end{bmatrix} \times 3$
Global average pooling (output: 1 $\times$ n)				
Softmax				

Table 3.1: Architectures of proposed fully convolutional network for time-domain waveform inputs. M3 (0.2M) denotes 3 weight layers and 0.2M parameters. [80/4, 256] denotes a convolutional layer with receptive field 80 and 256 filters, with stride 4. Stride is omitted for stride 1 (e.g., [3, 256] has stride 1). [...] $\times k$ denotes k stacked layers. Double layers in a bracket denote a residual block and only occur in M34-res. Output size after each pooling is written as $m \times n$ where m is the size in time-domain and n is the number of feature maps and can vary across architectures. All convolutional layers are followed by batch normalization layers, which are omitted to avoid clutter. Without fully connected layers, we do not use dropout [Sri+14] in these architectures.

Fully convolutional networks. Most deep convolutional networks for classification use 2 or more fully connected (FC) layers of high dimensions (e.g., 4096 in [SZ14b; KSH12a]) for discriminative modeling, leading to a very high number of parameters. We hypothesize that most of the learning occurs in the convolutional layers, and with a sufficiently expressive representation from convolutional layers, no FC layer is necessary. We, therefore, adopt a *fully convolutional* design for our network construction [He+15a; LSD15]. Instead of FC layers, we use a single global average pooling layer which reduces each feature map into one float by averaging the activation across the temporal dimension. By removing FC layers, the network is forced to learn good representation in the convolutional layers, potentially leading to better generalization. We support this design decision in our evaluation and demonstrate that fully convolutional networks perform comparably or better compared with their counterparts endowed with FC layers.

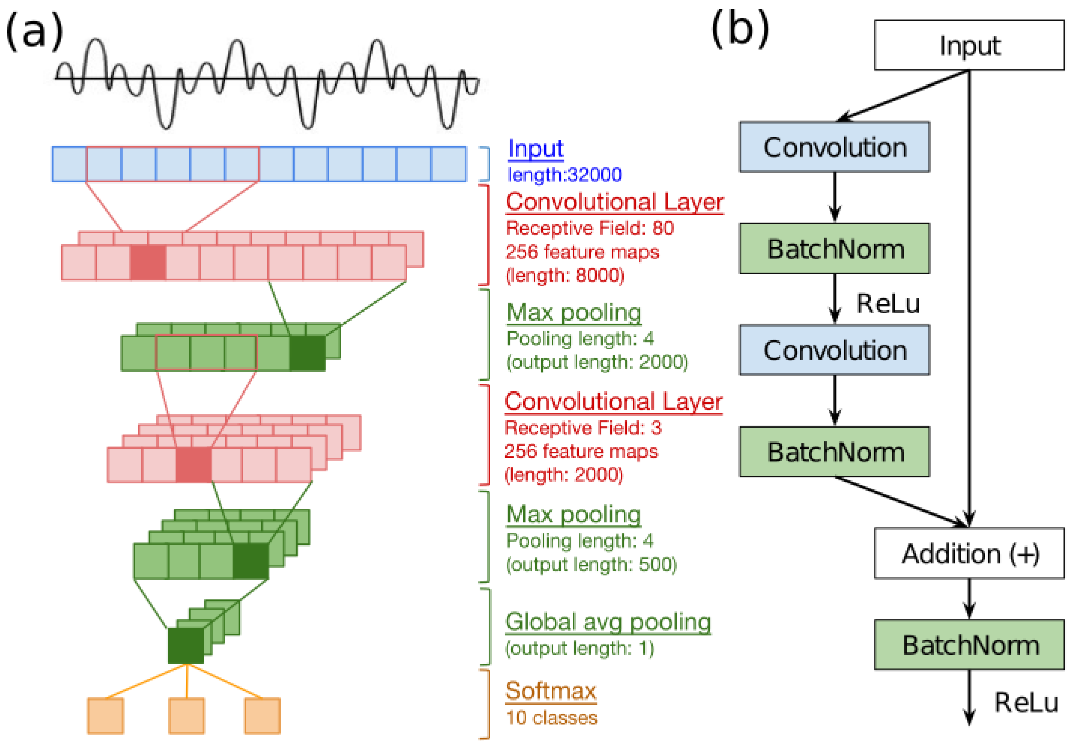


Figure 3.1: (a) The model architecture of M3 (Table 3.1).(b) Residual block (Res-Block) used in M34-res. The input audio is represented by a single feature map/channel. In each convolutional layer, a feature map encodes the activity level of the associated convolutional kernel. Note that the number of feature maps doubles as temporal resolution decreases by a factor of 4 in the max-pooling layers, capped by a global average pooling. A reduction by a factor of 4 in our max pooling layers is equal to a 2D max pooling with stride (2x2) used in many vision networks. A Res-Block consists of two convolution layers.

First layer receptive field. Time-domain waveforms at a reasonable sampling rate (e.g. 8000Hz) over a few seconds could have a very large number of samples along a single dimension. If we exclusively use the small receptive field for all convolutional layers such as in [SZ14b], which uses 3x3 in pixel for all layers, our model would need many layers in order to abstract high-level features, which could be computationally expensive. Furthermore, the audio sampling rate could affect the receptive field size in the first layer, since a field size of 80 at 8kHz sampling rate is at a different length scale than at 16kHz sampling rate. We thus choose our first layer receptive field to cover a 10-millisecond duration, which is similar to the window size for many MFCC computations. In Section 3.3 we show that a much smaller or larger receptive field gives poor performance.

Batch Normalization. We adopt auxiliary layers called batch normalization (BN) [IS15a] that alleviate the problem of exploding and vanishing gradients, a common problem in optimizing deep architectures. BN normalizes the output of the previous layer so the gradients are well-behaved. This makes possible training very deep networks (M18, M34-res) that were not studied previously [Ser+16]. Following [IS15a], we apply BN on the output of each convolutional layer before applying ReLU non-linearity.

Residual Learning. Residual learning [He+15a] is a learning framework to ease the training of very deep networks. Normally we train a block of neural network layers to fit a desired mapping $\mathcal{H}(\boldsymbol{x})$ of $\boldsymbol{x}$ ($\boldsymbol{x}$ being the input to the layers). In the residual framework, we instead let the block of layers approximate $\mathcal{F}(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$, the residual mapping. Residual learning is achieved through a skip connection in the residual block ("res-block", Figure 3.1b). We apply residual learning in M34-res (Table 3.1).

3.3 Experiment Details

We use the UrbanSound8k dataset which contains 10 environmental sounds in urban areas, such as drilling, car horn, and children playing [SJB14a]. The dataset consists of 8732 audio clips of 4 seconds or less, totaling 9.7 hours. We use the official fold 10 to be our test set, and the rest for training and validation. For computational speed, the audio waveforms are down-sampled to 8kHz and standardized to 0 mean and variance 1. We shuffle the training data but do not perform data augmentation.

We train the CNN models using Adam [KB14a], a variant of stochastic gradient descent that adaptively tunes the step size for each dimension. We run each model for 100-400 epochs (defined as a pass over the training set) until convergence. The weights in each model are initialized from scratch without any pre-trained model. We use Glorot initialization [GB10] to avoid exploding or vanishing gradients. All weight parameters are subjected to ℓ_2 regularization with a coefficient of 0.0001. Our models are implemented in Tensorflow [Aba+16] and trained on machines equipped with a Titan X GPU.[3]

Additional Models. To aid analysis, we train variants of models in Table 3.1. The "fc" models replace the global average pooling layer with 2 fully connected (FC) layers of dimension 1000 (Table 3.5), since many conventional deep convolutional networks use 2 FC layers of dimension in the thousands [KSH12a; SZ14b; Pic15a]. Following these works, we also use a dropout layer between each FC layer for regularization, with a dropout rate of 0.3. We insert a batch normalization layer after each FC layer to aid training. These models have substantially more parameters than the original models due to the FC layers (Table 3.5). Additionally, M3-big and M5-big (Table 3.4) are variants of M3 and M5,

[3]Due to copyright complications, we did not release our original implementation. We note there are very successful reproducing efforts: `https://github.com/philipperemy/very-deep-convnets-raw-waveforms.git`. Interested readers can follow their re-implementation.

respectively, with 50% and 100% more filters (e.g., 384/256 filters in the first convolutional layer in M3-big/M5-big).

3.4 Results and Analyses

Table 3.2 shows the test accuracies and training time for models in Table 3.2. We first note that M3 performs very poorly compared with the other models, indicating that 2-layered CNNs are insufficient to extract discriminative features from raw waveforms for sound recognition. This is in contrast with models using the spectrogram as input, which achieve good performance with just 2 convolutional layers [Pic15a], and shows that applying CNN directly on time-series data is challenging. M3-big, a variant of M3 with 50% more filters and 2.5x more parameters, does not significantly improve the performance (Table 3.4), showing that shallow models have limited capacity to capture time-series inputs even with a larger model.

Model	Test	Time
M3	56.12%	77s
M5	63.42%	63s
M11	69.07%	71s
M18	71.68%	98s
M34-res	63.47%	124s

Table 3.2: Test accuracies and training time per epoch (a sweep over the training set) for models in Table 3.1 on UrbanSound8k dataset using a Titan X GPU.

Deeper networks (M5, M11, M18, M34-res) substantially improve the performance. The test accuracy improves with increasing network depth for M5, M11, and M18. Our best model M18 reaches 71.68% accuracy, close to the reported test accuracy of CNNs on spectrogram input using the same dataset [Pic15a][4]. The performance increases cannot be simply attributed to the larger number of parameters in the deep models. For example, M5-big has 2.2M parameters (Table 3.4) but only achieves 63.30% accuracy, compared with the 69.07% by M11 (1.8M parameters). By using a very deep architecture, M18 outperforms M3 by as much as 15.56% in absolute accuracy, which shows that deeper architectures substantially improve acoustic modeling using waveforms. Furthermore, by using an aggressive down-sampling in the initial layers, very deep networks can be economical to train (Table 3.2 Time column). When we use stride 1 instead of 4 in the first convolutional layer for M11, we observe a 3.5x increase in training time but a lower test accuracy (67.37%) after 10 hours of training, compared with 68.42% test accuracy reached in 2 hours by M18.

[4]Figure 4 in [Pic15a] reports ~73% accuracy using CNN model on log-mel features, with probability voting. We point out that we have a different evaluation scheme: we use the 10th fold as a test set, while [Pic15a] performs a 10-fold evaluation. Also, we use sound at an 8kHz sampling rate while they use the original 44.1kHz.

Model	Test
M11-srf	64.78%
M18-srf	65.55%
M11-lrf	65.67%
M18-lrf	65.08%

Table 3.3: Test accuracies for M11 and M18 variants different receptive fields in the first convolutional layer. M11-srf and M18-srf have receptive field 8; M11-lrf and M18-lrf have 320.

Model	Test	# Parameters
M3-big	57.55%	0.5M
M5-big	63.30%	2.2M

Table 3.4: Test accuracies for M3, M5 variants with more filters in the convolutional layers. M3-big, M5-big have 50% and 100% more filters (384 and 256 filters in the first layers, respectively).

Interestingly, the performance improves with depth up to M18, at 71.68% test accuracy. M34-res only achieves 63.47% test accuracy. This is due to overfitting. We observe that with residual learning we have no problem optimizing deep networks like M34-res, and M34-res reaches an extremely high training accuracy of 99.21%, compared with 96.72% training accuracy by M18. We also observe overfitting in a residual variant of M11 network (not shown here) which reaches higher training accuracy but a lower test accuracy (by 0.17%). Overfitting caused by very deep networks is well documented [He+15a]. We believe that our dataset is too small to train M34-res without further regularization. Nonetheless, M34-res still outperforms M3 and M5.

We compare our fully convolutional network with conventional networks that use large fully connected layers (FC) for classification. Table 3.5 shows that FC layers can increase the number of parameters significantly and increase training time by 2∼95%. However, FC layers do not improve test accuracy, and in the cases of M3-fc and M11-fc the additional FC layers lead to lower test accuracy (i.e., poorer generalization). We believe that the lack of FC layers in our network design pushes learning down to convolutional layers, leading to better representation and generalization.

To understand the effect of the receptive field (RF) size in the first convolutional layer, we train M11-srf and M18-srf, variants of M11 and M18 with RF 8, and M11-lrf and M18-lrf with RF 320. Table 3.3 shows that the performance degrades significantly by 4.29% and 6.13% absolutely for M11 and M18, respectively compared to M11 and M18 with RF 80 shown in Table 3.2. Previous works have shown that the first convolutional layer, when trained on raw waveforms, mimics wavelet transforms [Tüs+14; Sai+15]. Our results suggest that a small RF popularized by vision models is insufficient to capture the necessary bandpass filter characteristics in the first convolutional layer, while a large RF smooth out local structures and cannot effectively detect local impulse patterns.

We study the effect of batch normalization (BN) in optimizing very deep networks (Table 3.6). Without BN, both M11-no-bn and M18-no-bn can be optimized to high training

Model	Test	# Parameters	Time
M3-fc	46.82%	129M	150s
M5-fc	62.76%	18M	66s
M11-fc	68.29%	1.8M	73s
M18-fc	64.93%	8.7M	100s

Table 3.5: Test accuracy for models in Table 3.1 endowed with fully connected (FC) layers. Time is training time per epoch.

Model	Train	Test
M11-no-bn	98.58%	69.38%
M18-no-bn	99.33%	62.48%
M34-no-bn	10.96%	11.45%

Table 3.6: Test accuracies for model variants without batch normalization.

accuracy. Note that M18-no-bn results in lower test accuracy, indicating that BN has a regularization effect [IS15a]. M34-no-bn could not be optimized without BN and performs close to random guess (10%) after 159 epochs of training.

Fig. 3.2 shows the learned kernels for M18 variants with different RF sizes in the first convolution layer. All of them learn a filter bank of bandpass filter. M18 (Fig. 3.2 left) has well-distributed filters. In contrast, the small RF model (Fig. 3.1 middle) has much more dispersed bands, and thus lower frequency resolution for subsequent layers. Conversely, a large RF model (Fig. 3.2 right) has fine-grained filters but does not have sufficient filters in the high-frequency range, showing that it cannot effectively respond to local high-frequency impulses.

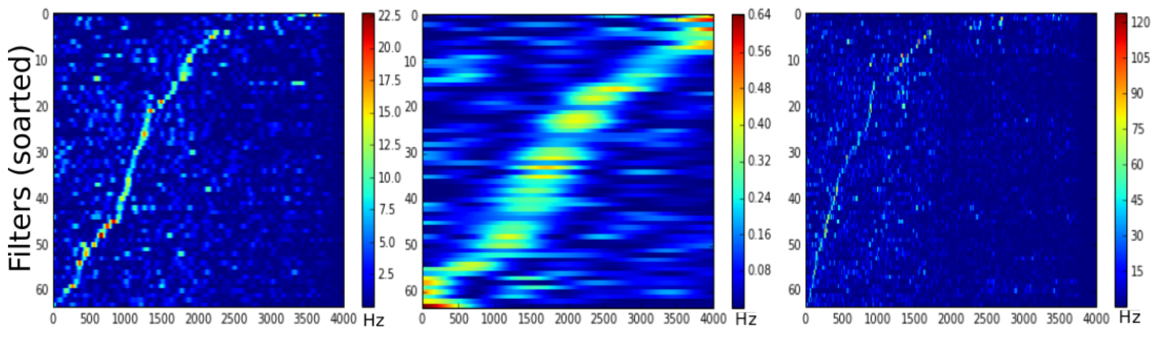


Figure 3.2: Kernels of the first convolutional layer (Conv1D) after Fourier transformation, sorted by activation frequencies.
Left: M18. Middle: M18-srf (small receptive field). Right: M18-lrf (large receptive field).

3.5 Conclusion

In this chapter, we analyzed very deep convolutional neural networks that operate directly on acoustic waveform inputs. Our networks, up to 34 weight layers, are efficient to optimize, thanks to the combination of batch normalization, residual learning, and down-sampling. We use a broad receptive field (RF) in the first convolutional layer and narrow RFs in the rest of the network. Our results show that a deep network with 18 weight layers outperforms networks with 2 convolutional layers by 15.56% accuracy absolutely and achieves 71.8% accuracy, competitive with CNNs using log-Mel spectrogram inputs [Pic15a]. Our fully convolutional networks compare favorably with those with fully connected layers. We also show that an appropriate receptive field size in the first convolutional layer based on the sampling rate of the input audio boosts performance considerably. The deep architectures we propose have the potential to enhance the performance of CNNs in audio recognition and other time-series modeling applications.

At the time of writing this thesis, this work has garnered quite some interest in the community over the years. There are 2 independent open-source effort reimplementing our system in Keras(TensorFlow)[5] and PyTorch[6]. We also observed 1D-CNN being the top choice to encode raw audio waveforms, SoTA neural Encodec system [Déf+22] achieving high-fidelity audio encoding and decoding adopted similar intuition.

[5] https://github.com/philipperemy/very-deep-convnets-raw-waveforms
[6] https://github.com/uvipen/Very-deep-cnn-pytorch

Chapter 4

Multiple Instance Learning and Pooling

4.1 Introduction

Previous chapters relied on training models using a supervised learning paradigm which requires strongly labeled data. However, given the difficulty and high resource requirement of annotating large datasets, there are only a few datasets that are publicly available and are often of limited size [SJB14b][MHV16b]. Motivated by this, many recent works have explored the use of weakly labeled data for training AED systems. One approach is to transform the audio into time-frequency representations and apply a convolutional recurrent neural network to tag or classify the entire clip [Ç+17][Cho+17]. These methods, however, involve high complexity and computation time as the recurrent and subsequent pooling layers require the full clip to be parsed before a decision can be made. Another approach for learning with weak labels is to treat segments in an audio clip as a *bag of instances* and apply multiple instance learning [Bab08]. The MIL model assumes independent labels for each instance and accounts for the uncertainty of the weak labels by assigning a positive bag label only if there is *at least one* positive instance. Evidently, this paradigm is more suitable for portable applications as the classifier can be applied to individual instances which is ideal for real-time operation.

In this chapter, we propose to enhance the framework for multi-class MIL using convolutional audio embeddings. Different from prior works, our proposed architecture addresses the issue of building low-complexity models with a small footprint for real-time applications. We propose the use of audio embeddings as input features and show that by using pre-trained embeddings the MIL model can be implemented with a simple DNN architecture. The use of audio embeddings also significantly improves AED accuracy compared to random initialization.

4.2 Multiple Instance Learning

4.2.1 MIL Framework

The task of detecting audio events using weakly labeled training data can be formulated as a multiple-instance learning problem [KR16]. In the case of binary classification, the relationship between instance labels and bag labels often obeys the standard multiple instance (SMI) assumption: the bag label is positive if and only if the bag contains at least one positive instance. In MIL, labels are assigned to *bags* of *instances* without explicitly specifying the relevance of the label to individual *instances*. All that is known is one or more *instances* within the *bag* contribute to the *bag* label. Applying this framework to our task, we view audio clip i as a *bag* of *instances* $B_i = \{x_{ij}\}$ where each *instance* x_{ij} is an audio segment j of shorter duration. We then assign all the labels of the clip to the bag so that each bag has the label $Y_i = \{y_{in}\}$ where $y_{in} = 1$ indicates the presence of audio event n. The goal of the MIL problem is then to classify labels of unseen *bags* given only the *bag* and label pairs (B_i, Y_i) as training data. In this we work we implement the MIL framework using neural networks.

4.2.2 MIL using Pooling layers

In our implementation, we generate instances by segmenting the audio clip into non-overlapping 1-second segments and taking the time-frequency representations. The segment size was chosen as a balance between the number of total instances and coverage of audio events. We use a frame size of 25ms with a 10ms shift in the short-time Fourier transform and integrate the power spectrogram into 64 Mel-spaced frequency bins. A log transform is then applied to the spectrogram. We also use the first delta as an additional input channel.

Since the spectrogram can be viewed as an image we employ convolutional layers for feature extraction. We reference CNN architectures proven to have good performance in the field of computer vision. Specifically, we use the first three conv groups from VGG-16 [SZ14a] and add two fully-connected layers of size 3072 and 1024. Batch normalization is added after each convolutional layer. The ReLU activation function is used in all layers. As our goal is a multi-label system we apply a sigmoid activation function and view the outputs as independent posterior probability estimates for each class. We use a reduced version of the full VGG model because (1) we are exploring compact models for portable applications and (2) the subset dataset does not contain enough samples to train large models without overfitting. This is where pooling functions are critical, they effectively aggregate the frame level predictions to bag level and made efficient use of the labels. After the aggregation of pooling, the multi-class MIL loss can then be defined as simply the cross entropy loss summed over all the classes, which is:

$$\mathcal{L} = -\sum_{n} \left(y_{in} \log \hat{y}_{in} + (1 - y_{in}) \log (1 - \hat{y}_{in}) \right)$$

We use the Chain rule to decompose its gradient w.r.t. the frame-level probability y_i and the frame-level weight w_i.

$$\frac{\partial \mathcal{L}}{\partial y_i} = \frac{\partial \mathcal{L}}{\partial \hat{y_{in}}} \frac{\partial \hat{y_{in}}}{\partial y_i}, \quad \frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial \hat{y_{in}}} \frac{\partial \hat{y_{in}}}{\partial w_i}$$

where the first term $\frac{\partial \mathcal{L}}{\partial \hat{y_{in}}} = -\frac{y_{in}}{\hat{y_{in}}} + \frac{1-y_{in}}{1-\hat{y_{in}}}$ does not depend on the choice of the pooling function. It is negative when the recording label is positive ($y_{in} = 1$), and positive when the recording label is negative ($y_{in} = 0$). Now, let's dive into the various pooling functions that we can use and analyze their corresponding gradients: $\frac{\partial \hat{y_{in}}}{\partial y_i}$ and $\frac{\partial \hat{y_{in}}}{\partial w_i}$

Max Pooling: To obtain a prediction for the entire bag we adopt a naïve approach and assign the label of the maximum scoring instance to the bag. The motivation behind this is in part due to the fact that since instances in a continuous audio clip are not i.i.d. many MIL algorithms are not applicable [Bab08]. However, this approach is still beneficial as it allows us to train an instance classifier that can be applied in a real-time scenario. Using this approach, the final bag label is obtained using a max pooling layer. That is

$$\widehat{Y_i} = \{\hat{y}_{in}\} = \{\max_j f_n(x_{ij}), \quad \frac{\partial \hat{y_{in}}}{\partial y_i} = \begin{cases} 1 & \text{if } y_i = \hat{y_{in}} \\ 0 & \text{otherwise} \end{cases} \}$$

where $f_n(x_{ij})$ is the predicted probability of class n on instance x_{ij}. The fact that only one frame receives a non-zero gradient may cause many frame-level false negatives. The gradient for this single frame, though, does have the correct sign: when $y_{in} = 1$, the gradient $\frac{\partial \hat{y_{in}}}{\partial y_i}$ is negative, so the frame-level probability y_i will be boosted in order to reduce the loss; when $y_{in} = 0$, the gradient is positive, so $y_i$will be suppressed.

Average Pooling: The average pooling function [Sha+18] assigns equal weight to all frames.

$$\widehat{Y_i} = \frac{1}{n} \sum_{i} y_i, \quad \frac{\partial \hat{y_{in}}}{\partial y_i} = \frac{1}{n}$$

This means the gradient is distributed evenly across all frames. For negative recordings, this will suppress the probability y_i of all frames, and this is correct behavior. For positive recordings, however, not all frames should be boosted, and the average pooling function can produce a lot of false positive frames. The equation appears to defy the SMI assumption, but it is reported to perform better than the max pooling function in [Sha+18].

Linear Softmax Pooling: Softmax pooling functions compute y as a weighted average of the y_i's, where larger y_i's receive larger weights. In this way, the recording-level probability is still mainly determined by the larger frame-level probabilities, but frames with smaller probabilities get a chance to receive an error signal. The linear softmax function assigns weights equal to the frame-level probabilities y_i themselves.

$$\widehat{Y_i} = \frac{\sum_i y_i^2}{\sum_i y_i}, \quad \frac{\partial \hat{y_{in}}}{\partial y_i} = \frac{2y_i - \hat{y_{in}}}{\sum_i y_i}$$

$\frac{\partial \hat{y_{in}}}{\partial y_i}$ is positive where $y_i > \hat{y_{in}}/2$, which gives rise to complicated and interesting behavior. For positive recordings ($y_{in} = 1$), the gradient is negative where $y_i > \hat{y_{in}}/2$, and positive where $y_i < \hat{y_{in}}/2$. As a result, larger y_i's will be boosted, while smaller y_i's will be suppressed. This is exactly the desired behavior under the SMI assumption: the frame-level probabilities are driven to the extremes 0 and 1, resulting in well-localized detection of sound events. For negative recordings ($y_{in} = 0$), the gradient is positive where $y_i > \hat{y_{in}}/2$, and negative where $y_i < \hat{y_{in}}/2$. This means all frame-level probabilities will be pushed toward $\hat{y_{in}}/2$. Considering that $\hat{y_{in}}$ is a weighted average of the y_i's, given enough iterations, all the y_i's will converge to zero as desired.

Exponential Softmax Pooling: The exponential softmax function assigns a weight of $exp(y_i)$ to the frame-level probability y_i.

$$\widehat{Y_i} = \frac{\sum_i y_i \exp(y_i)}{\sum_i \exp(y_i)}, \quad \frac{\partial \hat{y_{in}}}{\partial y_i} = (1 - \hat{y_{in}} + y_i)\frac{\exp(y_i)}{\sum_i \exp(y_i)}$$

$\frac{\partial \hat{y_{in}}}{\partial y_i}$ is always positive. As a result, the exponential pooling function also has the concern of producing too many false positive frames. Nevertheless, the problem is less serious compared to average pooling, because smaller y_i receives smaller gradients.

Attention pooling function: [Kon+18], the weights for each frame w_i are learned with a dedicated layer in the network. The recording-level probability y is then computed using the general weighted average formula. The attention pooling function appears to be most favored by researchers because of its flexibility, and variants have emerged such as the multi-level attention in [Yu+18].

$$\widehat{Y_i} = \frac{\sum_i y_i w_i}{\sum_i w_i}, \quad \frac{\partial \hat{y_{in}}}{\partial y_i} = \frac{w_i}{\sum_j w_j}, \quad \frac{\partial \hat{y_{in}}}{\partial w_i} = \frac{y_i - \hat{y_{in}}}{\sum_j w_j}$$

$\frac{\partial \hat{y_{in}}}{\partial y_i}$ is always positive. Therefore, the frame-level probabilities will be boosted or suppressed according to the recording label, with strengths proportional to the learned weights. This is correct behavior if frames with larger probabilities y_i also get larger weights w_i. However, because the weights w_i are also learned, we should also consider $\frac{\partial \hat{y_{in}}}{\partial w_i}$, the gradient of the loss function w.r.t. the weights: this term is positive where $y_i > \hat{y_{in}}$. When the recording is positive, this will cause the weight w_i to rise where the frame-level probability y_i is large and to shrink where the y_i is small, agreeing with the motivation that frames with larger probabilities y_i should get larger weights w_i. When the recording is negative, however, the opposite phenomenon will happen: larger weights will concentrate upon frames with smaller probabilities. This has serious consequences: while the recording-level probability $\hat{y_{in}}$ is indeed small, there are frames with large probabilities y_i and small weights w_i, which means the recording-level prediction and frame-level predictions will be inconsistent with the SMI assumption, and the frames with large probabilities will end up being false positives for localization.

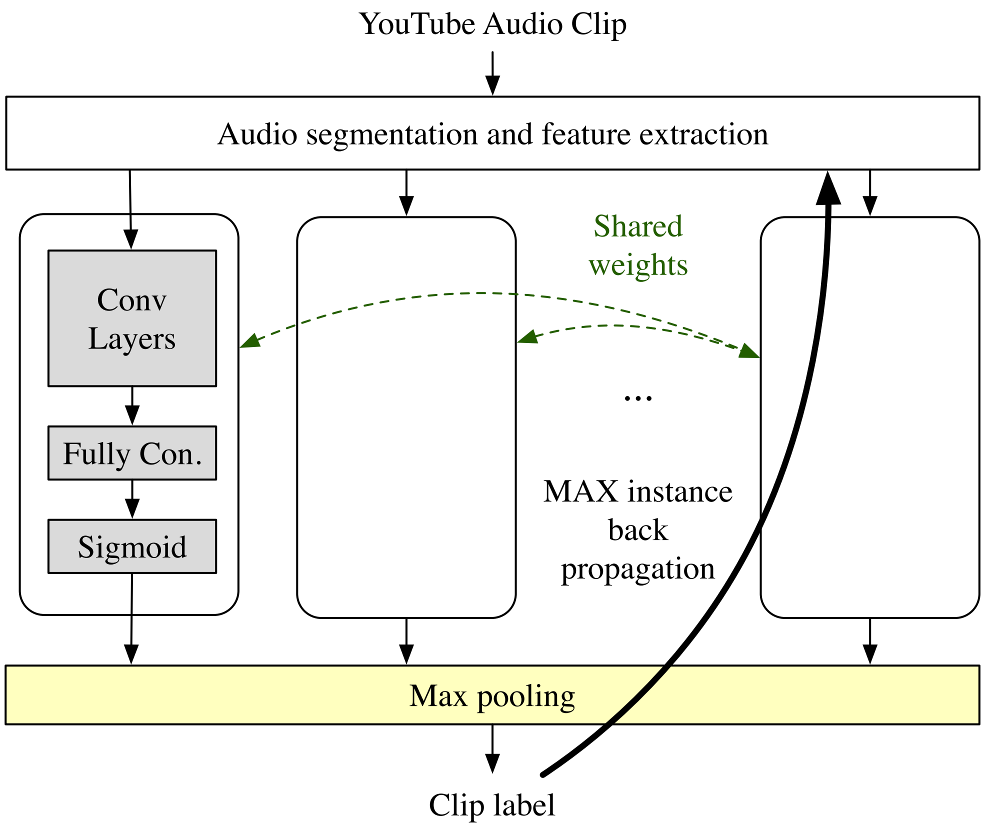


Figure 4.1: The architecture of MIL using CNN. Backpropagation is performed along the MAX instance for each class in the max pooling case. Other pooling functions will go through all blocks.

4.2.3 MIL using Audio Embeddings

Our model infers that for a certain class, the highest-scoring instances are most important and contribute directly to the corresponding bag label. The training of the neural network to identify these important instances is similar to an expectation maximization (EM) approach. However, there are two possible issues that may result from this model. The first is that as with most EM methods, system performance highly depends on the initialization point. With a bad initialization point, the model chooses the wrong instance as being indicative of the class label and optimizes on irrelevant input. These types of errors would be hard to recover from if there is a high variation for each individual audio event. A second issue is that by using a max pooling layer over all instances back-propagation will only propagate through the maximum scoring instance. This may result in some instances being ignored for most of the training. While this focus on relevant instances only is the central idea of MIL, it greatly reduces robustness to noise that occurs intermittently in the audio. We propose that the use of pre-trained audio embeddings can alleviate the above issues. By using audio embeddings as features we postulate that audio events as well as noise conditions can be better represented which can improve the performance of the MIL framework.

Similar to [Her+17a] we generate audio embeddings by training a CNN to give frame-wise predictions of the clip label. The input features are 128-bin log-mel spectrograms computed over 1-second segments of audio by short-time Fourier transform. We use the clip label as the target for all 1-second segments in the audio clip. The outputs from the penultimate layer of the CNN are then extracted and used as input to the MIL framework. We use the same CNN structure described in the previous section but add an additional fully-connected layer of size 512 to generate the final audio embedding. Since frame-wise training of the instances results in badly labeled data, the final model selection of the embedding CNN is crucial in generating meaningful embeddings. We use the maximum of frame-wise predictions as the predicted clip label and select the CNN model with the best performance at the clip-level using held-out validation data.

The MIL-DNN system is similar in architecture to the MIL-CNN but uses audio embeddings as features for each instance. The convolutional layers are replaced with fully-connected layers as we no longer deal with images. The best-performing system has four hidden layers using a ReLU activation function with layer sizes of 512, 512, 256, and 128. The architecture of the MIL-DNN framework is shown in Figure 4.2.

One step further, we added more pooling layers in between the CNN layers and added RNN blocks in the end to capture the temporal correlation. This architecture was named TALNet [WLM19], and could achieve competitive performance till 2022. The architecture of TALNet is shown in Chapter 6 Figure 6.1 (B).

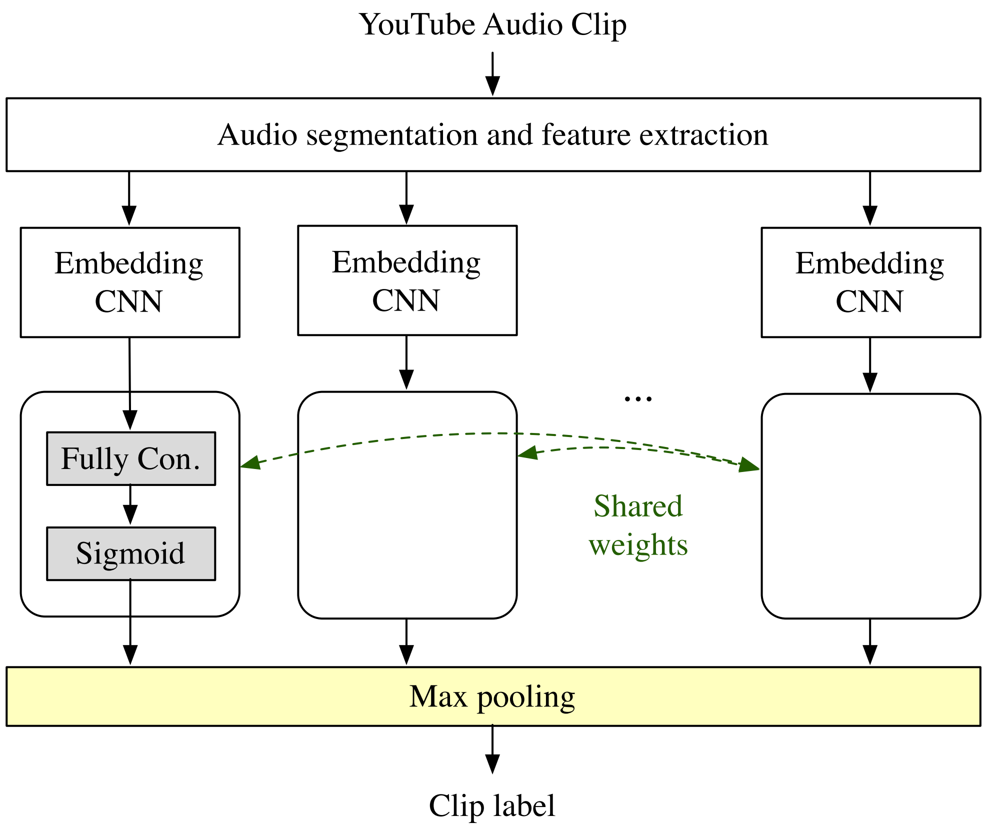


Figure 4.2: Architecture of MIL using audio embeddings.

4.3 Dataset & Challenges

4.3.1 Dataset

We evaluated our models on Google's AudioSet [Gem+17a]. AudioSet is an extensive collection of 10-second YouTube clips annotated over a large number of audio events. This dataset contains 632 audio event classes and over 2 million sound clips, however as a proof of concept we refer to a subset released by the DCASE 2017 challenge [BSE17]. The challenge subset contains 17 audio event classes divided into two categories : *Warning* and *Vehicle* sounds. These audio events are highly focused on transportation scenarios and are primed towards evaluating AED systems for self-driving cars, smart cities, and related areas. The subset contains 51,172 samples which are around 142 hours of audio. The class names and number of samples per class are shown in Table 4.1.

Class Name	**Samp #**	**Class Name**	**Samp #**
Warning Sounds		*Vehicle Sounds*	
Car alarm	273	Skateboard	1,617
Reversing beeps	337	Bicycle	2,020
Air/Truck horn	407	Train	2,301
Train horn	441	Motorcycle	3,291
Ambulance siren	624	Car passing by	3,724
Screaming	744	Bus	3,745
Civil defense siren	1,506	Truck	7,090
Police siren	2,399	Car	25,744
Fire engine siren	2,399		

Table 4.1: Class labels and number of samples per class.

4.3.2 Challenges of the Dataset

The main challenge of the dataset is the noisiness of YouTube data. As clips are user-submitted and mostly recorded using consumer devices in real-life environments, audio events are often far-field and corrupted with a variety of noise, including human speech, music, wind noise, etc. Another challenge is the variability of audio events. Even within the class, the characteristic of an audio event can vary drastically. An example of this is the use of different types of sirens by different regions which would make it hard to differentiate between *ambulance* and *fire truck sirens*. In short, it is possible that each label type encompasses all possible global variations of that category.

Finally, the number of samples per class is also highly imbalanced in the subset dataset. The imbalance ratio of the least occurring to most occurring class is 1:94. While this issue can be alleviated through machine learning techniques, the inherent shortage of information in minority classes may result in bad generalization of those classes.

These issues are further discussed in Chapter 6 Section 6.4. In Chapter 6, we also developed data-balancing techniques: a weight to to the loss proportional to the inverse frequency of each class to combat the class imbalance issue mentioned above.

4.4 Experimental Setup and Results

In all experiments, we used cross entropy as the loss function and the Adam optimizer [KB14b] to perform weight updates. To handle class imbalance the loss function was weighted inversely proportional to the number of samples for each class. For model selection of the embedding CNN, we adopted a clip-level validation scheme. The posterior class probabilities were averaged over all instances in a clip and the model with the best clip tagging accuracy was selected to generate audio embeddings.

We compared our MIL framework to an MLP baseline from the DCASE challenge [BSE17]. The best F1-score achieved by our MIL system using a CNN architecture on a two-fold cross-validation setup was 22.4%. Using audio embeddings as features and only a DNN as classifier the performance improved to 31.4% which is 20.5% absolute improvement from the DCASE baseline. We compared it to a MIL framework where the DNN classifier is replaced with a 3-layer Bi-LSTM RNN and found that results were comparable to DNNs. We also applied late-fusion to models with different hyper-parameters using a weighted majority voting scheme which improved the F1-score further to 35.3%. The weights of the voting scheme were based on model validation accuracy. Finally, we show that the performance of our MIL framework improves to 46.5% using embeddings from AudioSet. These embeddings are part of AudioSet and trained with a CNN architecture from [Her+17a] using the YouTube-8M dataset [Abu+16]. Table 4.2 shows the performance and parameter number of the different models trained on the DCASE 2017 subset and the full AudioSet [Gem+17b] respectively. Table 4.2 and Table 4.3 also empirically verified our theoretical analysis in Section 4.2.2. Evidently, all the other four pooling functions outperform max pooling in terms of F1 for both audio tagging and localization. Although the linear softmax system is not the best in terms of all the evaluation metrics, it is the only system that achieves a low error rate and a high F1 for localization. The max pooling system falls behind on the F1, while the other three systems exhibit excessively high error rates. The confusion matrix for the proposed MIL system is shown in Figure 4.3. Although there is high confusability in the *Car* class, which may be due to the imbalance of labels, the system is still able to distinguish between classes with relative accuracy.

4.5 Discussion

Under similar performance conditions, the MIL system using DNN reduces the number of parameters by a factor of almost 10 compared to a 3-layer Bi-LSTM RNN. In terms of evaluation runtime, the DNN model is also up to 5 times faster than RNNs. The DNN model is able to handle 2,500 samples per second compared to 500 samples with RNN using an NVIDIA GTX-1080 GPU.

In addition, by using independent instance classifiers our system is able to run in real-time and give running predictions of audio events. This property is crucial when applying AED in smart cars as events such as sirens and horns have to be detected as soon as they occur.

Finally, as shown by the gain in performance through the use of AudioSet embeddings, the MIL system can easily be improved through transfer learning of other sound events. An interesting observation from our experiments is that joint optimization of the pre-trained embedding CNN with the MIL loss did not improve performance much above random initialization. This shows that audio embeddings already contain rich acoustic information

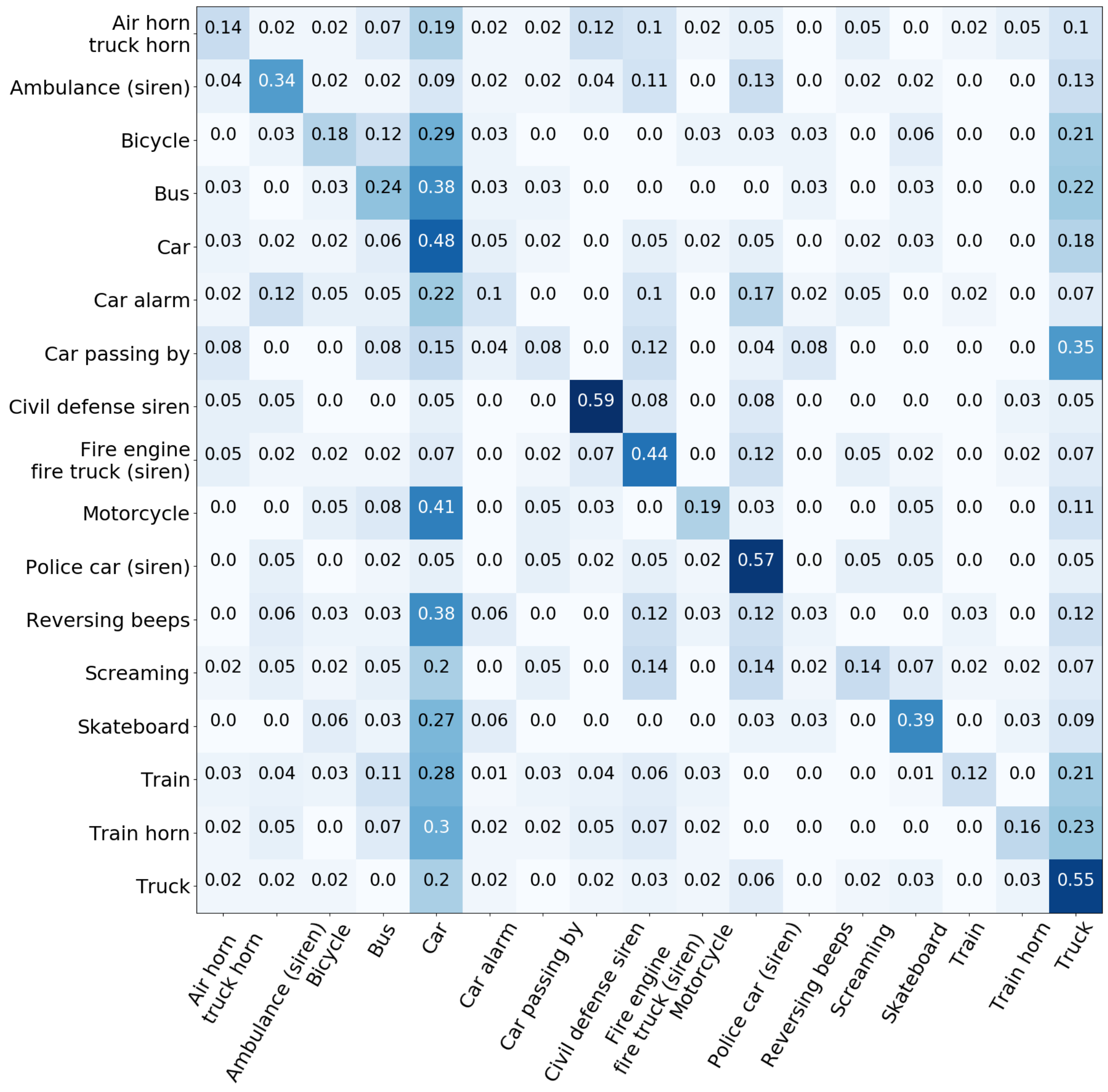
Model	**Prec.**	**Rec.**	**F1**	**Err Rate**	**Param #**
Trained and tested on DCASE 2017 subset 50k samples					
Baseline [BSE17]	7.9	17.6	10.9	-	13K
MIL-CNN	19.6	26.1	22.4	-	29M
MIL-RNN-Embed	23.7	38.1	29.2	-	6.5M
MIL-DNN-Embed	25.4	41.3	31.4	-	**700K**
Ensemble	28.6	46.0	35.3	-	-
MIL-DNN-AudioSet	41.9	52.2	46.5	-	**700K**
Trained on full Audioset 2M samples tested on DCASE 2017 subset 50k samples					
TALNet-Max Pooling	49.0	32.2	42.2	81.5	8M
TALNet-Average Pooling	42.2	49.6	**46.8**	101.8	8M
TALNet-Linear Softmax	46.2	48.5	45.4	**78.9**	8M
TALNet-Exp. Softmax	42.3	48.6	46.2	89.2	8M
TALNet-Attention	39.6	48.6	45.5	92.0	8M

Table 4.2: Comparisons of precision, recall, F1-score (%), and number of parameters for the various models.

Pooling	**mAP**	**AUC**	**d-prime**	**Train Set#**
[Her+17a]	0.314	0.959	2.45	1M
[Kon+18]	0.327	0.965	2.558	2M
[Yu+18]	0.360	0.97	2.66	2M
Max Pooling	0.351	0.961	2.497	2M
Average Pooling	0.361	**0.966**	2.574	2M
Linear Softmax	0.359	**0.966**	**2.575**	2M
Exp. Softmax	**0.362**	0.965	2.554	2M
Attention	0.354	0.963	2.531	2M

Table 4.3: Comparisons of mAP, AUC, d-prime, and number of training data for the various models on AudioSet Eval Set.

and can be trained in a task-independent manner. Such phenomenon became more evident when we trained the model on the entire AudioSet and test on the DCASE 2017 subset, the model trained on the entire AudioSet worked “out of the box", it achieved very competitive performance both on AudioSet tagging task as shown in Table 4.3, and could achieve competitive performance on DCASE 2017 subset’s localization task without fine-tuning.



	Air horn truck horn	Ambulance (siren)	Bicycle	Bus	Car	Car alarm	Car passing by	Civil defense siren	Fire engine fire truck (siren)	Motorcycle	Police car (siren)	Reversing beeps	Screaming	Skateboard	Train	Train horn	Truck
Air horn truck horn	0.14	0.02	0.02	0.07	0.19	0.02	0.02	0.12	0.1	0.02	0.05	0.0	0.05	0.0	0.02	0.05	0.1
Ambulance (siren)	0.04	0.34	0.02	0.02	0.09	0.02	0.02	0.04	0.11	0.0	0.13	0.0	0.02	0.02	0.0	0.0	0.13
Bicycle	0.0	0.03	0.18	0.12	0.29	0.03	0.0	0.0	0.0	0.03	0.03	0.03	0.0	0.06	0.0	0.0	0.21
Bus	0.03	0.0	0.03	0.24	0.38	0.03	0.03	0.0	0.0	0.0	0.0	0.03	0.0	0.03	0.0	0.0	0.22
Car	0.03	0.02	0.02	0.06	0.48	0.05	0.02	0.0	0.05	0.02	0.05	0.0	0.02	0.03	0.0	0.0	0.18
Car alarm	0.02	0.12	0.05	0.05	0.22	0.1	0.0	0.0	0.1	0.0	0.17	0.02	0.05	0.0	0.02	0.0	0.07
Car passing by	0.08	0.0	0.0	0.08	0.15	0.04	0.08	0.0	0.12	0.0	0.04	0.08	0.0	0.0	0.0	0.0	0.35
Civil defense siren	0.05	0.05	0.0	0.0	0.05	0.0	0.0	0.59	0.08	0.0	0.08	0.0	0.0	0.0	0.0	0.03	0.05
Fire engine fire truck (siren)	0.05	0.02	0.02	0.02	0.07	0.0	0.02	0.07	0.44	0.0	0.12	0.0	0.05	0.02	0.0	0.02	0.07
Motorcycle	0.0	0.0	0.05	0.08	0.41	0.0	0.05	0.03	0.0	0.19	0.03	0.0	0.0	0.05	0.0	0.0	0.11
Police car (siren)	0.0	0.05	0.0	0.02	0.05	0.0	0.05	0.02	0.05	0.02	0.57	0.0	0.05	0.05	0.0	0.0	0.05
Reversing beeps	0.0	0.06	0.03	0.03	0.38	0.06	0.0	0.0	0.12	0.03	0.12	0.03	0.0	0.0	0.03	0.0	0.12
Screaming	0.02	0.05	0.02	0.05	0.2	0.0	0.05	0.0	0.14	0.0	0.14	0.02	0.14	0.07	0.02	0.02	0.07
Skateboard	0.0	0.0	0.06	0.03	0.27	0.06	0.0	0.0	0.0	0.0	0.03	0.03	0.0	0.39	0.0	0.03	0.09
Train	0.03	0.04	0.03	0.11	0.28	0.01	0.03	0.04	0.06	0.03	0.0	0.0	0.0	0.01	0.12	0.0	0.21
Train horn	0.02	0.05	0.0	0.07	0.3	0.02	0.02	0.05	0.07	0.02	0.0	0.0	0.0	0.0	0.0	0.16	0.23
Truck	0.02	0.02	0.02	0.0	0.2	0.02	0.0	0.02	0.03	0.02	0.06	0.0	0.02	0.03	0.0	0.03	0.55

Figure 4.3: Confusion matrix for the proposed MIL system.

4.6 Conclusions

In this chapter, we proposed a multiple-instance learning framework using deep neural networks for audio event detection which can be trained using large-scale weakly-supervised data. We showed that by using pre-trained audio embeddings we can achieve good performance with a simple DNN model in an MIL framework. Audio embeddings were extracted from a CNN trained to give frame-wise predictions for the weakly labeled data. Further improvements were achieved by using embeddings from AudioSet which were trained with more data and additional labels. We postulate that audio embeddings map data into an acoustically meaningful high-dimensional space which is more indicative of audio events. Using these embeddings we can achieve a good trade-off between model size and performance. The insights gained in this chapter align with our subsequent studies regarding the utilization of embeddings learned from larger datasets.

Chapter 5

Adding Visual Cue to Audio Classification

5.1 Introduction

Strongly labeled datasets are scarce and usually domain-specific, which not only limits the potential for supervised systems trained on these datasets to be scaled up, and also severely impacts their generalizability to other domains. To tackle this problem, the audio research community has turned to a large-scale dataset with weak labels as described in Chapter 4. Compared with strongly labeled datasets, weakly labeled datasets are much less expensive to collect at scale and can cover a wider range of sound event types. In the weakly labeled datasets, the time spans of the events are not explicitly given, and only the presence or absence of events is known. For example, Google Audio Set [Gem+17b] encompasses a variety of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. Systems developed on these datasets can be suitable for a larger number of domains, such as speech sentimental analysis in paralinguistics. We discuss the MIL framework extensively in Chapter 4, which is used to tackle weak labels.

In the meantime, the vision research community is trying to explore the correlation between audio and video. The field of computer vision has been revolutionized by the emergence of massive labeled datasets [Rus+15] and learning deep representations [KSH12b; SZ14c]. Recent progress in this field has enabled machines to recognize scenes and objects in images and videos accurately, and a number of pre-trained models with very good accuracy [SZ14c; Sze+17] have been made available to the public. The success in visual recognition was well-translated into the recognition of audio: [AVT16] achieved a significant performance improvement from the state-of-the-art of audio-tagging models by transferring discriminative visual knowledge from well-established visual recognition models into the sound modality using unlabeled video as a bridge; [AZ17b] trained visual and audio networks from matching audio and video pairs without any supervision and the networks exhibited performance even superior to supervised models on audio tagging tasks; [Owe+16] showed that it was possible to synthesize audio tracks for silent videos of objects interacting with materials that could deceive the human ear. However, all of these methods require long hours and significant computation resources to train.

We consider this question: *What is the best of both worlds? Can we leverage knowledge from both the audio and vision communities to help us with audio tagging?* In this chapter, we propose a novel multimodal framework that incorporates information from both the audio and video modalities without incurring excessive computational costs. On one hand, we train a CRNN network using the audio data to produce clip-level predictions based on the multiple instance learning (MIL) paradigms. On the other hand, we extract key frames from the video data to summarize video clips, apply the pre-trained Inception V4 model [Sze+17] to recognize the visual objects in the keyframes, and then use pre-trained GloVe vectors [PSM14] to map the video labels to sound events. The final predictions are made by fusing the knowledge in the two branches. We evaluate our multimodal system on the audio tagging task of the DCASE 2017 challenge [BSE17], and show that our proposed framework improves over a baseline audio-only system by 5.3% in terms of F_1 score. In addition, our framework is lightweight and can be trained within 6 hours on a single GPU.

5.2 Methodology

5.2.1 Learning Sounds from Visual Information

As for the audio branch, we adopt the MIL framework described in Chapter 4, the structure of the MIL system [Amo13] is shown in Fig. 5.1 Because the audio and video tracks of video clips are often correlated and synchronized, we can also extract information from the video track that is indicative of the sounds in the audio track. We try to predict sound events from the video track using the following steps, which are shown in the right branch of Fig. 5.2.

Key Frame Extraction

The video track of each clip contains many video frames, some of which are nearly identical while others are distinct. Selecting a subset of frames at equal intervals can reduce the burden of subsequent object recognition, but the frames selected this way are not guaranteed to well represent the entire clip. In [Mit+17], it has been found important to select a representative set of frames (called *key frames*) to preserve the information in the video track. This is a clustering problem and could be solved using conventional clustering algorithms such as k-Means; we used the following sparse coding algorithm [ESS16] to select a representative set because it is more robust to outliers compared with K-Means.

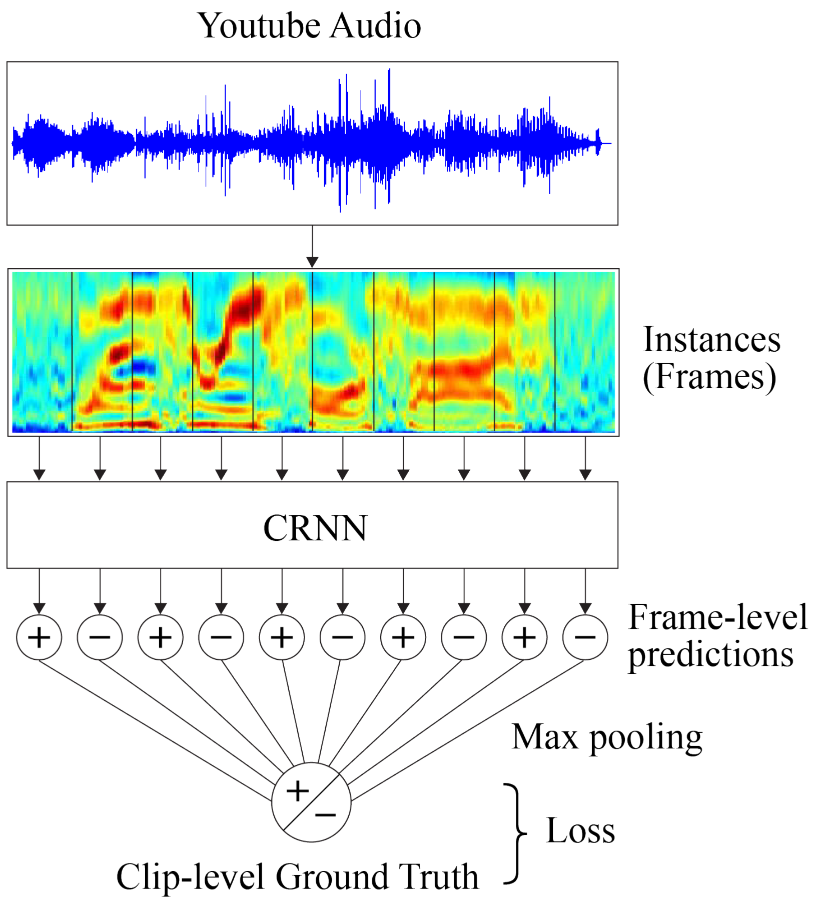


Figure 5.1: Note here, each frame only has a positive and negative prediction.

Given a set of objects $X = \{x_1, \ldots, x_n\}$, the sparse coding algorithm operates on a dissimilarity matrix $D \in \mathbb{R}^{n \times n}$, where the entry d_{ij} measures the dissimilarity between the objects x_i and x_j. In order to express the relationship of "representing" between the objects, a binary membership matrix $Z \in \mathbb{R}^{n \times n}$ is defined, in which $z_{ij} = 1$ means that x_i is a representative for x_j. The algorithm aims to minimize the total dissimilarity between each object and its representative, *i.e.* $\sum_{i,j} z_{ij} d_{ij}$. To ensure that each object is represented by one and only one object, the membership matrix Z must have each column normalized, *i.e.* $\sum_i z_{ij} = 1, \forall j$; to restrict the number of representative objects, the number of non-zero rows of Z must be as few as possible. The sparse coding algorithm, therefore, tries to solve the following optimization problem:

$$\min_{\{z_{ij}\}} \lambda \sum_{i=1}^{n} I(z_i) + \sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij} d_{ij}$$

$$\text{s.t.} \sum_{i=1}^{n} z_{ij} = 1, \forall j; z_{ij} \in \{0,1\}, \forall i, j$$

where $I(z_i) = 0$ if the i-th row of Z is all zeros and 1 otherwise, and λ is a regularization parameter that controls the size of the representative set. Because the optimization is NP-hard when the z_{ij} must be binary, the algorithm actually solves a relaxed version of the problem in which z_{ij} can be any real number in $[0, 1]$. The representatives can be selected by summing up each row of the matrix Z, and picking the top few rows with the largest sums.

In order to select key frames quickly, it is necessary to find an efficient way of encoding the frames into feature vectors. We use the light-weight, convolutional AlexNet [KSH12b] for this purpose: we feed each video frame into the network and extract features from its "conv5" layer. The dissimilarity matrix is made up of the Euclidean distances between the feature vectors. We select 4 keyframes for each clip.

Object Recognition and Knowledge Mapping

Once we get the keyframes, we can pass them through the pre-trained InceptionNet V4 model [Sze+17] to recognize the objects in them. For each keyframe, this produces a distribution of over 1,000 classes. To reduce the noise in the probabilities of unlikely classes, we "rectify" these distributions by retaining only the top 10 classes and renormalizing their probability mass among themselves. To obtain a clip-level representation of the video information, we sum up the rectified distributions of all the keyframes and rectify the result again. This yields 10 object classes and their probabilities.

Now we have a belief of what objects are present in the video track of a clip, we want to map it to the confidence of the sound events we are interested in. We conduct this knowledge mapping using pre-trained GloVe vectors (glove.840B.300d) [PSM14]. Since pre-trained GloVe vectors already contain a vector for each vocabulary from its knowledge source, we adopt the GloVe vectors from the descriptions of all the visual objects and sound events and compute the cosine similarity of each (object, sound event) pair. For each of the top 10 object classes, we assign its probability to the sound event that is closest to it in terms of cosine similarity; some sound events may receive probabilities from more than one object class. This results in an estimation of the probability of each sound event in the clip as indicated by the video track.

5.2.2 Fusion of Audio and Visual Knowledge

Both the audio track and the video track predict a probability for each sound event. The final fusion step tries to produce a refined prediction by combining the two knowledge sources. This combination is implemented with a weighted average of the two probabilities, where the weights are tunable for each sound event. The better-performing model on each individual class would have a higher weight on that class, and the weight combination is tuned toward achieving the best overall F_1 score.

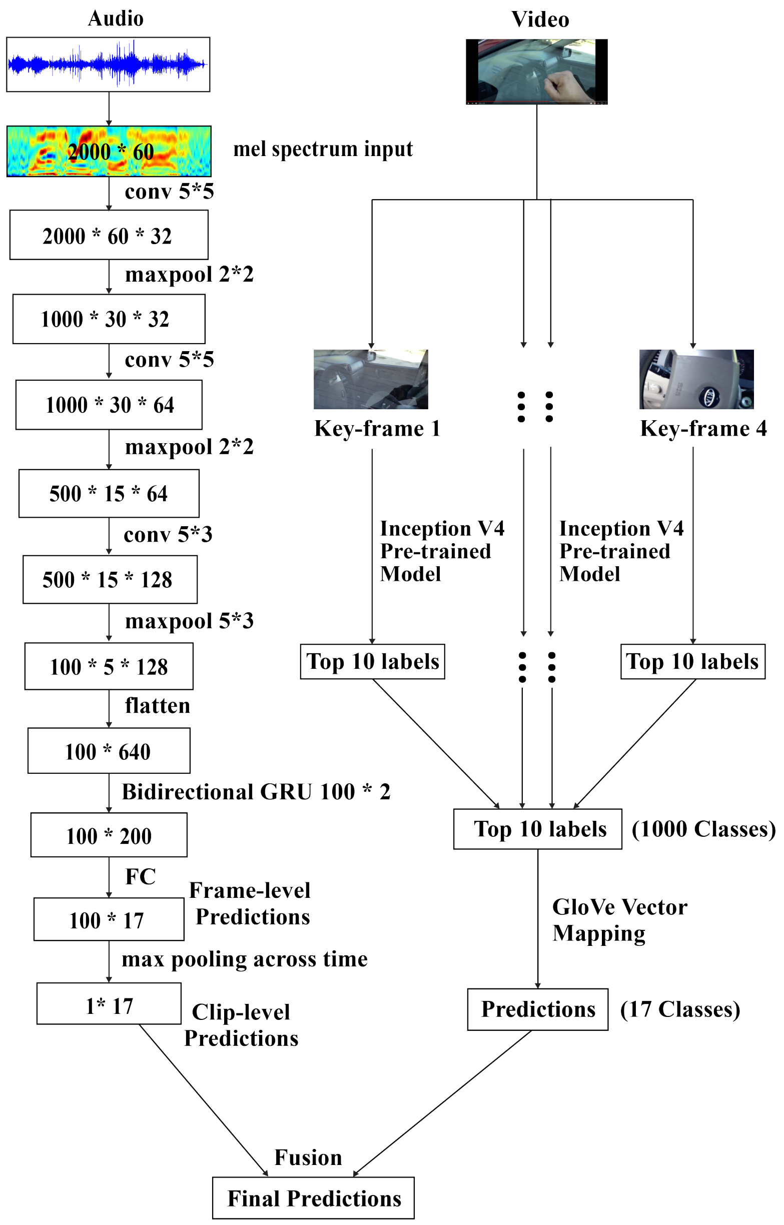


Figure 5.2: Block diagram of our proposed multimodel framework. Note here that both Audio and Video bottleneck features contain dynamics.

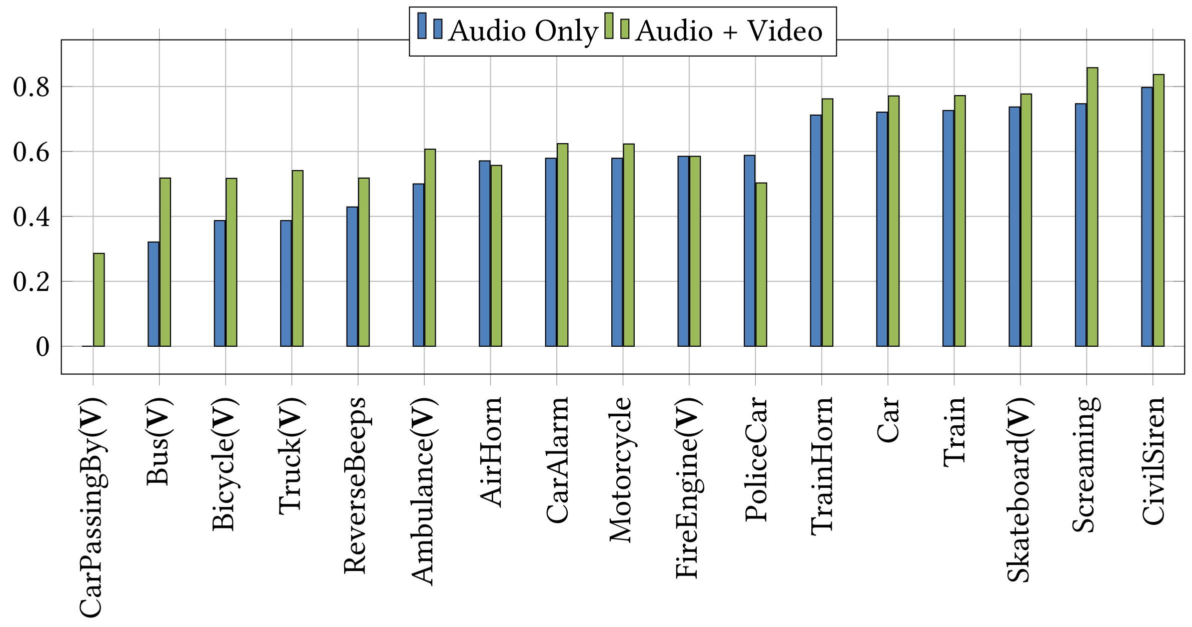


Figure 5.3: (**V**) indicates sound events for which the visual information receives a higher weight in the reranking.

5.3 Experiment

We evaluated our framework on the data for Task 4 of the DCASE 2017 challenge [BSE17], which is the same setup of Chapter 4 Section 4.3.1.

5.3.1 The Baseline Audio Tagging System

The structure of the baseline audio tagging system, which only uses the audio track, is shown in the left branch of Fig. 5.2.

The input to the system is Mel spectral features. We first resampled the raw waveform to 20 kHz, and then divided it into frames of 24 ms (480 samples) with a frame shift of 5 ms. We then applied a 1,024-point Fourier transform to each frame, and aggregate the output into a 60-dimensional Mel spectral feature vector. In this way, each 10-second clip is represented by a 2000×60 feature matrix.

The feature matrix is treated as a two-dimensional image, and passed through a series of convolutional and local max pooling layers. After the pooling layers, the feature map sequence represent an equivalent frame rate down to 10Hz from the original raw frame rate of 200Hz. The output of the last pooling layer is fed into a bidirectional Gated Recurrent Unit (GRU) layer with 100 neurons in each direction. Dropout with a rate of 0.2 was applied after each pooling layer and the GRU layer. Finally, a fully connected layer with 17 neurons and the sigmoid activation function predicts the probability of each sound event type at each frame, and these are aggregated across time using the max pooling function to get clip-level probabilities of the sound events.

We used the stochastic gradient descent (SGD) algorithm to minimize the cross-entropy loss averaged over clips and sound event types. We used a batch size of 95 clips. We applied a Nesterov momentum of 0.8, and a gradient clipping limit of 10^{-3}. The learning rate was initialized to 0.1, and was decayed by a factor of 0.8 when the validation loss did not reduce for 3 consecutive epochs.

We implemented the network using the Keras toolkit [Cho15], and trained it on a single GeForce GTX 1080 Ti GPU. Each epoch of training took 20 minutes, and the model would converge within 6 hours.

5.3.2 Class-Specific Thresholding

The clip-level probabilities predicted by the network must be thresholded to generate binary predictions for evaluation. Due to the imbalance between the sound event classes, the distribution of predicted probabilities may vary drastically across them, and we found it critical to tune the threshold for each class individually. The primary evaluation metric used in the DCASE challenge is the F_1 score, micro-averaged across all sound event types. We devised an iterative procedure to tune class-specific thresholds to optimize the micro-average F_1: (1) tune the threshold of each class to maximize the class-wise F_1; (2) repeatedly pick a random class and re-tune its threshold to optimize the micro-average F_1, until no improvements could be made. We tuned the thresholds on the validation data after each epoch of training and picked the model reaching the highest F_1 on the validation data as the final model. The thresholds obtained from the validation data were directly applied to the test and evaluation data.

5.3.3 Fusion of Audio and Visual Information

We used the procedure described in Section 5.2.1 to estimate the probability of the sound events for each clip using visual information. Each 10-second clip consists of 240 video frames; we selected 4 keyframes among them. Object recognition and knowledge mapping produce the probability of each of the 17 types of sound events.

To verify the quality of the video-only branch, we calculated the F_1 score of each sound event type using its predictions on the validation data. The video predictions outperformed the audio-only baseline system on seven sound events: ambulance, bicycle, bus, car passing by, fire truck, skateboard, and truck.

We combined the outcome of the audio and video branches using different weights: for the seven sound events where the video branch performed better, we assigned a weight of 0.8 to the video branch and 0.2 to the audio branch; for the remaining sound events, we gave a weight of 0.2 to the video branch and 0.8 to the audio branch. These weights were not tuned to the fullest; even better performance may be expected if they were tuned more carefully.

5.3.4 Results and Analysis

System	Development	Test	Evaluation
Audio Only	50.1	54.9	50.6
Audio + Video	60.6	60.9	55.9

Table 5.1: Micro-average F_1 score (in percent) of the audio-only system and the audio-visual reranked system on development, test and evaluation data.

Table 5.1 lists the micro-average F_1 score on the development, test and evaluation set of the baseline audio-only system and the multimodal system. With the help of the visual information, we achieved a remarkable improvement of 10.5%, 6.0% and 5.3% on the three sets, respectively. The final F_1 score on the test set, 55.9%, is the best performance known so far for the audio tagging task of the DCASE 2017 challenge [BSE17].

The contribution of visual information to the test F_1 score of each sound event type is shown in Fig. 5.3. The fusion improved the performance of 14 out of the 17 sound event types. The most significant improvement came from the “car passing by” class. This class is confusable with the “car” class and contains very few positive examples, and was totally missed by the baseline system. Most of the sound event types related to types of vehicles also saw an improvement, because the video tracks clearly show the type of vehicle involved. The “bicycle” class benefited from visual information because many clips show people talking about bicycles, in which bicycles are visible in the video track but not audible in the audio track. Adding visual cues could be a double edge sword as well, where we see classes like “police car" decreasing in performance, since they often appear at the same time in the visual domain as firetrucks and ambulances, introducing noise to the system. The analysis of this result reveals some problems of the data annotation, where the quality of labels of AudioSet is not 100% reliable. This annotation problem will be discussed in Chapter 6: Section 6.4.

We would like to emphasize that the entire multimodel system consumed very few extra computational resources. The CRNN in the audio branch took 6 hours to train on a GPU; the video branch, which only used pre-trained models, took less than 30 minutes to run on the validation, test, and evaluation data.

5.4 Conclusion

In this chapter, we showcased a multimodal framework for audio tagging, which combines information from both the audio and the video tracks of video clips to predict the sound events in the audio track. The system outperforms a strong baseline system on the audio tagging task of the recent DCASE 2017 challenge, boosting the test F_1 score by 5.3%. We also note introducing visual modality could be a double-edged sword, where some classes see drops in performance after mixing with visual cues. This framework is also lightweight compared to other models that use visual information because it makes extensive use of pre-trained state-of-the-art deep learning models and avoids training them from scratch. Reflecting upon this work at the point of drafting this thesis, it still exhibits the potency and effectiveness of late fusion. We will delve more comprehensively into this type of audio-visual system in Chapters 9 and 11. These discussions will include in-depth analyses of the optimal fusion strategy and its inherent robustness characteristics.

Chapter 6

CNN *versus* Transformer Family

6.1 Introduction

In the 2020s, after seeing tremendous success in language tasks[Vas+17], the ML community has been exploring a variety of methods for deploying attention-based architectures, e.g. Vision Transformers (ViT) [Dos+20]—in computer vision and other fields, and competitive performances are reported. Recently, AST [GCG21a] and PSLA [GCG21b] improved the SoTA performance of the AT task[1] on the AudioSet benchmark by leveraging a suite of improvements including DeiT[Tou+21] (distilled ViT) architecture, ImageNet pretraining, data augmentations, and ensemble. However, there is still no clear "winner-takes-all" approach in audio classification tasks that can have the best performance while being efficient.

Audio signals, 1D continuous by nature, require different processing than vision (2D) or language input sequences (discrete). Environmental sounds, compared to well-studied human speech, do not require a language model, but are more diverse and span a wide range of frequencies. Thus, the same techniques that worked well in speech are not guaranteed to work out of the box [Wan18]. Plus, the lack of well-defined strongly-labeled data makes environment sound recognition tasks not only more challenging but also understudied so far.

In Chapter 2, we identified CNN's superiority over MLPs, and RNNs. CNNs [For+19; Kon+19a] and its variant CRNNs [WLM19] have become the de facto architecture for acoustic event recognition tasks and have dominated the AudioTagging leaderboard until the rise of ViT [GCG21a]. The major difference between convolutional models and attention-based networks (e.g. ViTs) is the locality inductive biases. In a nutshell, the convolutional neural network sweeps through every consecutive pixel with its learned kernel, which is perfect

[1]Audio Tagging (AT) task aims to characterize the acoustic event of an audio stream by selecting a semantic label for it.

for learning *local features*[Li+17], but cannot see beyond its receptive field; whereas ViT networks skip through, attend between patches to build *global correlations*, and sometimes not even rely on the positional information.

In this chapter, we seek to thoroughly understand the difference between using each type of model on the AT task. We train 4 variants of transformer networks including CNN+Transformer, Vision Transformer (ViT), Transformer, and Conformer on the task of Audio Tagging, and compare them with ResNet, CRNN control group. Between these six architectures, we perform analysis on the largest available dataset: Google AudioSet [Gem+17b]. We also perform an analysis regarding different architectures' efficiency by training them on the Speech Commands dataset [War18]. Chapter contributions:

1. We systematically compared the performances of different transformer variants between several CNN variants under a permutation of different settings on the same large-scale dataset Audioset (e.g. w/. or w/o. pretraining. lr scheduling). Our experiments suggest that pretraining is *not* always necessary, LR scheduling & data augmentation are always helpful.

2. Our experiments shed light on critical optimization strategies & tradeoffs, e.g. feature size, LR schedule, batch size, momentum, normalization, and loss landscape, which have not been thoroughly explained previously.

3. We also compare five Convolutional Neural Network (CNN) architectures and one pure Transformer architecture optimized for edge deployment, train them for wake-word detection on the Speech Commands dataset [War18], and quantize two representative models. This is covered in § 6.9.

6.2 Tabular view of State-of-the-art Audio Event Detection Systems

Same as the setup in Chapter 4, reiterating for reader's convenience, **AudioSet [Gem+17b]** contains 2,042,985 10-second YouTube video clips, summing up to 5,800 hours annotated with 527 types of sound events (weak label[2]). The same group [Her+21] conducted quality assessments of these labels ranging from 0-100%. The *full* trainset has 2 subsets: classwise *balanced* set (22,176 samples) and *unbalanced* (2,042,985 samples) set, and *eval* set with 20,383 samples [Her+17b].
AudioTagging Benchmark: The left columns of Table 6.1 list state-of-the-art *single models* and training procedures for the AT task trained on the *full* AudioSet and test on

[2]does not specify which second specific event happens

	Previous approaches						Our Implementation			
Model	CNN Variants				Attention		see Table 6.2			
Procedure	PANNs CNN14 ([Kon+19a])	ResNet [For+19]	PSLA [GCG21b]	ERANNs [VBV21]	Conformer [Sri+21]	AST [GCG21a]	A1	A2	A3	A4
params	42.2M	23M	13.6M	54.5M	88.1M	88M	see Table 6.2			
train dataset	1934187	1953082	1953082	1803891	2063949	1953082	1998999		1998999	
eval set	18887	19185	19185	17967	20371	19185	20126		20126	
feature size	64×1001	64×1000	128×1056	128×1280	64×500	128×1024	128×1024		64×400	
pretrained	✗	✗	ImageNet	✗	SSL	ImageNet	ImageNet	✗	ImageNet	✗
epoch*	10	50	30	9	100	5	10	10	10	10
batch size	32	n/a	100	32	640	12	20	400	80	448
optimizer	adam	adam $_{0.95-0.999}$	adam $_{0.95-0.999}$	adam	adam $_{0.9-0.98}$	adam $_{0.95-0.999}$	adam	adam	adam	adam
maxLR	0.001	0.0001	1.0E-04	0.001	3.0E-04	1.0E-05	1.0E-05	4.0E-4	1.0E-05	4.0E-4
LR decay	✗	✗	step	one-cycle	linear	step	step	step	step	step
decay rate	✗	✗	0.5	[ST19]	3.00E-6	0.5	0.5	0.5	0.5	0.5
decay epochs	✗	✗	5	cyclic	100	2	2	2	2	2
weight decay	✗	5.0E-07	5.0E-07	✗	0.01	5.0E-07	✗	✗	✗	✗
warmup steps	✗	✗	1000	✗	10k	1000	1000	1000	1000	1000
dropout	✓	✓	✗	✗	0.1	✗	✗	✓	✗	✓
databalancing	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
mixup	✓	✗	✓	modified	✓	✓	0.3	0.3	0.3	0.3
TimeSpecAug	✓	✗	✓	✓	timeonly	✓	t192,f36	t192,f36	t75,f12	t75,f12
label enhance	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
normalize	✗	✗	x2	✗	✗	x2	x2	x2	x2	x2
train time	3 days	n/a	a week	n/a	n/a	a week	108 Hrs	16.45Hrs	21.0Hrs	8.84Hrs
GPU	V100×1	n/a	titanX×4	n/a	n/a	titanX×4	V100×4	V100×4	V100×2	V100×2
Best mAP	0.431	0.392	0.439	0.450	0.415	0.448	0.430	0.437	0.410	0.411

Table 6.1: Different training procedures for single-checkpoint Audio-Only models trained on the full AudioSet as of March 2022. SSL[Sri+21]: pre-trained on Self-supervised task using 3.9M (67k hours) proprietary data. To standardize reporting difference in steps *versus* epochs, epoch* here means 1 full iteration of the TrainSet. Adam optimizer's default β_1 =0.9,β_2=0.999, the ones without specified β are default ones. LR: learning rate. TimeSpecAug[Par+19]: t, f indicate the max length of time and frequency mask, ✓means specific setups are unknown. Label enhancement[GCG21b] involves altering the original labels. CNN14 [Kon+19a] achieves 0.442 mean average precision (mAP) using 128 mel bins. We are aware of PaSST [Kou+21b], due to its high similarity to AST except for dropout/ensemble, we do not separately list it.

the *eval* set [3]. Here, we *do not* include *multi-modal* or *ensembled* models, which would introduce tremendous extra variability, making fair comparison almost impossible [4]. Unlike some previous works, we *do not* list models trained only using the *balanced* subset since it only accounts for 1% of the AudioSet training set, which are 10-20% mAP worse than

[3]Previous benchmarks are excerpted from the original publications

[4]We consider weight-averaging(WA)[Izm+18] an implicit ensemble, hence we report single-checkpoint score w/o WA: AST [GCG21a] 0.459⟶ 0.448, and PSLA [GCG21b] 0.444 ⟶ 0.439. WavegramCNN in PANNs [Kon+19a] is also an implicit ensemble, thus leaving out of the comparison.

models trained on the *full* set.

6.3 Experiments & Results

We explored many variations with different optimization techniques, data augmentations, choice of regularization, and a reasonable amount of grid search for the hyperparameters. We offer 4 different training procedures with different costs and performance that cover different typical use cases, see table 6.1.
Procedure A1: aims at reproducing the best SoTA performance. Therefore, it is the longest in terms of training time using the larger feature.
Procedure A2: is to test whether we could reach SoTA without pre-training, except with larger batch size and LR.
Procedure A3 & A4: aim at matching SoTA performance using 5.12x($\frac{1}{2}$ #Mels, $\frac{1}{2.56}$ less time resolution) smaller features thus a lot faster. It can be trained on average 6x faster and could be a good setting for exploratory research.

Model	#param	Best mAP		Best mAP	
		A1	A2	A3	A4
AST/ViT	87.9M	**0.430**	0.274	**0.410**	0.268
Transformer	28.5M	-	0.230	-	0.209
CNN+Trans	12.1M	-	**0.437**	-	**0.411**
Conformer	88.1M	-	0.335	-	0.308
ResNet50	25.6M	0.410	0.399	0.382	0.370
CRNN	10.5M	-	0.429	-	0.406

Table 6.2: A1, A3 contain blanks since ImageNet pre-trained weights are only available for ResNet50 and AST/ViT. **Bolded:** the models reported in Table 6.1

.

AST/ViT (Table 6.2, Figure 6.1(C)). uses pre-trained weights from the DeiT-base-384 [Tou+21] model imported from the timm library [WTJ21]. In order to preserve the learned positional embedding in the pre-trained DeiT [Tou+21] model (24×24=576 patches), we choose the same 16×16 patch-size for all our AST experiments. We did a comparison on stride sizes as shown in Table 6.3. e.g. A3 procedure uses strides of 8 (8 pixels overlap), resulting in (7×49=343) patches. Shorter strides benefit smaller features more than larger features, but there is a catch: Computation cost grows quadratically due to the $O(p^2)$ attention mechanism, p being the number of patches.
Transformers (Table 6.2) We implemented pure transformers without convolution or ViT type patches, which takes in logMel spectrogram as input. Since the Transformer layer

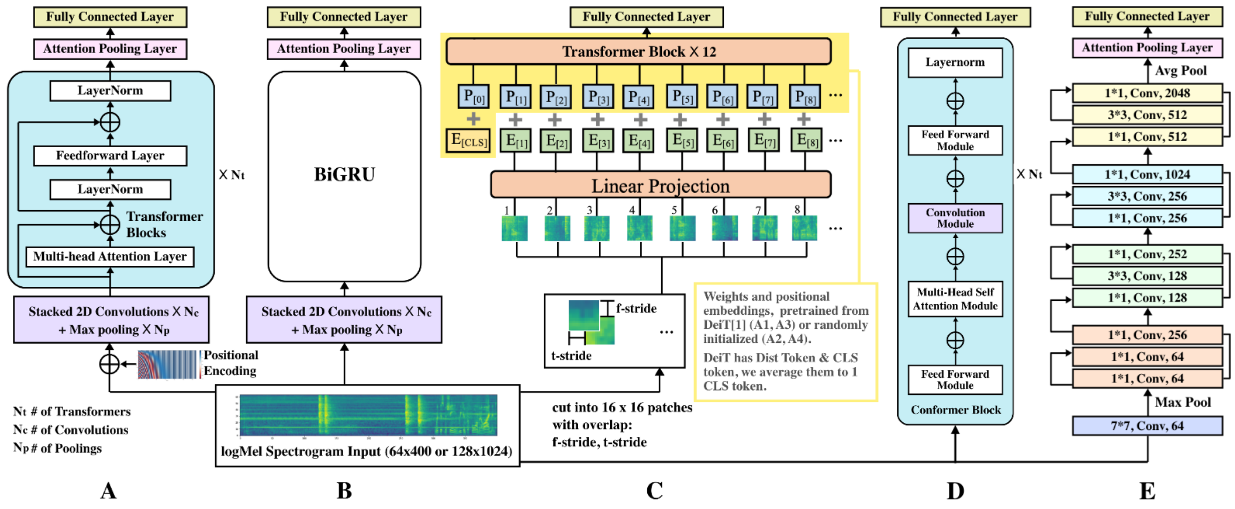

Fully Connected Layer
Attention Pooling Layer
LayerNorm
Feedforward Layer
Transformer Blocks
Multi-head Attention Layer
X Nt
Stacked 2D Convolutions X Nc + Max pooling X Np
Positional Encoding
Nt # of Transformers
Nc # of Convolutions
Np # of Poolings
BiGRU
Transformer Block X 12
Linear Projection
f-stride
t-stride
Weights and positional embeddings, pretrained from DeiT[1] (A1, A3) or randomly initialized (A2, A4).
DeiT has Dist Token & CLS token, we average them to 1 CLS token.
cut into 16 x 16 patches with overlap: f-stride, t-stride
logMel Spectrogram Input (64x400 or 128x1024)
Layernorm
Feed Forward Module
Convolution Module
Multi-Head Self Attention Module
Conformer Block
Avg Pool
Max Pool
7*7, Conv, 64
A
B
C
D
E

Figure 6.1: (A): CNN+Transformer (B): TALNet[WLM19](CRNN) (C): AST[GCG21a]/DeiT[Tou+21](ViT) (D): Conformer (E): ResNet

Procedure	tstride×fstride	4×4	6×6	8×8	10×10	12×12	16×16
A1 128×1024	mAP	-	-	-	**0.430**	0.429	0.421
	#patches	8192	3570	2048	1212	756	512
	Hrs/Epoch	-	-	-	10.8	6.19	5.53
A3 64×400	mAP	**0.414**	0.408	0.410	0.370	0.320	0.24
	#patches	1485	390	343	195	128	72
	Hrs/Epoch	12.67	2.45	2.1	1.44	1.01	0.8

Table 6.3: time, freq stride influence AST's mAP and train time, blanks are the experiment could not fit into GPUs.

is temporal agnostic, we introduce positional encoding to retain the temporal order of inputs. To add positional encoding to the audio model, we adopted the classic positional encoding [Vas+17], which is defined as follows, where d represents the dimension of the input, pos is the position in time and i is the dimension index in the input tensor.

$$\mathbf{PE}_{(pos,k)} = \begin{cases} \sin\left(pos/10000^{2i/d}\right) & k = 2i \\ \cos\left(pos/10000^{2i/d}\right) & k = 2i+1 \end{cases}$$

We added positional encoding to spectrograms before feeding into CNN layers and we scaled up the original input. Given the input x: $n \times d$, we scale up the input by square root of the input dimension: $x = x \cdot round\lfloor\sqrt{d}\rceil + \mathbf{PE}(x)$

The overall architecture is depicted in Figure 6.1(A) but without the stacked 2D conv + pooling layers. The resulting mAP is not as high as the AST/ViT type. In Table 6.2, we report the best performing one with $N_t = 8$ layers of Transformer blocks with MultiHeadAttention (MHA) layers (8 attention heads) after testing out $N_t = 2, 4, 6, 8, 10, 12$ with 4, 8, 12 attention heads set-ups. The same as [Kou+21b], to train faster and generalize better, we implemented dropout layers within the MHA layers. mAP for other setups is too low for meaningful comparisons. Transformers suffer from the same $O(n^2d)$ cost issue as ViT, n being the input length. Therefore, training for Transformers is quadratically slower than models with the same number of parameters without the attention mechanism.
CRNN (Table 6.2, Figure 6.1(B)) We follow the well-tuned TALNet architecture using $N_c = 10$ convolution layers with 3×3 kernels and $N_p = 5$ max-pooling layers in between, resulting in embedding size of 1024 feeding into the biGRU layer. The output of GRU is fed to a fully-connected layer of size 527 to predict frame-wise probabilities. Finally, the frame probabilities are aggregated with a pooling function to make the final prediction. We did not tune hyperparameters for TALNet $\sim O(md^2)$ where m is the stacked convolution output sequence length, d is the representation dimension.
CNN+Transformer (Table 6.2, Figure 6.1(A)) The first part of Stacked Conv blocks and Pooling is the same as the CRNN we implemented, which uses $N_c = 10$ Conv layers and $N_p = 5$ pooling layers. We replace the GRU layer with the aforementioned Transformers[5]$\sim O(m^2d)$, and experimented with $N_t = 1, 2, 4$ layers of Transformer layers with 4,8,12 attention heads. We use the same positional encoding as our aforementioned Transformer model. Here, we report the best performing $N_t = 2$ layers of transformers with 8 attention heads, followed by the attention pooling layer to aggregate frame probabilities.
ResNet (Table 6.2) Following [For+19], we replace the standard ResNet50's last layer with attention pooling layer to output 527 classes probabilities. We also implemented ResNet34, but since ResNet architectures are from the same family$\sim O(knd^2)$, k : kernel size, we only list ResNet50 in Table 6.2.
Conformer (Table 6.2) We implemented the same large conformer as [Sri+21]$\sim O(n^2d)$ using 12 conformer blocks with 768D encoder embeddings and 12 attention heads, as this setting was reported to be the best-performing setup for conformer architecture. Each conformer block is shown in Figure 6.1(D).

6.4 Data Quality & Data Efficiency

Missing files: In table 6.1, due to the downloading difference of AudioSet, the number of train&test set varies *a whopping* ±5% across previous works, especially the different test size could cause severe fluctuations in final mAP reporting as seen in Figure 6.2. e.g. one could have downloaded the lower-label-quality test samples that tank their score. We

[5]To avoid repetition, m, n, d are shared among different models

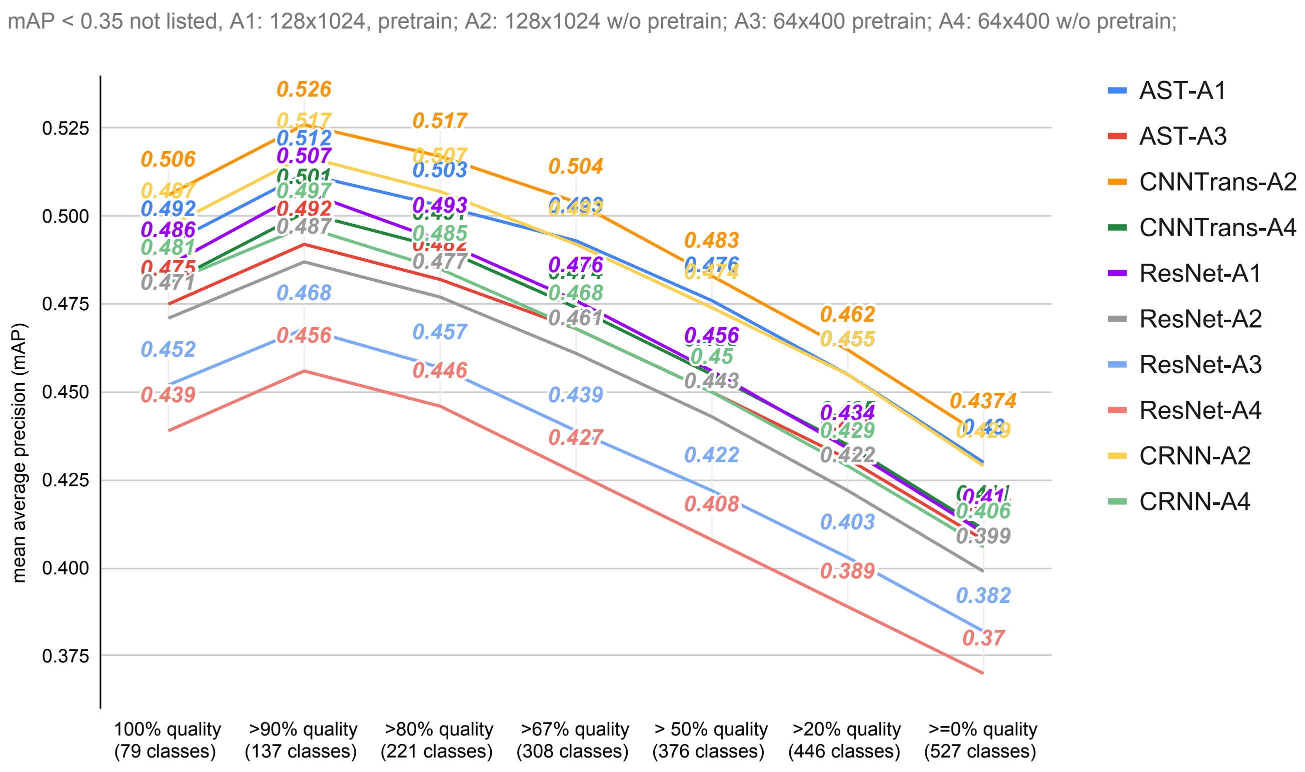


Figure 6.2: Performance(mAP) fluctuation on quantiles of data with varying label quality.

release our test set labels along with our implementation for consistency.

Speedup: As we can see in Table 6.1, previous approaches benefited from using larger features. In Table 6.3 we see dramatic speedup using 64(#mels)×400(time) logMel spectrogram[6] *versus* 128(#mels)×1024 filter bank features, this 5.12 times feature size reduction results in more than 6 times speedup. Note that *half of* the pipeline efficiency (for both train & inference) depends on the data efficiency, the rest is then about model efficiency. Using our smaller feature would result in 10^{-1} reduction of data time per sample. Therefore, we highly recommend our A3, A4 procedures for research or inference tasks, which could easily fit into a single 16GB-GPU with a lot less carbon footprint. Indeed, we trade off 2 points of mAP for 6 times speed up. To analyze the performance loss due to feature tempo-

[6]The waveform is downsampled to 16 kHz; frames of 1,024 samples (64 ms) are taken with a hop of 400 samples (25 ms); each frame is Hanning windowed and padded to 4,096 samples before taking the Fourier transform; the filterbank of 64 triangle filters spans a frequency range from 0 Hz to 8 kHz.

ral/frequency resolution loss, we took a closer look at different label quality quantiles of the data. As Figure 6.2 shows, larger features outperform smaller features in high-quality classes by 2±0.1% mAP, indicating models benefit, but very limited, from higher freq/time feature resolution.

6.5 Impact of Pretraining

Similar to the conclusion of [HGD19], we find ImageNet pretraining helps certain architectures to converge but is not a must. As shown in Table 6.2, CNN-Trans model *trained from scratch* can *outperform* AST models that are pre-trained on ImageNet and also be better than the Conformer model pre-trained using SSL in [Sri+21]. CRNNs are also competitive. We observe AST models need pre-trained weights to converge to optimal, cold-start weights/positional embeddings from random Gaussian results in a lot lower performance. This is maybe due to a huge distribution shift from Gaussian to optimal weight distribution.

6.6 Insights regarding Optimization Process

LR scheduling is crucial for training a high-performance model. The proven strategy is to scale **LR** $\propto$ **batch size** [Smi+18] when amplifying batch size, and to fill the GPU memory till full with large enough batch size since increasing batch size linearly speeds up training time. **LR decay** is also important and can be seen as simulated annealing [Smi+18]. We view Adam/SGD training as $\frac{dw}{dt} = -\frac{d\mathcal{L}}{dw} + \eta(t)$ where $\mathcal{L}$ is the BCE loss function in our case, summed over all training examples, and w denotes the parameters. $\eta(t)$ denotes Gaussian random noise updating in continuous "time" t towards convergence, which models the effect of estimating the gradient using mini-batches. In [Smi+18], they showed that the mean $\mathbb{E}(\eta(t)) = 0$ and variance $\mathbb{E}(\eta(t)\eta(t')) = gF(w)\delta(t-t')$, where $F(w)$ describes the covariance in gradient fluctuations between different parameters. They also proved that the "noise scale" $g = \epsilon(\frac{N}{B} - 1)$, where ϵ is the learning rate (LR), N the training set size and B the batch size. This noise scale controls the magnitude of the random fluctuations in the training dynamics. Intuitively, the initial noisy phase allows the model to explore a larger fraction of the parameter space without getting trapped in local minima. Once we find a promising region of parameter space, we reduce the noise to fine-tune the parameters. In Table 6.1 and all our experiments, we observe LR decay outperforms constant LR by at least 2 % mAP. We also implemented cyclic schedule [ST19] used by [VBV21], but the result is suboptimal *versus* step decay. Meanwhile, annealing the temperature in a series of discrete steps can sometimes trap the system in a "robust" minimum, in our experiments, this severely impedes training attention-based models such as AST, transformers. e.g when we initialize training AST in our A3 procedure with a non-optimal LR: say 1E-6, 1E-4, the model's mAP gets "stuck" below 0.1. This effect is less severe for non-attention or hybrid models. We postulate the culprit is the *sharp loss landscape* of the attention-based models [CHG22], when the gradient accumulation cannot adapt to changes in the loss landscape, training would be impeded since we would keep on exploring the wrong direction in parameter space. This also explained why our training also failed when we used *gradient accumulation* step size$\geq$ 2.

In Table 6.1, we can see some previous works have tuned the β_1 of Adam to 0.95, this is actually letting gradient updating **twice** as slower than the default 0.9. $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ where m_t is the first moment of the gradient and g_t is the gradient at t step. This trick might compensate for the aforementioned sharp-loss-landscape phenomenon, but we did not achieve the same performance gain using the trick, hence sticking to default. Some previous works advocate *regularization* for better model generalizability by adopting weight decay or dropout, we observe 1-2% mAP performance drop using regularization in our experiments and therefore did not include them in our procedures. We note [Kou+21b] used AdamW [LH17] optimizer. AdamW itself was designed to fix weight decay within the Adam update iteration, we do not use regularization in this chapter, hence AdamW is not studied here.

6.7 Data Augmentations & Normalization & Balancing

AST and PSLA [GCG21a; GCG21b] applied an unusual normalization (Normalize*) to the feature input: $x = \frac{x-\mu}{2\sigma}$, which results in the normalized feature with $\mathbb{E}(x) = 0$, and $var(x) = 0.25$. In our experiment, Training AST using the input of $var(x) = 1$ or $var(x) = 0.0625$ would lead to mAP of 0.09, 0.02 respectively. The latter phenomenon can be explained by vanishing gradients, whereas the former behavior is likely due to the training noise $\eta(t)$ (§6.6) being too large. When the input variance is too large, the training would continue exploring suboptimal regions of the parameter space. Normalization remains a key step in modern deep learning systems as it is noted in both work along the line of Batch Normalization and Layer normalization. We will discuss once again in chapter13 how this affects the audio generative modeling.

Previous works in Table 6.1 reported that data augmentation tricks improve performance across all models/procedures. Table 6.4 shows their specific impact after ablation. Note that Mixup[Zha+18], TimeSpecAug[Par+19] shifts the input distribution's mean and variance, the higher the mixup coefficient/the longer time/freq mask$\longrightarrow$the lower variance of the input$\longrightarrow$more influence on end performance. We also compared another augmentation cutMix [Yun+19], which marginally outperformed the mixup method. Overall, we find data balancing helps, but due to the stochasticity(random sampling/shuffling) of the vanilla (no-balancing) train loader, its influence also fluctuates.

Balanced Sampling: In our experiments, as we observe in Table 6.4, balanced sampling would improve the performance, and this is in line with [GCG21b]. However, a recent study [Moo+23] challenged this view. It is difficult to reproduce the results in [Moo+23] since the held-out eval set is not public, but it is worth noting that different distributions of the training data could lead to different levels of capability to generalize to unseen data.

Augmentation	data balancing	Mixup	SpecAug	Normalize*	CutMix
mAP drop(%)	3.9±2.0	1.5±0.8	1.5±0.3	16.5±9.0	2.0±1.1

Table 6.4: Data Augmentation's influence on mAP through ablation, on all models trained with A4 procedure

6.8 Discussion of Metric

The case of huge performance fluctuation in § 6.4 also serves as an example where using mAP as the sole metric could potentially distort the impression we got from the results. In fact, there are also discussions in [Moo+23] that motivate to adopt an alternative of mAP such as d-prime (in-class discriminability) or label-rank probability, or other metrics that factors in model calibration, e.g. ECE (mentioned in § 2.2.5). To gain a better view of the problem, let us revisit the math first. The Mean Average Precision(mAP) is calculated as the mean of the Average Precision (AP) over all queries or all instances, and AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are the precision and recall at the nth threshold. Average Precision (AP) is used to summarize a Precision-Recall curve, and it provides a single value measure that represents the overall quality of the model across all threshold levels. However, it is worth noting that the AP is sensitive to class imbalance (this is the case of AudioSet), and it puts more emphasis on retrieving all positive examples (recall).

It's widely understood that varying thresholds lead to a balance or trade-off between precision and recall. By summarizing this curve with a single value (the AP), we can compare different models. But, we lose information about how well each model does at specific thresholds or the shape of the precision-recall curve (scenarios where we care about the model performance when it's high-precision or high-recall).

On the other hand, AudioSet[Gem+17b]'s other official metric ROC AUC, is a graphical representation of the true positive rate (TPR, or Recall) against the false positive rate (FPR) at various threshold settings. The area under the ROC curve (AUC) gives a measure of the classifier's ability to distinguish between positive and negative classes. The ROC curve is generally used when the observations are balanced between each class. The slightly different but different AUC is PR AUC: The area under the Precision-Recall curve is a plot of Precision (positive predictive value) against Recall (TPR). There are cases when a model can have a high ROC AUC while the PR AUC is low, indicating poor precision or recall. This can happen when the negative class dominates, as the ROC curve can be *overly optimistic with imbalanced data.* Conversely, the PR curve will be more sensitive to differences in these rates for the minority class (the positive class in many imbalanced datasets).

AudioSet[Gem+17b]'s 3rd official metric d' provides a single-number summary of the receiver operating characteristic (ROC) curve. A higher value of d' indicates better discriminability. If $d' = 0$, the system or the observer cannot discriminate signal from noise at all. If d' is negative, it suggests that the system or observer is systematically wrong. In the specific implementation, d' is calculated by applying the percent point function (inverse of the cumulative distribution function) of the standard normal distribution to the ROC AUC: $d' = \Phi^{-1}(\mathrm{AUC}) \cdot \sqrt{2}$, where Φ^{-1} is the inverse of the cumulative distribution function of the standard normal distribution, often referred to as the quantile function or percent-point function. It is multiplied by $\sqrt{2}$, based on the scaling of the AUC ROC, which ranges from 0.5 (random guessing) to 1 (perfect classification). Note that this calculation is a rather rough approximation and can be applicable under the assumption that both distributions are normal with equal variances.

Using other model calibration-aware metrics like label-rank probability, ECE can significantly improve the trustworthiness and usefulness of a model's predictions. However, model calibration can add an extra layer of complexity and computation time to the training process. It often requires additional steps such as temperature scaling, Platt scaling, or isotonic regression. It is important to note that calibration should be performed using a separate validation set, not on the training or test sets, to avoid overfitting and to give a realistic assessment of the model's performance. Also, calibration will not usually improve the model's ability to rank predictions (e.g. it will not improve the AUC-ROC).

As we can see, each metric is shipped with trade-offs, we focus mainly on mAP throughout this thesis, as it is the most widely-established metric in the audio research community that allows for apple-to-apple comparison. We further our discussion through the lens of the confusion matrix in the top-k case in Chapter 12.3.

6.9 Efficiency Accuracy Tradeoff GPU *versus* CPU for CNNs *versus* Transformers

Having grasped the variations in the training pipelines of different models, we also set a secondary objective to scrutinize these models in real-world settings and evaluate their efficiency. We observe that the over-the-air test accuracy of trained models on parallel devices (GPU/TPU) usually degrades when deployed on edge devices using a CPU for over-the-air, real-time evaluation [Ryb+20]. Further, the differing inference time when migrating between GPU and CPU varies across models. This drop is due to hardware latency and acoustic impulse response, while the non-uniform expansion of inference time results from varying exploitation of hardware acceleration by architectures. We seek to quantify their accuracy-efficiency tradeoffs to inform researchers and practitioners about the key components in models that influence this tradeoff.

Dataset and Setup: We trained eight wake-word detection models using six different architectures on wake-word audio samples from the Google Speech Commands dataset [War18] with PyTorch: VGG19_bn, DSCNN, EfficientNet_b1, EfficientNet_b7, EfficientNetV2_m, EfficientNetV2_xl, ResNet50, and Transformer[BOC21]. All architectures were non-streaming[7]. We also used PyTorch Eager Quantization [Con22] to perform static PTQ on VGG19_bn and DSCNN. Here, we compared quantization in families of models - VGG-based models, represented by VGG19_bn (VGG19_bn, ResNet), and MobileNet-based models, represented by DSCNN (DSCNN, EfficientNetV1, EfficientNetV2). Digital testing analyzed models' performance on 807 audio files from the Speech Commands dataset, of which 187 contained the wake word.

Over-the-air Test: We performed over-the-air testing to analyze overall model performance and layer-wise latencies. Three subjects conducted testing, two with lower voices, and one with a higher voice. All trials took place at the same location in the same room, ensuring a consistent room impulse response across trials. During over-the-air testing, subjects completed five trials for each model on a GPU platform and then a CPU platform. Table 6.5 details the results from analyzing digital versus over-the-air performance for all trained models on GPU and CPU devices. Digital Test F1 reflects the F1 score calculated by checking the correctness of models' predictions for whether or not an input contained the wake word. Over-the-Air Test F1 reflects the F1 scores calculated from individual subjects' analysis of model prediction correctness for each trial. Table A.2 (Appendix) breaks model leaf node layers into nine operational categories across the eight examined models: Conv, BN, ReLu, MaxPool, Linear, Dropout, AvgPool, LayerNorm, and GELU. We used PyTorch hooks to calculate the average latency for each layer category, and the THOP library [Zhu22] to calculate the average FLOPS for each layer category.

[7]We used the same definition of streaming and non-streaming in [Ryb+20].

GPU (Mac-M1)	Digital F1	Over-the-Air F1	Latency
VGG19_bn [Ryb+20]	**93.37**	**96.69** ± 2.99	**17.3**
DSCNN [Ryb+20]	92.15	93.78 ± 5.61	**37.6**
EfficientNet_b1 [TL19]	92.75	93.16 ± 4.90	85.5
EfficientNet_b7 [TL19]	**93.87**	**96.55** ± 2.59	146.0
EfficientNetV2_m [TL21]	92.88	92.14 ± 5.60	94.0
EfficientNetV2_xl [TL21]	92.64	**94.06** ± 2.08	146.3
ResNet50 [He+16a]	91.86	90.52 ± 7.48	**56.5**
Transformer [BOC21]	**93.87**	91.47 ± 4.56	65.8
CPU (Raspberry Pi 4B)	**Digital F1**	**Over-Air F1**	**Latency**
VGG19_bn [Ryb+20]	**93.37**	**90.80** ± 6.54	372.0
DSCNN [Ryb+20]	92.15	**90.28** ± 4.02	**151.0**
EfficientNet_b1 [TL19]	92.75	89.83 ± 4.31	**283.0**
EfficientNet_b7 [TL19]	**93.87**	89.20 ± 5.20	1044.0
EfficientNetV2_m [TL21]	92.88	**93.45** ± 6.61	808.0
EfficientNetV2_xl [TL21]	92.64	71.85 ± 5.54	1673.0
ResNet50 [He+16a]	91.86	90.18 ± 10.37	324.1
Transformer [BOC21]	**93.87**	88.96 ± 11.30	**182.6**

Table 6.5: Model performance on Speech Commands Dataset. F1 scores are scaled to 100. Bold font highlights better latencies.

Preliminary Observations: VGG19_bn and EfficientNet_b7 achieved the highest F1 scores during over-the-air GPU testing, surpassing the larger EfficientNetV2_xl model by about two points (Table 6.5). VGG19_bn also achieved the lowest latency. *VGG19_bn's success is likely linked to the efficiency of its convolution method on GPU.* Moving from GPU to CPU, over-the-air performance dropped for many architectures. All models except for EfficientNetV2_m saw their F1 scores slip below the digital baseline, consistent with previous research [Ryb+20]. The EfficientNetV2_xl model in particular showed a steep decline in performance, accompanied by a large increase in latency (Table 6.5). *EfficientNetV2_xl's drop in performance is likely a result of its increased latency, in turn, caused by its convolution method's inefficiency on CPU.* DSCNN achieved the lowest latency, enduring only a small performance drop. The Transformer maintained a relatively low latency across both GPU and CPU, though its accuracy dropped moving to CPU. In the discussion below we drill into the fine-grained components of each architecture to elaborate on these italicized sections.

6.9.1 CNNs *versus* Transformers' Efficiency Difference

Figure 6.3 breaks down individual network components, analyzing their Percentages of Aggregate Runtime (PAR), the relative time models spent computing each type of layer.

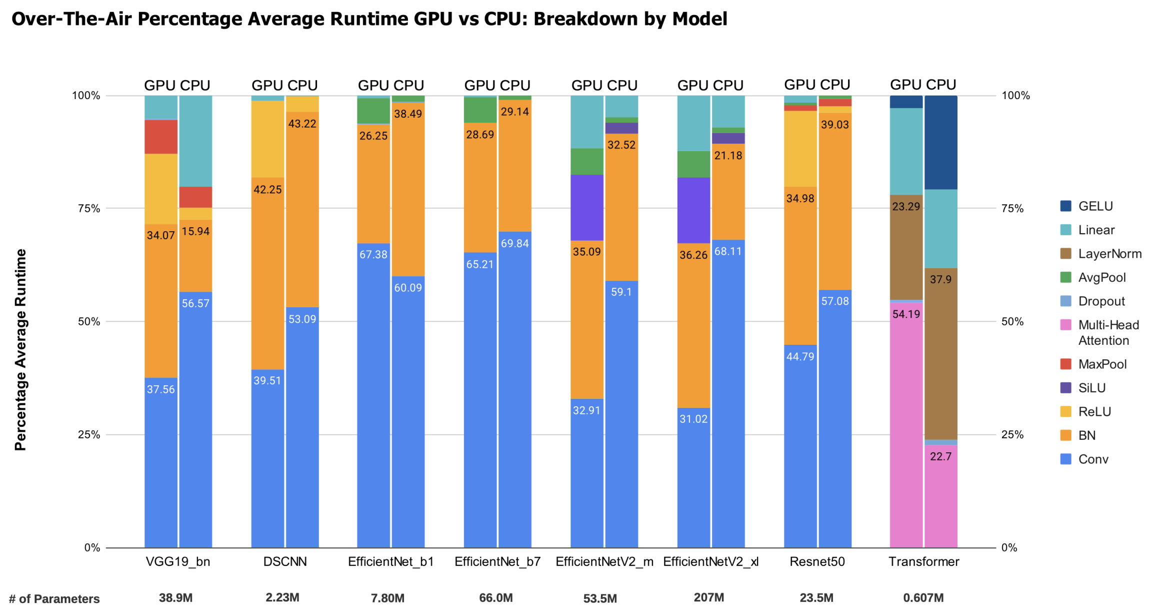


Figure 6.3: Percentage of Aggregate Runtime (PAR) of different architectures on GPU *versus* CPU, PAR measures the percentage of a model's total runtime spent computing layers in a specific category across trials. Contrast to Table A.2

Matrix Multiplication in Convolution: Convolution layers in particular often take longer to compute on CPU as opposed to GPU. This divergence stems from hardware optimization differences for matrix multiplication. Hardware optimization strongly affects the efficiency of convolution layers because most of the convolution's computation comes from matrix multiplication. Convolution is computed as follows: Let x be the input matrix of dimensions $h \times w \times c_{in}$, where h is image height, w is image width, and c_{in} is the number of input channels. Let W be a matrix of weights with dimensions $c_{in} \times c_{out} \times k \times k$ where c_{out} is the number of output channels and k is the kernel size. For each input channel, these matrices x and W are multiplied together. The sum of these multiplications yields an output matrix z of dimensions $h \times w \times c_{out}$. The exact formula is shown below, resulting in a total cost of $O(h \cdot w \cdot c_{in} \cdot c_{out} \cdot k^2)$, where r is a specific input channel and s is a specific output channel:
$z[:,:,s] = \sum_{r=1}^{c_{in}} x[:,:,r] \times W[r,s,:,:]$
Note that only the forward pass is involved in wake-word detection; there is no back-propagation.

Matrix Multiplication Optimization: The parallelization abilities of CPUs are limited by their low numbers of cores and caches. GPUs, however, are equipped with a larger array of cores and caches to support the acceleration of parallelizable computations, including matrix multiplication. Matrix multiplication can be broken down via tiling [Nvi22], enabling one large matrix multiplication to be calculated as the multiplication of a series of much smaller, parallelizable chunks. This strategy helps avoid costly memory accesses. GPUs can take advantage of this parallel character to significantly speed up matrix multiplication. For most of the models we tested, convolution took a larger share of time to execute on CPU versus on GPU (See Table A.2 Conv rows and PAR columns). Parallelization and tiling offer a good explanation for this pattern.
MBConv Blocks [San+18] versus Vanilla CNNs: The cost of convolution on GPU versus CPU can also be affected by model architecture. Models with MBConv blocks (like DSCNN, EfficientNet, and EfficientNetV2) use pointwise convolutions, forcing depthwise convolutions to be done in sequential order. This method reduces the cost of convolution to $O(h \cdot w \cdot c_{in} \cdot (c_{out} + k^2))$ [San+18]. Notably, the sequential computation design of MBConv blocks reduces their parallelizability, and consequently, MBConvs can only receive a minor latency boost on GPU. Yet moving to CPU, they show less slowdown and reap the benefits from requiring fewer operations (Table A.2 shows about 10x slow down GPU versus CPU in Conv latency). Meanwhile, models using vanilla CNN blocks (like VGG19_bn) enable greater parallelization and therefore high GPU performance but suffer a more severe slowdown transitioning to CPU. Table A.2's latency columns show that the VGG19_bn model's vanilla convolution layers saw an exponentially larger latency increase (100x GPU versus CPU) compared to models using MBConv blocks (DSCNN, EfficientNet, EfficientNetV2). This difference clarifies that theoretical algorithm complexity alone is not sufficient to determine an algorithm's practical efficiency on GPU; parallel algorithm complexity must also be accounted for.
Accelerating Convolution: Convolution receives significant acceleration on GPU devices. As CPUs cannot take advantage of parallelism to accelerate matrix multiplication, convolution is often much slower on CPU devices, even with architecture optimization like MBConv blocks. This is corroborated by our findings in Table A.2, which shows a greater latency increase moving from GPU to CPU in models which were more reliant on convolution blocks, like EfficientNet_b1 and EfficientNet_b7. Yet the value of architecture optimization cannot be ignored. Without a deeper understanding of how GPUs accelerate convolution and how different models optimize for GPU or CPU, it is hard to fully comprehend the performance tradeoff between parallel and sequential devices.
Fused-MBConv Blocks in EfficientNetV2: EfficientNetV2 was NAS trained on GPU to find the optimal combination of MBConv and Fused-MBConv blocks. In order to take better advantage of GPU acceleration, Fused-MBConv blocks remove the pointwise convolution, forcing depthwise convolutions to be sequentially applied for each individual channel [TL21]. This change echoes the structure of vanilla convolution blocks.

In Table A.2, EfficientNetV2 shows similar patterns to VGG19_bn, which used vanilla convolution. Specifically, convolution has a much lower PAR on GPU than on CPU for EfficientNetV2 (Table A.2), a marked difference from EfficientNet where PAR for convolution was similar across CPU and GPU. This is consistent with its use of Fused-MBConv blocks. Also, EfficientNetV2 was designed using NAS to find the optimal block combination on GPU, without guaranteeing that the combination is optimal on CPU.

Batch Normalization (BN) VS Layer Normalization: Data from Table A.2's latency columns shows that for most models, batch normalization slows down from GPU to CPU, but less than convolution. BN layers normalize over mini-batches of an activation matrix and require cell-by-cell operations, so the work of normalizing one column is $O(m)$ where m is batch size. Normalization occurs for all n batches in the activation matrix, giving batch normalization work of $O(mn)$. Batch normalization can be done in parallel on GPU, yielding a span of $O(m)$. LayerNorm layers normalize over the rows of the activation matrix. The work of normalizing one row is $O(c)$ where c is the number of channels in the activation matrix. Since normalization occurs for all m (h, w) dimensions in the activation matrix, layer normalization has work of $O(mc)$, and a parallelized complexity of $O(c)$ since multiple (h, w) dimensions can be computed at once. This correlates our findings in TableA.3, since $c = 1$ for over-the-air audio input. Thus cost of the Transformer's LayerNorm on GPU is almost $O(1)$, while loss of parallelization moving to CPU causes a large increase in latency.

Multi-Head Attention Parallelizability: The Multi-Head Attention layer computes the self-attention equation Softmax$(\frac{KQ^T}{\sqrt{d}}) \cdot V$ where $K, Q, V \in \mathbf{R}^{n \times d}$. Ultimately, Multi-Head Attention takes $O(n^2 d)$ work, all resulting from matrix multiplication. While matrix multiplication receives acceleration boosts both from PyTorch and GPU, Multi-Head Attention takes more work than convolution by a factor of n, limiting the speedup potential. Though the latency of Multi-Head Attention decreases moving from CPU to GPU, it does not show a major decrease akin to convolution (Table A.3). This is likely attributable in part to the limited optimization potential for attention layers.

Summing up, model efficiency-wise, when considering deployment on CPUs, MobileNet-related architectures (CNN) emerged as a more efficient choice due to their use of MBConv blocks, which reduced computational cost. However, for GPUs both vanilla CNNs and Transformers are good candidate architectures to take full advantage of GPU acceleration. It is also worth noting that Neural Architecture Search (NAS) optimization methods used for image data do not necessarily translate to benefits for audio data.

6.10 Conclusion:

In conclusion, our study revealed several key insights. Pre-training is not always a necessity, but it becomes important when dealing with models that have a large number of parameters.

We found that smaller features could lead to a sixfold increase in efficiency with only a 2% loss in mean average precision (mAP). Models that incorporate both local and global information demonstrated the best performance and were highly efficient. Attention-based models, though they have more parameters and are somewhat more challenging and sensitive during training, exhibited a robust resistance to noise despite their sharp loss landscape [Li+21]. Data augmentation was consistently beneficial, with proper data balancing and normalization identified as crucial factors for its success.

At this juncture, we are noticing a distinct data limitation that could potentially cap the performance of the supervised learning paradigm. This observation necessitates that we broaden our research scope in the following chapter to explore learning paradigms that aren't solely dependent on labeled data.

Chapter 7

Self-Supervised Training: Masked AutoEncoder

7.1 Introduction

As Transformers [Vas+17] and self-supervised learning [Dev+19; Bro+20b; Liu+19; He+20; CXH21; He+21b] are dominating computer vision (CV) and natural language processing (NLP) research. The revolution first started in NLP with the invention of the Transformer architecture and self-attention [PXS17]. Masked autoencoding with BERT [Dev+19] set a new state-of-the-art on various NLP tasks by self-supervised pre-training on large-scale language corpus. Similarly in the CV community, Vision Transformers (ViT) [Dos+20] have become popular for CV tasks, and, for self-supervised image representation learning, Masked Autoencoders (MAE) [He+21b] have brought the CV community closer to the success of BERT in NLP. In addition to the existing masked autoencoders that can read (BERT) or see (MAE), in this chapter we study those that can *listen*.

As mentioned in the previous chapter, Transformer-based models have recently refreshed leaderboards for audio understanding tasks as shown in Chapter 6. A key technique to train Transformers is to initialize audio model weights with ImageNet pre-trained supervised models (*e.g.*, DeiT [Tou+21]) by deflating patch embeddings and interpolating positional embeddings for encoding audio spectrograms. However, exploiting ImageNet pre-trained models could be sub-optimal. Unlike initializing video models with weights from image models (*e.g.*, the initial weights of I3D [CZ17a] or 3D-ResNets [FPW16] are inflated from ImageNet pre-trained image models), there are clear and notable discrepancies between spectrograms representing audio content and natural images. It remains unclear why such heterogeneous image-to-audio transfer is useful beyond arguably similar low-level semantics such as shapes of spectrograms and shapes of visual objects. Further, any label bias would inevitably be transferred to audio models.

Addressing these concerns, self-supervised audio representation learning has recently attracted much research attention. Based on BEiT [BDW21] that learns to reconstruct image patches or learnt patch tokens, SS-AST [Gon+21] extends to the audio domain and exploits spectrograms (akin to 1-channel 2D images) and uses both contrastive and reconstruction objective as self-supervision. Without using any labels, the key enabler to effective self-supervised representation learning is large-scale pre-training data. In this chapter, instead of using AudioSet [Gem+17b] for supervised training, we use it for pre-training without using the majority of its labels. Performing large-scale training with Transformer architectures is challenging as self-attention in Transformers has quadratic complexity w.r.t. the length of input sequence.

This computational burden has been addressed in different ways. A popular approach is to reduce the sequence length in self-attention. Various ViT-based architectures have been developed to alleviate such issues for image and video understanding. For example, Swin-Transformer [Liu+21] only performs local attention within windows that shift across layers. MViT [Wei+21] employs pooling attention to construct a hierarchy of Transformers where sequence lengths are downsampled. For self-supervised learning, MAE [He+21b] efficiently encodes only a small portion (25%) of visual patches while the majority of patches are discarded. The simplicity and scalability in MAE make it a promising framework for large-scale self-supervised learning.

In this chapter, we study MAE for AED and the unique challenges of the audio domain. We present Audio-MAE (Fig. 7.1) as a unified and scalable framework for learning self-supervised audio representations. Similar to MAE, it is composed of a pair of a Transformer encoder and decoder. Sound is first transformed and embedded into spectrogram patches. Before feeding them into the Transformer encoder, we mask and discard the majority and only feed a small number of non-masked embeddings into the encoder for efficient encoding. After padding encoded patches with learnable embeddings to represent masked patches, it then restores the order of these patches in frequency and time and propagates them through a Transformer decoder to reconstruct the audio spectrogram.

Unlike image patches, spectrogram patches are mostly locally correlated. For example, formants, the vocal tract resonances, are typically grouped and continuous locally in the spectrogram. The location in frequency and time embeds essential information that determines the semantics of a spectrogram patch and how it sounds like. To this end, we further investigate using localized attention and a hybrid architecture in the Transformer decoder to properly decode for reconstruction. This simple-yet-effective upgrade leads to improved performance for Audio-MAE.

Similar to MAE for images, we minimize the patch-normalized mean square error. At the fine-tuning stage, we discard the decoder and fine-tune the encoder with patch masking. Empirically, Audio-MAE sets a new state-of-the-art performance on six audio and speech classification tasks. It is the first audio-only self-supervised model that achieves state-of-the-art mAP on AudioSet-2M, outperforming other recent models with external

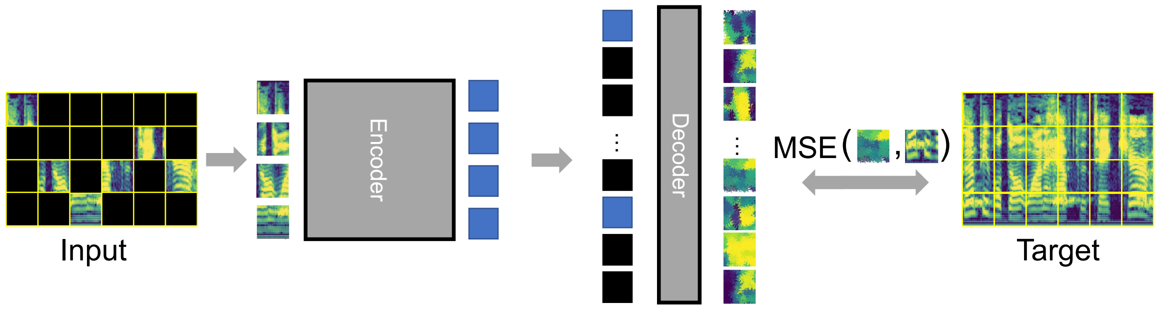


Figure 7.1: An audio recording is first transformed into a spectrogram and split into patches. We embed patches and mask out a large subset (80%). An encoder then operates on the visible (20%) patch embeddings. Finally, a decoder processes the order-restored embeddings and mask tokens to reconstruct the input. Audio-MAE is minimizing the mean square error (MSE) on the masked portion of the reconstruction and the input spectrogram.

supervision. We further provide the visualization and audible examples to qualitatively demonstrate the effectiveness of the Audio-MAE decoder.

7.2 Related Work

Masked pre-training in vision Masked/Denoising autoencoders [Vin+08; Vin+10; Dev+19] are a general representation learning methodology by reconstructing sources from masked or corrupted inputs. In CV, visually masked pre-training has made recent progress [Pat+16; Che+20b; He+21b; Wei+21]. Based on ViT [Dos+20] that applies Transformers to image patches, BEiT [BDW21] and MAE [He+21b] present masked image modeling frameworks. BEiT [BDW21] learns to predict discrete visual tokens generated by VAE [Ram+21] in masked patches. MAE [He+21b] reduces sequence length by masking a large portion of image patches randomly and encoding only non-masked ones for the reconstruction of pixel color information. MaskFeat [Wei+21] studies features for masked pre-training and find that Histograms of Oriented Gradients (HoG) [DT05], which are in turn related to spectrogram features, perform strongly for image and video classification models.

Out-of-domain pre-training for audio. Transferring ImageNet supervised pre-trained ViT [Dos+20] or ResNet [He+15b] has become a popular practice for audio models [GCG21a; Kou+21a; Nag+21a; Che+22; GCG21b; Gon+22a] as is covered in Chapter 6. After pre-training, these models operate over audio spectrograms by deflating from 3-channels (RGB) into 1-channel (spectrogram) in the pre-trained patch embedding and employing the rest of the backbone on top. HTS-AT employs Swin Transformer [Liu+21] to hierarchically encodes spectrograms. Without using out-of-domain (non-audio) data, Audio-MAE focuses on audio-only self-supervised pre-training from scratch.

In-domain pre-training for audio. Existing in-domain (audio-only) self-supervised methods can be broadly categorized by the input signal (*e.g.*, raw waveform [Sch+19; Bae+20; Bae+22], frame-level features [Hsu+21; SHM22; Sri+21]
or spectrogram patches [Gon+21; BPH22]) and the objective used for self-supervision (*e.g.*, contrastive [OLV18a; Bae+20; AZ18; Pat+21; Hsu+21] or prediction/reconstruction [Gon+21; Bae+22; Sri+21; SHM22]). For example, wav2vec 2.0 [Bae+20] takes raw waveform as inputs and exploits contrastive learning to discriminate contextualized representations in different time segments. Mockingjay [Liu+20] proposed a masked acoustic model pretext task to reconstruct frame-level Mel-features of masked time frames. SS-AST [Gon+21] is a self-supervised learning method operates over spectrogram patches and employs joint contrastive and reconstructive objectives on masked patches. Previous methods generate audio representations by encoding full-view of both masked and non-masked time or spectrogram segments for self-supervised pre-training. In contrast, Audio-MAE encodes only the non-masked patches. There exist independent and concurrent works [BPH22; Cho+22; Nii+22] using related methods. which we also compare against in this chapter.

7.3 Audio Masked Autoencoders (Audio-MAE)

Our Audio-MAE is a conceptually simple extension of MAE to audio. Fig. 7.1 shows an overview.
Spectrogram Patch Embeddings. Following [GCG21a; Gon+21], we transform audio recordings into Mel-spectrograms and divide them into regular grid patches. These patches are then flattened and embedded by a linear projection. As in MAE [He+21b], we add fixed sinusoidal positional embeddings to the embedded patches.

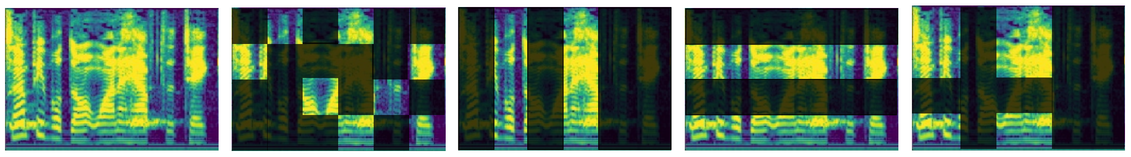

(a) Original (b) Unstructured (c) Time (d) Frequency (e) Time+freq

Figure 7.2: Audio-MAE's masking strategies on Mel-spectrograms.

Masking Strategies. Audio-MAE masks out a large subset of spectrogram patches. As a spectrogram can be viewed as a 2D representation of time/frequency components of a sound, it is reasonable to explore treating time and frequency differently during masking. In this chapter, we explore both the *unstructured* (*i.e.*, random masking without any prior) and *structured* (*i.e.*, randomly masking a portion of time, frequency, or time+frequency of a spectrogram) in the pre-training and fine-tuning phase. Illustrative examples are shown in Fig. 7.2. We show masked regions with a dark overlay.

The masking mechanism, as introduced in MAE [He+21b], is the key ingredient for efficient self-supervised learning. Masking reduces the sequence length and encourages learning global, contextualized representations from limited "visible" patches. We observe that akin to images, a large masking rate (80% in our experiments for spectrogram patches, which is similar to 75% in MAE for images) is feasible for learning self-supervised audio representations. Unlike BERT [Dev+19] that uses a 15% masking rate for self-supervised learning in NLP, most of the tokens/patches can be discarded for spectrograms as well as images due to high redundancy in these modalities. Beyond self-supervised pre-training, we further explore the effectiveness of masking in the supervised fine-tuning stage. Empirically, we found unstructured (random) masking at a higher ratio for pre-training and structured (time+frequency masking) at a lower ratio for fine-tuning provide the best accuracy (ablations are in §7.4.4).

Encoder. Audio-MAE uses a stack of standard Transformers [Vas+17] as its encoder. The encoder only processes (20%) non-masked patches to reduce computation overhead which is quadratic to the input sequence length. We use the 12-layer ViT-Base (ViT-B) [Dos+20] Transformer as our default. Given that masking is a crucial step in prompting the model to learn the distribution, it renders Convolutional Neural Network (CNN) architectures less suitable in Mean Absolute Error (MAE) scenarios. This is due to the inability of CNNs to manage zero-masking, resulting in errors.

Decoder with Local Attention. The decoder is also composed of standard Transformer blocks. The encoded patches from the encoder are padded with trainable masked tokens. After restoring the original time-frequency order in the audio spectrogram, we add the decoder's (fixed sinusoidal) positional embeddings and feed the restored sequence into the decoder. At the top of the decoder stack, we add a linear head to predict and reconstruct the input spectrogram.

To address the unique characteristics of audio spectrograms, our work investigates an enhancement to the vanilla MAE decoder. Image-based MAE uses *global self-attention* in the Transformer decoder which is appropriate for visual context, because visual objects are typically invariant under translation or scaling, and their exact position may not affect the semantics of an image. In contrast, the position, scale, and translation of spectrogram features however *directly affects* the sound or semantics of an audio recording. Consequently, global self-attention is sub-optimal for spectrograms if the time-frequency components is predominantly local. For instance, we would have better success to use the harmonics (*e.g.*, Fig. 7.2a) in lower bands of a vowel to predict the spectrogram patch vertically in a higher frequency band rather than horizontally in the time domain. Similarly, a frictional sound of a consonant likely only correlates to other part of the consonant, and is without dependency to other silence segments in the audio recording. Compared to images, the spectrogram patches are more similar to speech or text tokens where its order and position is more relevant.

To address the nature of audio spectrograms, in addition to using Transformers with global self-attention as in vanilla MAE, we incorporate the *local attention mechanism* which groups and separates the spectrogram patches in to local windows in self-attention for decoding. We investigate two types of local attention: (1) Shifted window location: Inspired by the shifted window in Swin Transformers [Liu+21], we shift window attention by 50% between consecutive Transformer decoder layers. For padding the margin when shifting, we cyclically shift the spectrogram to the top-left direction. Fig. 7.3 illustrates the localized decoder attention by shifted windows. (2) Hybrid window attention (global+local attention): Inspired by [Li+22c], to add better cross-window connections, we design a simple hybrid (global+local) attention that computes local attention within a window in all but the last few top layers. In this way, the input feature maps for the final reconstruction layer also contain global information. For simplicity, we use *no* pooling or hierarchical structure. Decoders with different attention types are compared in §7.4.4.

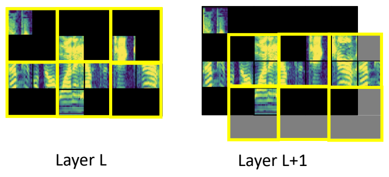


Figure 7.3: Decoder's local attention and shifted window (right).

Objective. The Audio-MAE decoder learns to reconstruct the input spectrogram by predicting the values in the spectrogram patches or their per-patch normalized ones. The objective is the mean squared error (MSE) between the prediction and the input spectrogram, averaged over unknown patches. Empirically we found employing the reconstruction loss alone is sufficient while including additional contrastive objectives (*e.g.*, InfoNCE loss [OLV18b]) does not improve Audio-MAE.

Fine-tuning for Downstream Tasks. During fine-tuning, we only keep and fine-tune the encoder with the decoder removed. Different from the original MAE, and inspired by [BAM19; Kou+21a], we also explore to employ masking in the fine-tuning stage to remove a portion of patches to further regularize learning from a limited view of spectrogram inputs, which, as a side effect, also reduces computation during fine-tuning. Compared to SpecAug [Par+19] which takes full-length input with the masked portion set to zero, Audio-MAE sees only a subset of real-valued input patches. Following MAE, we apply average pooling over encoded non-masked patches and employ a linear layer on top for classification.

7.4 Experiments

We perform an extensive evaluation on six tasks, including audio classification on AudioSet (AS-2M, AS-20K) and Environmental Sound Classification (ESC-50), and speech classification on Speech Commands (SPC-1 and SPC-2) and VoxCeleb (SID). We use AudioSet for ablation studies.

7.4.1 Datasets and Tasks

AudioSet [Gem+17b] (AS-2M, AS-20K) The setup is the same as Chapter 6. For the AS-2M experiments, we use the union of unbalanced and balanced training audio for pre-training and fine-tuning. For the AS-20K experiments, we use AS-2M for pre-training and the 20K balanced set for fine-tuning.

Environmental Sound Classification (ESC-50) [Pic15b] is an audio classification dataset consists of 2,000 5-second environmental sound recordings. There are 50 classes in ESC. We report accuracy under 5-fold cross-validation with the same split used by [GCG21a].

Speech Commands (SPC-2, SPC-1) [War18] are two keyword spotting tasks. In SPC-2, there are 35 speech commands. The training/validation/testing set has 84,843/9,981/11,005 1-second recordings, respectively. In SPC-1, there are 10 classes of keywords, 1 silence class, and 1 unknown class that includes all the other 20 common speech commands. We use the data and split provided in the SUPERB [Yan+21] benchmark to report the testing accuracy.

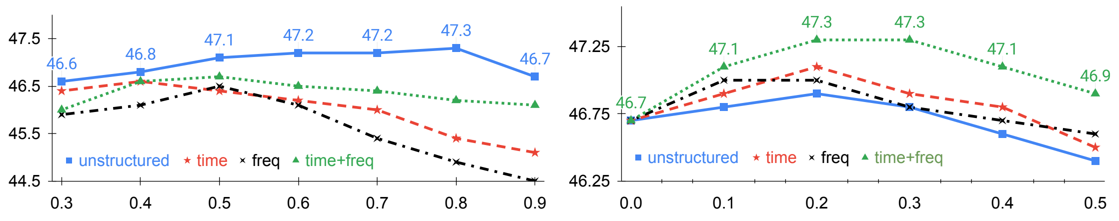
VoxCeleb (SID) [Nag+20] contains 150K utterances from 1,251 speakers. The speaker identification task (SID) is to classify the utterances to identify its original speaker. We use the V1 standard train (138,361), validation (6,904), testing (8,251) sets and report the testing accuracy.

7.4.2 Implementation Details

We use a vanilla 12-layer ViT-B by default as the Transformer encoder. For the decoder, we use a 16-layer Transformer with shifted local attention. We investigate the vanilla (global attention) and hybrid (global+local attention) decoder variants (see Table. 7.1c).

Following [GCG21a; Nag+21a], we transform raw waveform (pre-processed as mono channel under 16,000 sampling rate) into 128 Kaldi [Pov+11a]-compatible Mel-frequency bands with a 25ms Hanning window that shifts every 10 ms. For a 10-second recording in AudioSet, the resulting spectrogram is of $1\times1024\times128$ dimension.

For patch embedding, we use convolutional kernels with $(16, 16)$ size and stride in time and frequency (thus, patches are non-overlapping) to avoid short-cuts via overlap in self-supervision (though, at high masking ratios such short-cuts are less severe). By default, we use a masking ratio of 0.8 with (unstructured) random masking for pre-training. During fine-tuning, we employ a lower masking ratio (0.3 in time and 0.3 in frequency). Ablations on these design choices are given in §7.4.4.



(a) Pre-training masking (b) Fine-tuning masking

Figure 7.4: For pre-training, a *higher* ratio and *unstructured* masking (random) is preferred. For fine-tuning, a *lower* ratio and *structured* masking (time+frequency) is better. The y-axes are mAP on AS-2M and the x-axes are masking ratio. This ablation format follows [He+21b].

7.4.3 Pre-training and Fine-tuning

We use AudioSet-2M for pre-training and randomly iterate over all audio recordings. We train for 32 epochs with a batch size of 512 and a 0.0002 learning rate. We distribute the training load over 64 V100 GPUs and the total training time is ~36 hours. For each audio, we randomly sample the starting time, cyclically extract 10-second audio, and randomly jitter its magnitude by up to ± 6dB. We use only natural audio spectrograms and apply *no* augmentations (*e.g.*, [Par+19; Yun+19; Zha+18]) as we do not find these strong augmentations helpful in the pre-training phase.

In the fine-tuning phase, we remove the decoder and only fine-tune the encoder. For the supervised fine-tuning on AudioSet-2M, since the size of training samples are uneven across classes (unbalanced), we follow the common practice of using a weighted sampling to balance the classes during training. In each epoch, we sample 200K instances (~10% of AudioSet-2M) without replacement. We fine-tune for 100 epochs, which aggregate to ~10 full epochs of AudioSet-2M. The probability of sampling an instance is inversely proportional to the dataset-wise occurrences of its classes. Fine-tuning on 64 GPUs takes ~12 hours. For the smaller balanced AudioSet-20K, we fine-tune on 4 GPUs for 60 epochs without weighted sampling.

7.4.4 Ablations and Model Properties

Masking Strategies in Pre-training and Fine-tuning. In Fig. 7.4, we compare different pre-training and fine-tuning masking strategies for Audio-MAE. First, in Fig. 7.4a we explore the *pre-training masking ratio*. We observe, similar as in MAE for images [He+21b], that a high pre-training masking ratio (80% in our case) is optimal for audio spectrograms. This is due to the fact that both audio spectrograms and images are continuous signals with significant redundancy. Further, we find the unstructured random masking works the best for self-supervised pre-training over more structured masking (*e.g.*, time+frequency).

Unlike MAE for images, there are clear performance differences among masking strategies when pre-training with audio spectrograms. Comparing Audio-MAE reconstructions between Fig. 7.7a to 7.7e and 7.7d to 7.7h, under the same masking ratio, we observe the unstructured random masking is comparably easier than structured masking (*i.e.*, time and/or frequency) as the model can guess the missing component by extrapolating nearby context (*e.g.*, formants in vowels and frictional sounds in consonants around). We also observe that for higher masking ratios, the structured masking alternatives drop in performance, presumably because the task becomes too difficult while random masking improves steadily up to 80%. This result show that designing a pretext task with *proper hardness* is important for effective self-supervised learning of audio representations. We therefore use random masking with ratio of 80% as our default for pre-training.

Fig. 7.4b studies the effect of masking during the *fine-tuning* phase. We see that in this case, it is more beneficial to use structured masking: time+frequency performs better than time- or frequency-based masking, and these perform better than unstructured masking. Overall, we see that the optimal masking ratios are *lower* than for pre-training and we use 0.3 as our default in the fine-tuning phase.

In general, we observe that for task-agnostic pre-training, unstructured masking with a higher ratio is preferred. While in task-specific fine-tuning, structured masking with lower ratios performs better.

Impact of Patch Size and Stride. We compare the performance of Audio-MAE trained with different patch sizes and strides in Table 7.1a. A non-zero overlap (*i.e.*, stride < patch size) between patches will increase the number of patches and quadratically increase computation in floating point operations (FLOPs), as reported in the table. Most prior works follow AST [GCG21a] to use overlapped patches (patch = 16 and stride = 10) to boost end task performance. As shown in Table 7.1a, we do not observe a performance improvement using overlapped patches for Audio-MAE (both 47.3 mAP), presumably because due to overlap, the patch embedding can leak information into the masked patches. The non-overlapped 16×16 patches achieve a good balance between computation and performance. By default, we use this setup in our experiments.

Encoder. We investigate the design choices of encoder and decoder architectures in Audio-MAE. Table 7.1b shows the trade-off between encoder model size and performance. As expected, larger models achieve better performance, at a cost of computation and memory. The accuracy gain of ViT-L over ViT-B/S is more significant on the smaller and balanced AS-20K. For ViT-S, the performance gap to ViT-B can be significantly closed (5.0 → 2.3 mAP) when fine-tuning with more in-domain data (AS-20K → AS-2M).

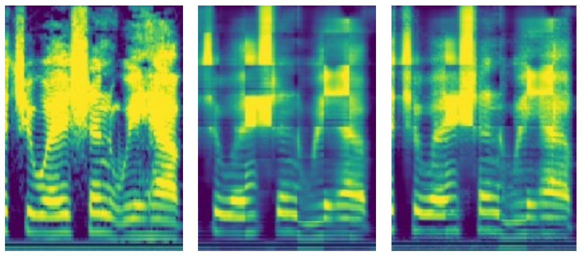


Figure 7.5: Decoder reconstruction comparison.

Decoder. Table 7.1c compares decoder attention types in Audio-MAE. Note that decoders are discarded after pre-training and only the equal-sized ViT-B encoders are fine-tuned for the end task. Our results show that *local attention* with shifted window achieves the best performance. Combining local and global attention (*i.e.*, hybrid attention, Hwin) also improves vanilla global self-attention. Fig. 7.5 shows the qualitative reconstruction comparison. In the spectrogram of vowels, the decoder with local attention reconstructs better harmonics and recovers more context in the spectrogram. Similar phenomena are observed in the frictional sound in the middle consonant.

Table 7.1d ablates the impact of decoder depth on mAP. A deeper 16-layer decoder achieves better performance against its shallower variants. Note that our decoder uses local window attention by default where only a fraction of tokens (4×4 local windows *vs.* 64×8 with global attention) are attended. For global attention we find 8-layer decoders to perform better than 16-layer. Table 7.1e compares decoder width (embedding dimension). A 512-dimension decoder achieves a good trade-off between computation and performance as a wider one is not better.

Pre-training Data and Setup. Table 7.1f summarizes the impact of pre-training dataset size when finetuned with the full AS-2M dataset. Overall the model performance is monotonically increasing when using more data for pre-training. Comparing the performance of using 1% well-annotated AS-20K balanced data to using randomly sampled 20K unbalanced data for pre-training, the similar mAPs (39.4 vs 39.6) suggest that the *distribution* of data classes (balanced *versus* unbalanced) is *less* important for pre-training. This is generally the same for the case when we finetune the model with the same fraction of the dataset, as is shown in Table 7.1g. We also compared with AST (IN-SL), the difference is more dramatic. Evidently, AudioMAE SSL is more robust to the skewed distribution than the supervised learning method. The performance gap can only be closed given enough data, as is shown in figure 7.6. Millions of additional data brought up the mAP of the majority of the classes. Meanwhile, as shown in Table 7.1h, training for longer is beneficial yet the performance saturates after the 24-*th* epoch.

Patch size, stride	Seq shape	FLOPs	mAP
(16,16), (16,16)	64×8	48.6	**47.3**
(16,16), (10,10)	101×12	130.5	**47.3**
(32,16), (16,16)	63×8	47.8	46.6
(16,32), (16,16)	64×7	42.1	46.8

(a) **Patch size and stride**

Backbone	#Params	AS-20K	AS-2M
ViT-S	22M	32.1	45.0
ViT-B	86M	37.1	47.3
ViT-L	304M	**37.6**	**47.4**

(b) **Model size (encoder)**

Attention type	AS-20K	AS-2M	ESC-50	SID
Global$^{(8)}$ (vanilla)	36.6	46.8	93.6	94.1
Local$^{(16)}$ (shifted)	**37.1**	**47.3**	**94.1**	94.8
Hwin (local$^{(8)}$+ global$^{(4)}$)	36.8	**47.3**	93.8	**95.0**

(c) **Decoder attention comparison**. Attn type$^{(depth)}$

Depth	mAP
2	46.8
8	47.2
16	**47.3**

(d) **Decoder depth**

Width	mAP
256	46.9
512	**47.3**
768	47.3

(e) **Decoder width**

% of AS-2M	mAP
1% (AS-20K)	39.4
1% (AS-2M)	39.6
10%	42.6
50%	46.4
100%	**47.3**

(f) **Pre-training size for AS-2M FineTune**

% of AS-2M	AudioMAE-mAP	AST[GCG21a]-mAP
1% (AS-20K)	37.1	34.7
1% (AS-2M)	32.3	23.6
10%	37.6	30.7
50%	44.5	41.1
100%	**47.3**	45.9

(g) **Pre-training size same as FineTune size**

epoch	mAP
8	46.5
16	46.8
24	47.2
32	**47.3**
40	47.3

(h) **Pre-training epoch**

scenario	IN-SSL	IN-SL	AS-SSL	AS-20K	AS-2M
(1)			✓	**37.1** (-0.0)	**47.3** (-0.0)
(2)	✓			32.1 (-5.0)	45.4 (-1.9)
(2)	✓	✓		32.5 (-4.6)	45.9 (-1.4)
(3)	✓		✓	36.9 (-0.2)	47.1 (-0.2)
(3)	✓	✓	✓	36.2 (-0.9)	46.9 (-0.4)

(i) **External ImageNet (IN) pre-training**. SSL: w/ self-supervised MAE. SL: w/ supervised (fine-tuned) MAE.

Table 7.1: **Ablation studies on AS-2M**
The gray entries are the default Audio-MAE setup (ViT-B encoder, decoder with shifted local attention, pre-trained for 32 epochs). Table format follows [He+21b].

Out-of-domain Pre-training on ImageNet. Initializing audio models from ImageNet pre-trained weights has become popular for audio classification. However, as there are significant discrepancies between image and audio modalities, it is questionable if out-of-domain pre-training benefits audio representation learning. In Table 7.1i we design 3 scenarios to investigate this for Audio-MAE: (1) Audio-only pre-training (AS-SSL) from scratch. We consider this the ideal schema for learning audio representations as it is a simple and clean setup that prevents uncontrollable bias transfer from other modalities. (2)

Directly using self-supervised ImageNet MAE models (IN-SSL) and its fine-tuned variant (IN-SL). (3) Audio-MAE self-supervised pre-training on top of these ImageNet weights.

The results show that (1) from-scratch *audio-only* pre-training is the best. For scenarios (2) and (3), we observe that ImageNet pre-training alone (2) is not sufficient (especially when the downstream data is smaller, AS-20K), and, in self-supervised pre-training on AudioSet, ImageNet initialization (3) does not help but degrades accuracy. Also in (3), supervised ImageNet pre-training (IN-SL) seems harmful. Consequently, the result suggests that out-of-domain pre-training (*i.e.*, ImageNet) is not helpful for Audio-MAE, possibly due to domain shift.

Model	Backbone	PT-Data	AS-20K	AS-2M	ESC-50	SPC-2	SPC-1	SID
No pre-training								
ERANN [VBV21]	CNN	-	-	45.0	89.2	-	-	-
PANN [Kon+19a]	CNN	-	27.8	43.1	83.3	61.8	-	-
In-domain self-supervised pre-training								
wav2vec 2.0 [Bae+20]	Transformer	LS	-	-	-	-	96.2^*	75.2^*
HuBERT [Hsu+21]	Transformer	LS	-	-	-	-	96.3^*	81.4^*
Conformer [Sri+21]	Conformer	AS	-	41.1	88.0	-	-	-
SS-AST [Gon+21]	ViT-B	AS+LS	31.0	-	88.8	98.0	96.0	64.3
Concurrent MAE-based works								
MaskSpec [Cho+22]	ViT-B	AS	32.3	47.1	89.6	97.7	-	-
MAE-AST [BPH22]	ViT-B	AS+LS	30.6	-	90.0	97.9	95.8	63.3
Audio-MAE (global)	ViT-B	AS	36.6±.11	46.8±.06	93.6±.11	**98.3**±.06	**97.6**±.06	94.1±.06
Audio-MAE (local)	ViT-B	AS	**37.0**±.11	**47.3**±.11	**94.1**±.10	**98.3**±.06	96.9±.00	**94.8**±.11
Out-of-domain supervised pre-training								
PSLA [GCG21b]	EffNet	IN	31.9	44.4	-	96.3	-	-
AST [GCG21a]	DeiT-B	IN	34.7	45.9	88.7	98.1	95.5	41.1
MBT [Nag+21a]	ViT-B	IN-21K	31.3	44.3	-	-	-	-
HTS-AT [Che+22]	Swin-B	IN	-	47.1	$97.0^\dagger$	98.0	-	-
PaSST [Kou+21a]	DeiT-B	IN	-	47.1	$96.8^\dagger$	-	-	-

Table 7.2: on audio and speech classification tasks. Metrics are mAP for AS and accuracy (%) for ESC/SPC/SID. For pre-training (PT) dataset, AS:AudioSet, LS:LibriSpeech, and IN:ImageNet. †: Fine-tuning results with additional supervised training on AS-2M. We gray-out models pre-trained with external non-audio datasets (*e.g.*, ImageNet). Best single models in AS-2M are compared (no ensembles). *: linear evaluation results from [Yan+21].

7.4.5 Comparison with the State-of-the-art

Table 7.2 compares Audio-MAE (with 3-run error bars) to prior state-of-the-art. We categorize the comparison into 3 groups. For a fair comparison, our main benchmark is the models in the middle group with self-supervised pre-training on in-domain (audio) datasets (AudioSet and LibriSpeech). For reference we also list other models without pre-training (the top group) and other models with supervised pre-training on out-of-domain ImageNet (the bottom group), where the latter contains the previous best systems on the datasets.

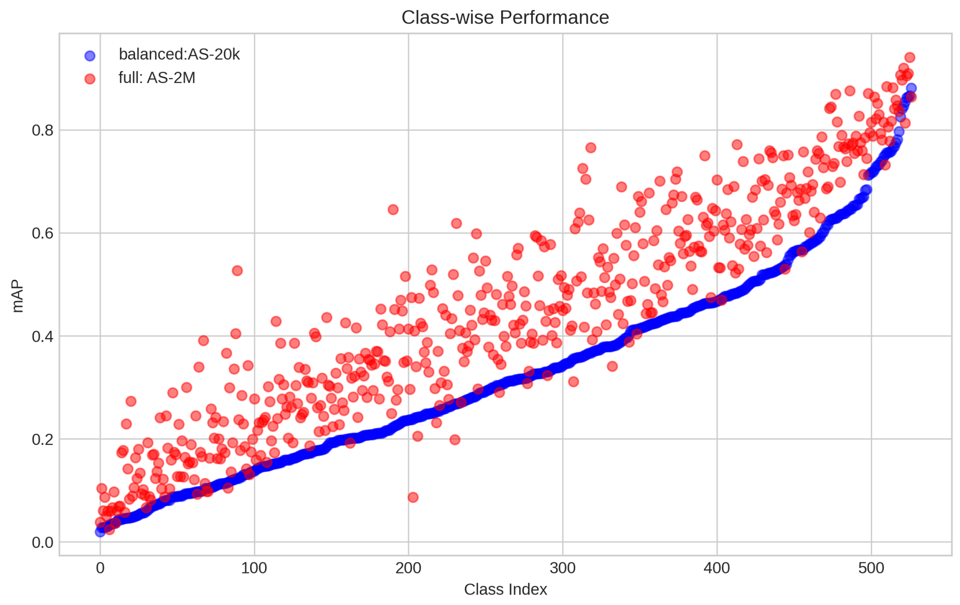
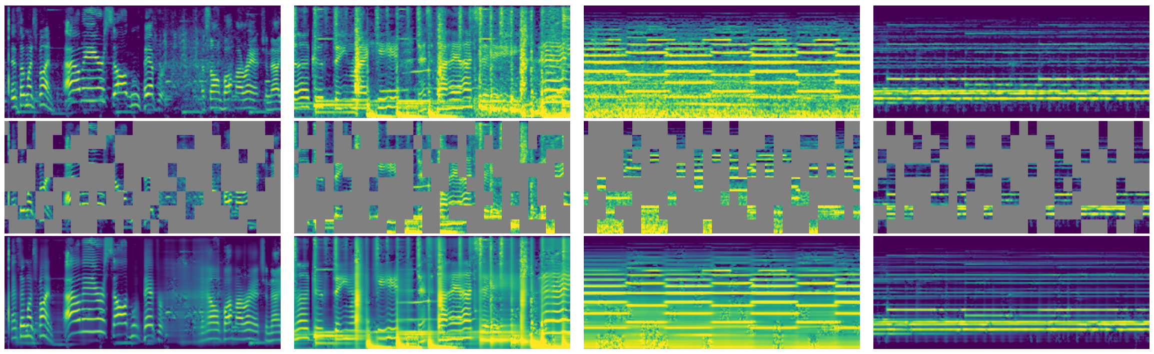


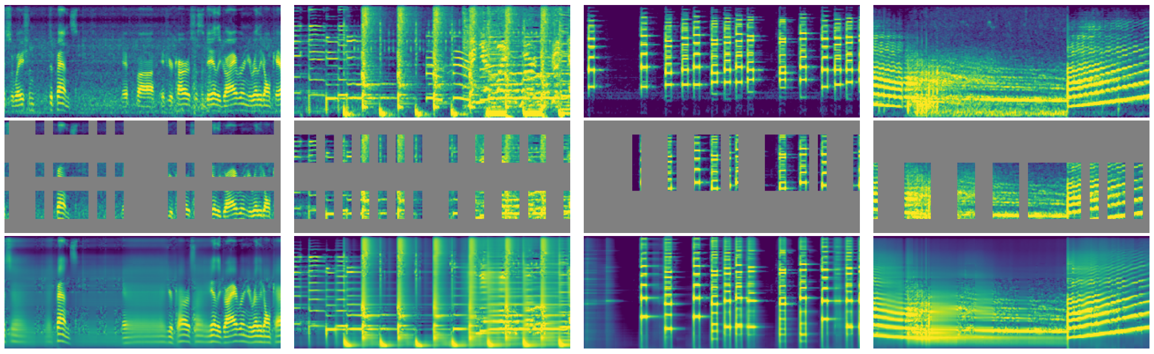
Figure 7.6: Perfomance of AS20k vs AS2M

Pre-trained on AudioSet, Audio-MAE achieves the best performance across all tasks compared to other models with in-domain self-supervised pre-training. On AudioSet-20K, its 37.1 mAP significantly outperforms all other approaches including concurrent works and other models with out-of-domain pre-training. On AudioSet-2M and ESC-50, our method also outperforms Conformer [Sri+21] and SS-AST [Gon+21]. Notably, unlike SS-AST and concurrent MAE-AST [BPH22], which trained with additional 1,000 hours of speech in Librispeech, we use only AudioSet for pre-training. Figure 7.6 showcased that most classes benefited linearly from more data. This trend also applies to all the models discussed in Chapter 6, the low-performing classes mostly remain low-performing, there are 81 of the bottom 100 performance classes remained in the bottom 100 classes when going from AS-20k trained to AS-2M trained. For instance, female/ male speech classes

(0.03 mAP, 0.05 mAP) are much worse than speech class (0.81mAP), this could be due to the highly imbalanced distribution of the training data. This issue could be alleviated through oversampling as will be shown in Chapter 13 § 13.7, but it is difficult to eliminate due to various other issues such as labeling errors and lack of distinguishability.



(a) Unstructured 1 2 3 (b) Unstructured 1 2 3 (c) Unstructured 1 2 3 (d) Unstructured 1 2 3



(e) Structured 1 2 3 (f) Structured 1 2 3 (g) Structured 1 2 3 (h) Structured 1 2 3

Figure 7.7: Spectrogram reconstruction visualizations on the AudioSet *eval* set. Column-wise type: speech, music, event, others. Masking type: (a-d) unstructured (random); (e-h) structured (time+frequency). Masking Ratio: 70%. In each group, we show the original spectrogram (1, top), masked input (2, middle), and MAE output (3, bottom). The spectrogram size is 1024×128; patch size is 16×16. Each sample has 64×8=512 patches with 154 (70% masked) patches being visible to Audio-MAE. Please click (1 2 3) for audible *.wav*s.

In the bottom group of Table 7.2, Audio-MAE also outperforms previous state-of-the-art models with ImageNet supervised pre-training. Note that the proposed Audio-MAE does not rely on any out-of-domain data and labels, nor use knowledge distillation (*e.g.*, DeiT) from additional CNN-based models. Also, compared to HTS-AT [Che+22] and PaSST [Kou+21a], Audio-MAE is trained with audio under 16K sampling rate. As experimented in [Kon+19a], there could be up to 0.4 potential mAP improvement for Audio-MAE if audio with 32K sampling rate is available.

For the speech tasks (SPC-1, SPC-2, and SID), Audio-MAE outperforms other models without pre-training (ERANN [VBV21], PANNs [Kon+19a]), supervised (AST) and self-supervised models (SS-AST, MAE-AST). We further list other works (marked with *) to include the latest results introduced in the SUPERB [Yan+21] benchmark. But note that these results are not strictly comparable since SUPERB employs linear evaluation where the underlying pre-trained models are not end-to-end fine-tuned.

In summary, with audio-only from-scratch pre-training on AudioSet, our Audio-MAE performs well for both the audio and speech classification tasks.

7.4.6 Visualization and Audible Examples by Audio-MAE Decoder

For better visualization, we follow MAE [He+21b] to use MSE over non-normalized spectrograms as the self-supervised objective. We use ViT-L as the Audio-MAE encoder for visualization. Fig. 7.7 illustrates the reconstruction results sampled from the AudioSet-2M *eval* set. We further reconstruct *.wav*s using the Griffin-Lim [GL84] algorithm, audible under the anonymous links (accessible in respective 1 2 3).

As can be seen and heard, for various masking strategies and different sounds, our Audio-MAE generates reasonable reconstruction. It works well for noisy event sounds (*e.g.*, the reconstructed siren in Fig. 7.7c-3), as well as speech and music (*e.g.*, the reconstructed singing in Fig. 7.7b-3). Notably, unlike visual contents that are typically scale/translation/position invariant [Liu+21], absolute positions and arrangement of spectrogram components are critical for humans to understand sound [ST04]. For example, shifting a pitch will make an audio sounds completely different. Also, phoneme sequences in time are important cues for speech understanding. Consequently, unstructured masking produces better aligned outputs that are closer to the ground truth (top row in each subfigure) as the model can make better predictions based on nearby spectrogram patches; while structured masking is harder (less accurate or with words missing), especially when masking is performed over the time axis. A failure example (missing words) is the reconstructed speech in Fig. 7.7e-3.

7.5 Conclusion

Now, we have explored a simple extension of MAE [He+21b] to audio data. Our Audio-MAE learns to reconstruct masked spectrogram patches from audio recordings and achieves state-of-the-art performance on six audio and speech classification tasks. We have drawn four interesting observations: First, a simple MAE approach works surprisingly well for audio spectrograms. Second, we find that it is possible to learn stronger representations with local self-attention in the decoder, reconstruction seems to be the most promising pretext task for AED. Third, we show that masking can be applied to both pre-training and fine-tuning, improving accuracy and reducing training computation. The optimal strategy depends on the nature of the data (audio, image, *etc.*) and the learning type (self-/supervised). Fourth, the best performance can be achieved by pre-training and fine-tuning under the same modality, without reliance on cross-modality transfer learning.

In the concluding chapter of this section, it is evident that Transformers are surpassing Convolutional Networks. Due to their inability to handle zero-masking without producing errors, Convolutional Networks can't be masked and hence are unsuitable for Mean Absolute Error (MAE) scenarios. This limitation restricts the applicability of ConvNets for MAEs. Conv nets are only left with augmentation-based contrastive-learning or denoising autoEncoder-style SSL, which are recently shown to be suboptimal [Ass+23].

Part II

Robustness of Audio-visual Systems

Part II Overview:

In this part, we start with verifying our motivation – verifying the existence of threats from adversarial attacks in the real world. We demonstrated threats from adversarial perturbation could exist in both unimodal visual and audio domains even without tampering with the target of interest. (Chapter 8)

In order to learn a robust multimodal model, we first study whether we could build a model to take advantage of multimodal input, by building a better joint embedding. We show that a better loss objective and more sophisticated fusion methods tend to strengthen the robustness of a system by better leveraging information from auxiliary modalities. We based our study on the Video-text retrieval task. (Chapter 9)

To gain a deeper insight into the resilience of multimodal systems, we use adversarial perturbations as scalpels to stress test their performance since adversarial attacks are formulated to be the "worst case" scenarios for any model. We show some intriguing properties of multimodal AED models under the stress test of adversarial perturbation. (Chapter 10) In this way, not only can we quantify robustness, but we could also be better informed about how to design robust multimodal systems in practice, where we showed that strategies such as convexity-aware and density-aware mixup are effective in improving the adversarial robustness of a multimodal system. (Chapter 11)

Due to the prohibiting cost of adversarial training, we foresee the continued prevalence of standard-trained models in the near future. We believe it is important to understand standard-trained models' robustness without any defense. We performed an empirical study of audio models against common corruption and adversarial perturbation(Chapter 12). We also study the correlation between robustness and sharpness in this chapter. This study showcased some limitations of the canonical definition of robustness, and the gap between adversarial robustness and real-world noise robustness.

Chapter 8

Verifying Threats

8.1 Introduction

Recent demonstrations of the ease with which machine learning systems can be "fooled" have caused a strong impact in the field and in the general media. In the majority of the cases studied, however, these attacks are considered in a "purely digital" domain, that is they consider perturbations that directly modify the digital input to the classifier in order to fool the model. Do we need to really worry about them if they only exist digitally?

A smaller but still substantial line of work has emerged to show that these attacks can also transfer to the physical world: these have considered cases of printing out pictures intended to fool classifiers [Bro+17], placing physical stickers on objects to fool a classifier [Evt+18], or directly manufacturing objects intended to fool a classifier (even when observed from multiple angles) [Ath+17]. In all these cases, though, the primary mode of attack has been manipulating the *object* of interest, often with very visually apparent perturbations. Compared to attacks in the digital space, physical attacks have not been explored to their full extent: we still lack baselines and feasible threat models that work robustly in reality. In this chapter, we are going to show 2 adversarial examples that work in reality in both vision and audio that do not require manipulating the object of interest.

8.2 Adversarial Camera Sticker

We show a method for physically fooling a network such that all *objects* of a particular class are misclassified using a visually *inconspicuous* modification. To accomplish this, we develop an "adversarial camera sticker", which can be affixed to the lens of the camera itself, and which contains a carefully constructed pattern of dots that causes images viewed by the camera to be misclassified. These dots look like mostly-imperceptible blurry spots

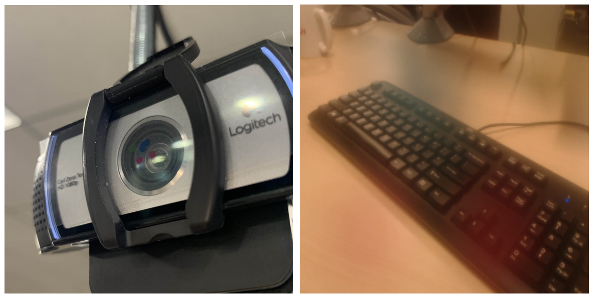


Figure 8.1: Illustration of our approach: (left) our adversarial sticker affixed to a camera lens; (right) view from the camera with sticker, where the keyboard is reliably classified as a computer mouse for most viewpoints and scales.

in the camera image, and are not obvious to someone viewing the image, as is exemplified in Figure 8.1. The main challenge of creating this attack relative to past work is that the space of possible perturbations we can feasibly introduce with this model is very limited: unlike past attacks which operate in the pixel-level granularity of the images, the optics of the camera mean that we can only create blurry dots without any of the high-frequency patterns typical of most adversarial attacks. The overall procedure consists of three main contributions and improvements over past work:

1. Our threat model is the first of its kind to inject a perturbation on the optical path between the camera and the object, without tampering with the object itself.

2. Ours is the first instance we are aware of a "universal" physical perturbation (universal perturbations in digital space were considered by Moosavi-Dezfooli et al. [MD+17]). Owing to the fact that the sticker will apply the same perturbation to all images, we need to create a single perturbation that will cause the classifier to misclassify over multiple angles and scales.

3. Our method jointly optimizes the adversarial nature of the attack while fitting the threat model to the perturbations achievable by the camera. This is due to the challenge of finding the "allowable" set of perturbations that can be physically observed by the camera.

We carefully crafted our dot attacks and the visual threat model (i.e., the set of allowable physical perturbations) using a differentiable alpha blending module (will be explained in Section 8.2.1). We train the attack, and conducted evaluation experiments, both on a real camera using printed stickers, and (virtually, but with physically realistic perturbations) on the ImageNet dataset [Den+09]. Our experiments show that on real video data with a physically manufactured sticker, we can achieve an average targeted fooling rate of 52%

over 5 different class/target class combinations, and furthermore reduce the accuracy of the classifier to 27%. On the ImageNet test set, we can construct a (digital, but physically realistic) attack that reduces the classification accuracy of the "street sign" ImageNet class from 64% to 18%, and achieves an average of a 33% targeted fooling rate over 50 randomly selected target classes. In total, we believe this work substantially adds to the recent crucial considerations of the "right" notion of adversarial threat models (see e.g. Papernot et al. [Pap+18] for additional discussion on a similar point), demonstrating that these pose a physical risk when an attacker can access a camera.

8.2.1 A threat model for physical camera sticker attacks

Traditional attacks on neural network models work as follows. Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, we want to find some perturbation function $\pi : \mathcal{X} \to \mathcal{X}$, such that for any input $x \in \mathcal{X}$, $\pi(x)$ looks "indistinguishable" from x, yet is classified incorrectly by f even when x is classified correctly, i.e., $f(x) \neq f(\pi(x))$. Typically π is taken to be some norm-bounded additive perturbation, i.e.,

$$\pi(x) = x + \delta \tag{8.1}$$

with $\|\delta\|_p \leq \epsilon$ for some bound ϵ (we generally refer to the space of all such allowable function as Π), though in this chapter we will specifically consider a much more limited class of perturbations, to ensure that they are actually achievable in a real system.

The goal of standard adversarial attacks is, for a given $x \in \mathcal{X}, y \in \mathcal{Y}$, to find a perturbation $\pi \in \Pi$ that maximizes the loss

$$\max_{\pi \in \Pi} \mathcal{L}(f(\pi(x)), y). \tag{8.2}$$

A slight variant of this approach is to consider *targeted* attacks that specifically try to maximize the loss of the true class and minimize the loss of some target class

$$\max_{\pi \in \Pi} \left(\mathcal{L}(f(\pi(x)), y) - \mathcal{L}(f(\pi(x), y_{\text{targ}})) \right). \tag{8.3}$$

Finally, whereas the standard attacks are able to adapt π to a specific input, another variant (which we will need to consider here), is a *universal* perturbation, where a single perturbation function π must be chosen to maximize expected error over multiple samples drawn from some distribution $\mathcal{D}$

$$\max_{\pi \in \Pi} \mathbf{E}_{(x,y) \sim \mathcal{D}} \left[\mathcal{L}(f(\pi(x)), y) \right] \tag{8.4}$$

and where we can also consider the targeted variant as well.

To design a threat model for a physical camera attack, we need to consider the approximate effect of placing small dots on a sticker. Owing to the optics of the camera lens, a small opaque dot placed upon the camera lens will create a small *translucent* patch on the image itself. Assuming sufficient lighting, such translucent overlays can be well-approximated by an alpha-blending operation between the original image and an appropriately sized and colored dot.

More formally, explicitly considering x to be a 2D image with $x(i, j)$ denoting the pixel at the (i, j) location, we consider the perturbation function for a *single* dot in the image $\pi_0(x; \theta)$, (where θ denotes the parameters of the perturbation model, which we will discuss shortly), given by

$$\begin{aligned} \pi_0(x;\theta)(i,j) &= (1-\alpha(i,j)) \cdot x(i,j) + \alpha(i,j) \cdot \gamma \\ \alpha(i,j) &= \alpha_{\max} \cdot \exp(-d(i,j)^{\beta}) \\ d(i,j) &= \frac{(i-i^{(c)})^2 + (j-j^{(c)})^2}{r^2} \end{aligned} \tag{8.5}$$

where the parameters:

$$\theta = \left(\gamma, (i^{(c)}, j^{(c)}), r, \alpha_{\max}, \beta\right) \tag{8.6}$$

correspond to the following aspects of the dot:

- $\gamma \in [0,1]^3$ – RGB color
- $(i^{(c)}, j^{(c)}) \in \mathbb{R}^2$ – center location (pixel coordinates)
- $r \in \mathbb{R}_+$ – effective radius
- $\alpha_{\max} \in [0,1]$ – maximum alpha blending value
- $\beta \in \mathbb{R}_+$ – exponential dropoff of alpha value

Intuitively, this perturbation model captures the following process. Each dot is parameterized by its center location in the image $(i^{(c)}, j^{(c)})$ and its color γ. Each pixel $\pi_0(x;\theta)(i,j)$ in the perturbed image is given by a linear combination between the original pixel and the color γ, weighted by the position-dependent alpha mask $\alpha(i,j)$. For pixels (i,j) that are closer than the radius r to the point, the corresponding $\alpha(i,j)$ value will be close to $\alpha_{\max}$, whereas for pixels that are far away, the $\alpha(i,j)$ value will be essentially zero. Finally, the β parameters control the "smoothness" of the dropoff: for $\beta \longrightarrow \infty$, the alpha mask will hard dropoff at a radius r, whereas for smaller values the dropoff in alpha values will be more gradual.

To form our final perturbation model, we simply compose K of these single-dot perturbations, that is

$$\pi(x;\theta) = \pi_0(\cdot;\theta_K) \circ \ldots \circ \pi_0(\cdot;\theta_2) \circ \pi_0(x;\theta_1) \tag{8.7}$$

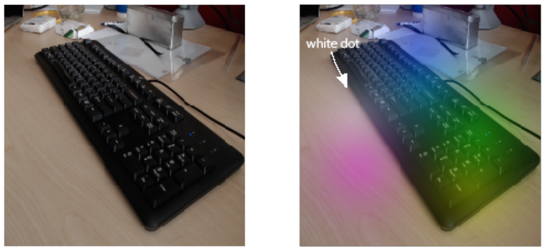


Figure 8.2: There are 8 dots in this case, note that the white dot cannot be printed in the real world.

where the total parameters $\theta = (\theta_1, \ldots, \theta_K)$ are simply the concatenation of the parameters for each dot. A visualization of a possible perturbation under this model, with multiple dots with different center locations and colors, $\alpha_{\max} = 0.3$, $\beta = 1.0$, and a radius of 40 pixels is shown in Figure 8.2. One point that is important to emphasize is that resulting image $\pi(x; \theta)$ is a *differentiable* function of the parameters θ: all parameters are real-valued quantities and each pixel value is a continuous function of the parameters. In other words, we can implement the perturbation model within an automatic differentiation toolkit (we implement it in the PyTorch library), a feature that will be exploited to both fit the perturbation model to real data and to construct adversarial attacks, in the following sections.

8.2.2 Achieving inconspicuous, physically realizable perturbations

Although the perturbation model above captures a reasonable approximation to attacks that can be created by a physical sticker, there are obvious problems with simply optimizing an attack over its parameters θ. First, it is easy to create attacks that fool a classifier, but which are very obvious to an observer (consider a fully opaque "dot" that covers the entire image). But second, and subtly, most parameterizations in such a class will not correspond to a perturbation that can be physically achieved on the camera. Indeed, this can be seen in Figure 8.2: the neon-colored dots are possible to reproduce with a printed sticker, and the precise radius, opacity, and smoothing dropoff of an actual physical dot are not perfectly controllable but are limited to a very narrow range of achievable parameters.

Owing to this fact, instead of constructing attacks under the general perturbation model first and then attempting to manufacture them physically, we take an opposite approach: we manufacture physical perturbations that we *know* to be both inconspicuous and (by definition) physically realizable, and we fit the parameters of the perturbation model to recreate these physical perturbations. Ultimately, this will give us a class of perturbation function Π corresponding to *most* of the parameters in our perturbation model being fixed (radius, opacity, dropoff, and a discrete set of allowable colors), while only a few remain free (namely the specific choice of color and the location of the dot).

In more detail, our process works like the following. We print a single small physical dot on transparent paper, and collect two images of the same visual scene (with the camera rigidly mounted so as not to move between taking the two images), one with a “clean" view, referred to as $x^{(0)}$ and one with the dot placed in front of the camera, referred to as $x^{(1)}$; one instance of such images are shown in Figure 8.8. We then used the structural similarity (SSIM) [Zha+17d], which measures the similarity of 2 pictures, to fit the parameters of our perturbation model to reconstruct the observed perturbation as much as possible; that is, we solve the optimization problem

$$\max_{\theta} \ \mathrm{SSIM}(\pi_0(x^{(0)};\theta), x^{(1)}) \tag{8.8}$$

where we note that by convention, we want to *maximize* the SSIM to make the images as close as possible.

In theory, because both the SSIM and the perturbation model π are differentiable functions, we could simply use projected gradient descent (PGD) to optimize our perturbation model. In reality, however, the loss landscape of this problem is highly non-convex, and most random initializations will have uninformative gradients. Consider the case where the initial guess of the dot location does not overlap at all with the actual dot location: in this case, there will be no informative gradient over the dot location (and hence no informative gradients over the other dot parameters either). In practice, the search procedure, therefore, requires that we find a very good initialization for the dot location, its color, and the other parameters, and only run PGD from this initial location. The precise details of our initialization procedure are perhaps somewhat unimportant to the main messages of this paper, but since the practical implementation is a key aspect to this work overall, very briefly, we use the following strategies to find initial points and optimize the SSIM:

- To initialize the position we simply make an informed guess based upon the known position of the dot in each such physically perturbed images.

- To initialize color space, we fit a linear transformation between the RGB values on the digital version to be printed, and the actual RGB values observed by the camera, that is, we printed 50 separate colors to estimate the transform, $k_{\mathrm{R/G/B}}$ and $b_{\mathrm{R/G/B}}$ are scalars.
$$\gamma^{(\mathrm{observed})}_{\mathrm{R/G/B}} = k_{\mathrm{R/G/B}} \cdot \gamma^{(\mathrm{printed})}_{\mathrm{R/G/B}} + b_{\mathrm{R/G/B}} \tag{8.9}$$

- Initial values for the remaining radius, $\alpha_{\max}$, and β values are simply fixed to values that produce reasonable-looking images.

- In maximizing the SSIM, we found it was advantageous to employ a kind of "block coordinate descent" to iteratively optimize over each set of parameters (center location, color, $\alpha_{\max}$, β, and radius) individually, each using gradient descent.

We performed this procedure for 50 different physical dots, to learn a single set of $\alpha_{\max}$, β, and r parameters, and 50 different colors (and of course, 50 different locations, though these parameters were only fit here in order to fit the remaining parameters well). After fitting these parameters, we restricted the allowable class of perturbations Π to contain up to 10 composed dots, each with fixed $\alpha_{\max}$, β, and r, and with the color gamma coming from a set of 10 possible color choices $\gamma \in \Gamma \equiv \{\gamma^{(1)}, \ldots, \gamma^{(10)}\}$, where we chose the 10 colors according to how often a single dot of this color (placed randomly) would cause a ResNet to misclassify an example. In other words, the ultimate free parameters of our perturbation model are 1) the center location $(i^{(c)}, j^{(c)})$ of the dot, and 2) the discrete choice of color $\gamma \in \Gamma$; these are the parameters we subsequently optimize over to create adversarial inputs.

8.2.3 Constructing adversarial examples

Given this perturbation model, our final goal is to find some $\pi \in \Pi$ (that is, choose center locations and colors) to maximize the fooling rate of the attack. We specifically consider building a targeted universal adversarial attack against a *single* class $y^\star$, i.e., we want to fool all observed instances of some class $y^\star$ to be labeled as a target class y_{targ} instead. This is formally specified by the optimization problem

$$\max_{\pi \in \Pi} \mathbf{E}_{x \sim D(x|y^\star)} \left[\mathcal{L}(f(\pi(x)), y^\star) - \mathcal{L}(f(\pi(x)), y_{\text{targ}})\right]. \tag{8.10}$$

In practice, we approximate this loss by maximizing the loss on some subset of images for a given class. That is, if $x^{(1)}, \ldots, x^{(M)}$ represent a set of images for class $y^\star$, we maximize the loss

$$\mathcal{L} = \sum_{l=1}^{M} \left(\mathcal{L}(f(\pi(x^{(l)})), y^\star) - \mathcal{L}(f(\pi(x^{(l)})), y_{\text{targ}})\right). \tag{8.11}$$

Again, because the perturbation model and loss are differentiable in the parameters $(i_k^{(c)}, j_k^{(c)})$ and γ_k (where the subscript k denotes the parameters for the kth dot), we could in theory optimize these with pure (projected) gradient descent. However, we here have only a discrete set of allowable color choices $\gamma \in \Gamma$, and as before, the gradients with respect to the dot positions present a highly non-convex loss surface; and unlike in the case of fitting the perturbation model parameters, there is no "natural" initialization for these parameters in this highly non-convex space. Owing to this, we resort to a greedy block coordinate descent search over individual block positions and colors. That is, we discretize the image into a 45 x 45 grid of possible locations for a dot center (45 is chosen such that each is 5 pixels apart for ImageNet), which we denote C, plus the 10 discrete possible colors. After running coordinate descent in this manner, we finally fine-tune the position of the dots (since the colors are discrete, we cannot fine-tune these), using gradient descent over the loss function as well. The full procedure is shown in Algorithm 1.

Algorithm 1: Coordinate Descent: maximize the attack

```
Input: Number of dots $K$, True class $y^\star$, target class $y_{\text{targ}}$, dataset $x^{(1)}, \dots, x^{(M)}$ of
images for class $y^\star$
Output: perturbation $\pi \in \Pi$ parametarized by number of dot centers and colors
$(i_k^{(c)}, j_k^{(c)})$ and color $\gamma_k \in \Gamma$ for $k = 1, \dots, K$.
Initialization: $(i_k^{(c)}, j_k^{(c)}) := \emptyset$ (i.e., no visible dot)

// Greedy coordinate descent
repeat
  for $k \in 1, \dots, K$ do
    // Test all locations and colors for this dot
    for $(i_k^{(c)}, j_k^{(c)}), \gamma_k \in C \times \Gamma$ do
      Evaluate loss (8.11)
    end for
    Choose $(i_k^{(c)}, j_k^{(c)}), \gamma_k$ to be the parameters that achieved the highest loss.
  end for
until Loss converges

// Gradient descent fine-tuning
repeat
  for $k \in 1, \dots, K$ do
    Update $i_k^{(c)}, j_k^{(c)}$ via gradient descent on loss (8.11)
  end for
until Loss converges
```

8.3 Vision Experiments and Results

Here we present experiments of our attack evaluating both the ability of the digital version of the attack to misclassify images from the ImageNet dataset (still restricting perturbations to be in our physically realizable subset), and evaluating the system on two real-world tasks: classifying a computer keyboard as a computer mouse and classifying a stop sign as

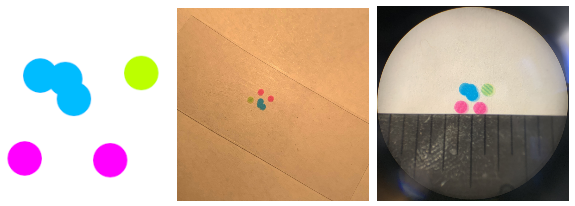


Figure 8.3: Illustration of $r = 0.025$ inch dots printed on the camera sticker from Photoshop, corresponding to $r = 40 pixel$ dots in camera space: (left) bitmap representation of sticker; (middle) printed sticker on transparency; (right) microscope view.

a guitar pick. We also detail some key results in the process of fitting the threat model to real data.

8.3.1 Experimental setup

All our experiments consider fooling a ResNet-50 [He+16b] classifier, pre-trained using the ImageNet dataset [Den+09]; we specifically use the pre-trained model included in the PyTorch library [Pas+17]. We use an HP Color LaserJet M253 to print the dot patterns on transparency papers; both the printer and the paper are off-the-shelf office supplies. We can print different-sized dots with radius no smaller than 0.01 inches in multiple colors. Figure 8.3 shows the printing pipeline.

8.3.2 Training and Classification on ImageNet

To train and evaluate the system on a broad range of images, we use the same ImageNet dataset used to train the classifier originally. For example, to train a “computer keyboard” to “computer mouse” attack, we optimize the parameters of our attack perturbation, using Algorithm 1, with the 1000 images in the ImageNet dataset corresponding to computer keyboards, and using the computer mouse class as the target class. Figure 8.4 shows the learned perturbations for two instances. Note that these are still *digital* representations of the perturbations, but they are constrained such that we already know how to manufacture the physical realization of these dots, as we will show in the subsequent section; however, all results in this section will be dealing with the digital versions.

Table 8.1 shows the ability of our learned perturbations (6 dots) to fool images from the ImageNet test set for these two categories. We also showed the average success rate to fool stop sign into 50 random classes. More generally, we also include the averaged results for fooling 50 random classes into other 50 random target classes. While our attacks cause smaller increase in error compared to what is common from traditional purely-digital attacks, we emphasize that the allowable class of perturbation is very much smaller in our

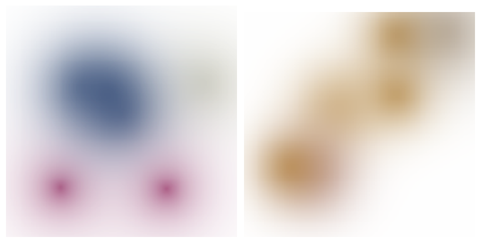


Figure 8.4: corresponding to the targeted attacks found for the (left) “computer keyboard” to “computer mouse” and (right) “street sign” to “guitar pick” adversarial attacks. In both cases the 6-dot attacks can be physically manufactured, these figures show a digital representation.

		Prediction		
Class	Attack	Correct	Target	Other
Keyboard →	No	85%	0%	15%
Mouse	Yes	48%	36%	16%
Street sign →	No	64%	0%	36%
Guitar Pick	Yes	32%	34%	34%
Street sign →	No	64%	0%	36%
50 classes	Yes	18%	33%	49%
50 Classes →	No	74%	0%	26%
50 classes	Yes	42%	31%	27%

Table 8.1: Performance of our 6-dot attacks on ImageNet test set.

setting. In particular, by constraining perturbations to be both visually imperceptible *and* realizable by placing a sticker on the lens, we are significantly constraining the allowable parameter space, and further constraining it to have relatively low-frequency components as compared to traditional attacks. Thus, the fact that we can decrease the accuracy so substantially with a relatively small overlay is quite notable. Of course, the real test of the method is the ability to transfer these attacks to the real world, as we discuss in the next section.

Unrealizable attacks Before moving to the physical attacks, we want to emphasize that the threat model we discuss, if *not* constrained to be physically realizable *or* particularly inconspicuous, can produce much higher fooling rates; By simply running PGD on the underlying parameters of the threat model within allowable range $\gamma \in [0,1]^3$; $(i^{(c)}, j^{(c)}) \in$

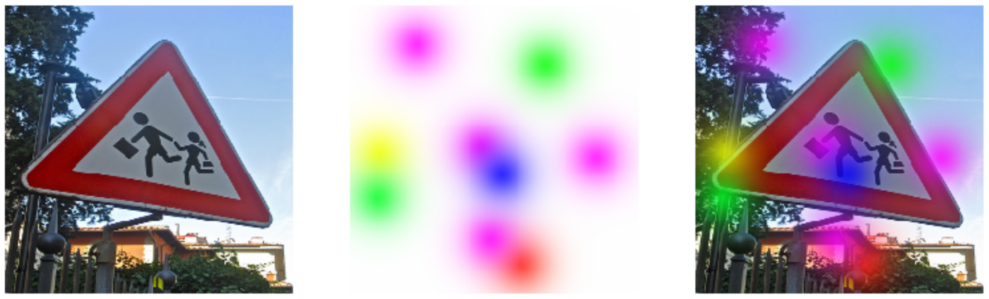

(a) “Street sign" (b) Perturbation (c) “Projector"

Figure 8.5: Physically unrealizable results generated by directly optimizing digital attacks. Here, “projector" is the model’s prediction of the perturbed image of a “street sign".

$[0, 224]^2$; $r \in [1, 60]$; $\alpha_{\max} \in [0, 1]$; $\beta \in \mathbb{R}_+$, we can indeed get a strong (universal) perturbation, but one which would still be considered “adversarial” by the context of much past work (in that a human could still identify the true underlying image). Figure 8.5 illustrates an additional instance of universal perturbations in digital space that fools “street sign" into “projector" which achieves an 83% fooling rate on the ImageNet test set. However, Figure 8.5b shows an attack being overly opaque. It also requires bright colors that are not possible to print and view via a transparent sticker. Thus, the result mainly serves to emphasize that explicitly fitting the threat model to physical data is crucial for achieving realistic perturbations.

8.3.3 Evaluation of attacks in the real world

In this section we present our main empirical result of the work, illustrating that the perturbations we produce can be adversarial in the real world when printed and applied physically to a camera, and when viewing a target object at multiple angles and scales. A video demonstrating the attack is available at `https://youtu.be/wUVmL33Fx54`. This is the first time that such an adversarial attack has been demonstrated in the real world (that is, an attack that modifies the camera viewing the scene instead of the object), and it opens up a new attack vector for adversarial examples against deep learning systems.

Specifically, using the process above, we printed the physical stickers corresponding to the two attacks mentioned in the previous section and affixed them to our camera. We then recorded short videos of the camera viewing these two physical objects at a number of different angles and scales. In particular, for each case, we created a 1000 frame video of the camera (with the sticker attached) viewing the target object from multiple angles, and used our ResNet-50 classifier to predict the class of each object.

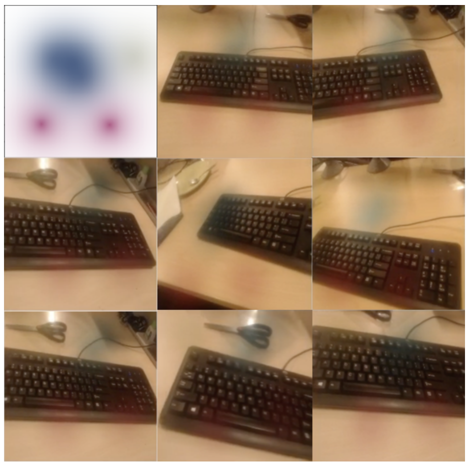


Figure 8.6: Sticker perturbation to fool “computer keyboard" class to “mouse" class

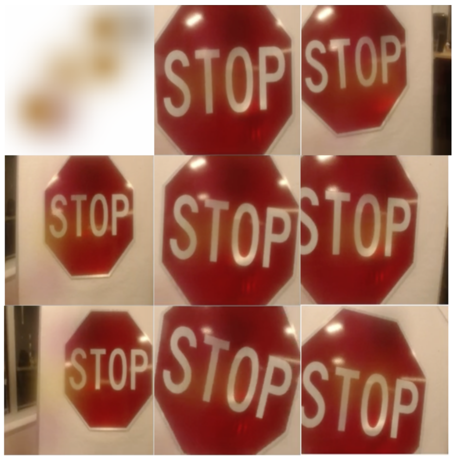


Figure 8.7: Sticker perturbation to fool “street sign" class to “guitar pick" class

Figure 8.6 and 8.7 show several snapshots of the process for both the keyboard and stop sign tests. The overlay created by the adversarial sticker is visible in all cases, but relatively inconspicuous, and would be unlikely to be noticed (or more likely discarded as merely some dust on the camera) by someone not primed to look for the particular patterns. Table 8.2 shows the performance of the ResNet-50 classifier more quantitatively for all 1000 frames of each video. In this table, we break down the number of frames predicted correctly (as a keyboard or street sign), the number of frames predicted as the target class of the attack (mouse or guitar pick respectively), and the number of frames predicted as some other class. The inclusion of the target class here (and the fact that the majority of images are predicted as this class) is important: many innocuous transformations to ImageNet-like images, such as rotations, translations, or non-traditional cropping, can fool a classifier [Eng+17; AW18]. However, since we are, the majority of the time, producing the *target* class, we emphasize that this is not merely a manner of random noise fooling the classifier; rather, this specific pattern of dots is tuned precisely to produce the intended target class, and this manifests both in the digital and physical domains.

	Prediction		
Class	Correct	Targeted	Other
Keyboard	271	548 (mouse)	181
Keyboard	320	522 (space bar)	158
Street sign	194	605 (guitar pick)	201
Street sign	222	525 (envelope)	253
Coffee mug	330	427 (candle)	243

Table 8.2: Fooling performance of our method on two 1000 frame videos of a computer keyboard and a stop sign, viewed through a camera with an adversarial sticker placed on it targeted for these attacks. The "correct predictions" column indicates how many times the object is correctly classified; the "targeted attack prediction" indicates how often the target class is predicted (computer mouse and guitar pick for the two classes respectively); and the third column indicates how often another class is predicted.

8.3.4 Other experiments

Finally, because there are several aspects of interest regarding both the power of the thread model we consider and our ability to physically manufacture such dots, we here present additional evaluations that highlight aspects of the setup.

Effect of the number of dots Although most of the aspects of our dots (besides color and position) are fixed by fitting them to observed data, one free parameter that we *do* have at our disposal is the number of dots we place on the sticker. Table 8.3 shows the targeted fooling rate for the computer $->$ keyboard to computer $->$ mouse attack (here evaluated in the digital realm on the ImageNet test set), varying the number of dots. As expected, additional dots increases the fooling rate, without reaching diminishing returns even at 10 dots. Thus, the limiting factor in the attack is largely related to how many dots we can physically print while remaining indistinct; for the physical attacks in the previous section, for this reason, we limited the stickers to 6 dots (which also highlights that the fooling rate on these real instances is substantially higher than for the ImageNet test set).

Number of Dots	Targeted Fooling Rate
1	27.9%
3	32.0%
5	34.2%
7	38.2%
10	49.6%

Table 8.3: Number of Dots and Targeted Fooling Rate against ResNet-50 model on the ImageNet test set

Effect of printed dots on camera perturbations Last, although these are large points of discussion, we highlight some important properties about the effects of the physical printed dots on the perturbations observed by the camera. It should be noted above that all our physically manufactured stickers, such as those shown in Figure 8.3, are (small) solid opaque dots, which we then place over the camera lens. A natural question arises as to whether we could create even smaller dots in the perturbation space by simply printing a smaller dot. However, this does not work: due to the optics of the camera, after a certain point a smaller size printed dot does not result in a smaller size visual dot in the camera, but rather merely a *more transparent* dot of the same size. This process is shown in Figure 8.8, and also highlights why we opt to use small solid printed dots to fit our threat model, rather than allowing for the printed dots to be transparent.

8.4 Audio Adversarial Attack: Adversarial Music

Systems that use voice and audio such as Amazon Alexa, Google Assistant, and Microsoft Cortana are growing in popularity. The hidden risk of those advancements is that those systems are potentially vulnerable to adversarial attacks from an ill-intended third party. Despite the recent growth in the consumer presence of audio-based artificial intelligence products, attacks on audio and speech systems have received much less attention than the image and language domains so far.

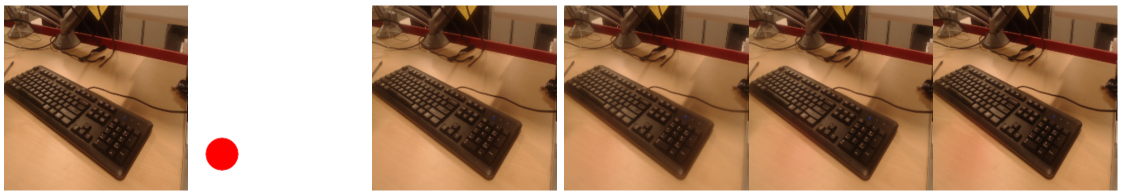

(a) Background (b) Sticker (c) r=0.011in (d) r=0.015in (e) r=0.02in (f) r=0.025in

Figure 8.8: Resulting effects of a single red dotted sticker with different radius (the dot is applied to the bottom left corner of the camera view)

Despite a number of recent works attempting to create adversarial examples against Automatic Speech Recognition (ASR) systems [CW18; Sch+18; Qin+19], we believe robust playable-over-the-air real-time audio adversaries against commercial ASR systems still do not exist. There exists no adversary that can be played from a different speaker rather than the source. Moreover, as consumers facing products, Voice Assistants (VAs) such as Amazon Alexa and Google Assistant are well-maintained by their infrastructure teams. It is revealed that these teams retrain and update ASR models very frequently on their cloud back-end, making a robust audio adversary that can consistently work against these ASR systems almost impossible to craft. Attackers would not only lack knowledge of the backend models' parameters and gradients, but would also struggle to keep up with the ever-evolving models.

However, all the existing VAs rely solely on wake-word detection to respond to people's commands, which could potentially make them vulnerable to audio adversarial examples. In this chapter, rather than directly attacking the general ASR models, we target our attack on the wake-word detection system. Wake-word detection models always have to be stored and executed onboard within smart-home hardware, which is usually very limited in terms of computing power due to form factors. It is also revealed that updates to the wake-word models are much more infrequent and difficult compared with the backend ASR models. Thus, our proposed attack could be particularly more damaging. Our goal is to jam the model so as to deactivate the VAs while our audio adversary is present. Specifically, we create a parametric attack that resembles a piece of background music, making the attack inconspicuous to humans.

We reimplemented the wake-word detection system used in Amazon Alexa based on their latest publications on the architecture [Wu+18]. We leveraged a large amount of open-sourced speech data to train our wake-word model, testing and making sure it has on-pair performance compared with the real Alexa. We collected 100 samples of "Alexa" utterances from 10 people and augmented the data set by varying the volume, tempo and speed. We created a synthetic data set using publicly available data sets as background noise and negative speech examples. This collected database is used to validate our emulated model and be compared with the real Alexa.

After successfully training a high-performance emulated wake-word model, we synthesized guitar-like music to craft our audio perturbations. Projected gradient descent (PGD) is used here to maximize the effect of our attack while keeping the parameters of our adversarial music within realistic boundaries. During the training of our audio perturbation, we considered expectation over transforms [Ath+18] including psychoacoustic masking and room impulse responses. Finally, we tested our perturbation over the air against our emulated model in parallel with the real Amazon Echo and verified that our perturbation works effectively against both of them in the real world. Specifically, the recognition F1 score of our emulated model was reduced from 93.4% to 11.0%, and the F1 score of the real Alexa was also brought down from 92.5% to 11.0%. Our adversarial music poses a real threat against commercial-grade VAs, leading to many safety concerns such as distraction in driving and malicious manipulations of smart devices.

8.5 Audio Adversarial: Background

Following successful demonstrations in the vision domain, adversarial attacks were also successfully applied to natural language processing [Pap+16; Ebr+18; RK16; Iyy+18; Nai+18]. This trend gives rise to defensive systems such as Cisse et al. [Cis+17] and Wong and Kolter [WK18], and thus provides a guideline to the community about how to build robust machine learning models.

Attacks on audio and speech systems have received much less attention so far. As recently as last year, Zhang et al. [Zha+17a] pioneered a proof-of-concept that proved the feasibility of real-world attacks on speech recognition models. This work however, had a larger focus on the hardware part of the Automatic Speech Recognition (ASR) system, instead of its machine learning component. Until very recently, there was not much work done on exploring adversarial perturbation on speech recognition models. Carlini et al. [Car+16] was the first to demonstrate that attack against HMM models are possible. They claimed to effectively attack based on the inversion of feature extractions. This work was preliminary since it only showcased a limited number of discrete voice commands, and the majority of perturbations are not able to be played over the air. As a follow-up work, Carlini and Wagner [CW18] and Qin et al. [Qin+19] showcased that curated white-box attacks based on adversarial perturbation can easily fool the Mozilla speech recognition system. Again, their attacks would only work in with their special setups and are very brittle in the real world. Meanwhile, Schönherr et al. [Sch+18] attempted to leverage psycho-acoustic hiding to improve the chance of success of playable attacks. They claimed to have verified their attacks against the Kaldi ASR system, whereas the real-world success rate was still not satisfying, and the adversary itself cannot be played from a different source. Unfortunately, state-of-the-art audio adversaries can only work digitally against the specific model they were trained against, but cannot work robustly over the air against state-of-the-art commercial-grade ASR systems. We verified all these adversaries mentioned above are not effective against Alexa or Siri. Alexa and Siri can still transcribe correctly in the presence of these audio perturbations. We took a different approach that did not involve facing the ASR models deployed in commercial products directly. Our proposed attack targets the more manageable but equally critical wake-word detection system and effectively demonstrates that it can be playable over the air.

8.6 Synthesizing Adversarial Music against Alexa

This section contains the main methodological contributions of our paper: the algorithmic and practical pipeline for synthesizing our adversarial music. To begin, we will first describe the training setup and the performance of our emulated wake-word detection model. We then describe the general threat model we consider in this chapter. Unlike past works which often considered ℓ_p norm bounded perturbations on the entire spectrum, we require a threat model that can generate a piece of audible music. Next, we describe the approach we used to make our threat model robust over the air. Finally, we described how we optimized the parameters of our threat model to synthesize a robust over-the-air attack.

8.6.1 Emulate the Wake-word Detection Model

Wake-word detection is the first important step before any interactions with distant speech recognition. Due to the compacted space of embedded platforms and the need for quick reflection time, models of wake-word detection are usually truncated and vulnerable to attack. Thus, we target our attack at the wake-word detection function.

Since the Alexa model is a black-box to us, the only way to attack it is either to estimate its gradients, or to emulate its architecture and later transfer the white-box attack against the emulated model to its original model. Estimating its gradient using first-order optimization techniques would be extremely computationally expensive [IEM18], making it difficult if not impossible to implement. Luckily, the architecture of Amazon Alexa was published in Panchapagesan et al. [Pan+16b], Kumatani et al. [Kum+17], and Guo et al. [Guo+18], allowing us to emulate the model as if it is a white-box attack. We implemented the time-delayed bottleneck highway networks with Discrete Fourier Transform (DFT) features as shown in Figure8.9. The architecture is following the details in Guo et al. [Guo+18], which is the most up-to-date information on the model architecture.

The architecture of the emulated model is shown in Figure 8.9. The model is a two-level architecture that uses the highway block as the basic building block. Specifically, our architecture contains a 4-layer highway block as a feature extractor, a linear layer acting as the bottleneck, a temporal context window that concatenates features from adjacent frames, and a 6-layer highway block for classification.

The training data for wake-word detection systems is very limited, so our model is first pre-trained with several large corpora [CMW04; GHM92; RDE12] to train a general acoustic model. Then it is adapted to the wake-word detection model by using a small amount of wake-word detection training data. The emulated model could detect the wake-word “Alexa" by recognizing the corresponding phonemes of AX, L, EH, K, S and AX.

We trained the model as a binary classification problem over a time sequence, distinguishing between the wake-word and non-wake-word. The performance is evaluated over a reserved test set. Care has been taken to ensure that augmented copies of the same raw audio sample will not occur in the train and test set simultaneously. Common performance metrics are listed in Table 8.4. Given that our emulated model's architecture is very similar to the Alexa model, the gradients of the two models should also share great similarities. We believe that the white-box attack computed by gradient-based attacks against our emulated model would be effective against Alexa's original model. Figure 8.10 shows the Detection Error Tradeoff (DET) in a similar style as [Guo+18]. We note that our performance is not exactly the same as Alexa model due to the lack of training data, but our performance is in the same order of magnitude. Thus, we believe we should have a high-fidelity emulation of the Alexa wake-word detection model.

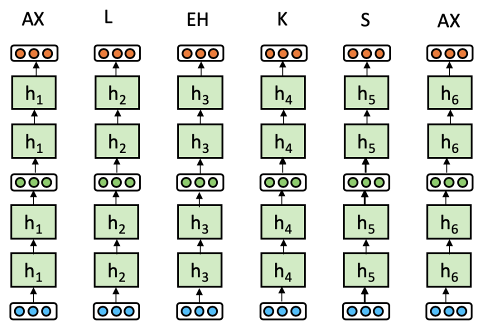


Figure 8.9: Emulated Wake-word model

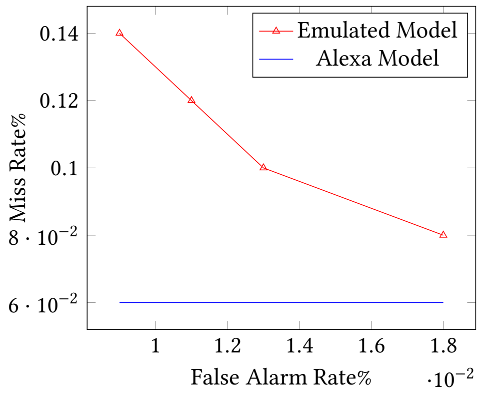


Figure 8.10: Detection Error Tradeoff Curve. The curve of Alexa model is shown in a flat line as its False Alarm Rate is not published

8.6.2 Adversarial Music Synthesizer (Threat Model)

To perform the adversarial attack on the audio domain, we introduce a parametric model to define a realistic construction of our adversary δ_θ parameterized by θ. We use the

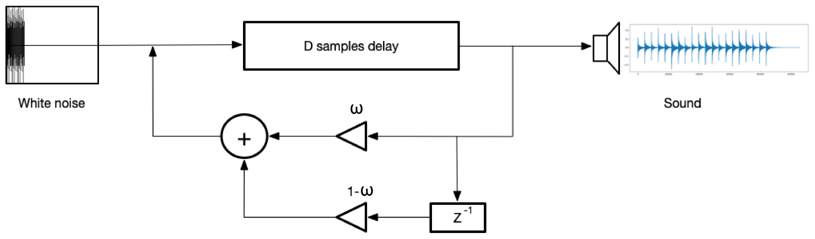


Figure 8.11: String instruments with the one-zero low-pass filter approximation. The synthesis process first generates a short excitation D-length waveform. It is then fed into the filter iteratively to generate the sound.

Karplus-Strong algorithm to synthesize guitar-timbre sounds. The goal is to generate a sequence of L guitar notes $\{(fr_i, d_i, vol_i)\}_{i=1}^{L}$, with given Beats Per Minute (BPM) bpm, Sampling Frequency fr_s, where fr_i is the frequency (key) of the i^{th} note, d_i is the note duration, and vol_i is the note volume. In our experiment, we update the $\theta = \{(fr_i, vol_i)\}_{i=1}^{L}$ of all notes' frequency fr and volume vol, while fixing the notes' duration $\{d_i\}_{i=1}^{L}$.

The Karplus-Strong algorithm is an example of digital waveguide synthesis to simulate string instruments physically. It mimics the dampening effects of a real guitar string as it vibrates by taking the decaying average of two consecutive samples: $y[n] = \gamma(y[n-D] + y[n-D+1])/2$, where γ is the decay factor, y is the output wave, D is the sampling phase delay. This is similar to a one-zero low-pass filter with a transfer function $H(z) = (1+z^{-1})/2$ [JS83], as shown in Figure. 8.11.

When a guitar string is plucked, the string vibrates and emits sound. To simulate this, the Karplus-Strong algorithm consists of two phases, as shown in Figure. 8.11:

Plucking the string: the string is "plucked" by a random initial displacement and initial velocity distribution $y[0 : D-1] \sim \mathcal{N}(0, \beta)$, where β is the displacement factor. The plucking time, i.e. the sampling phase delay D for the note frequency fr is calculated using $D = fr_s/fr$.

The resulting vibrations: The pluck causes a wave-like displacement over time with a decaying sound. The decay factor depends on the note frequency fr and the delay period $n_D = d \times fr/fr_s$. It is calculated as: $\gamma = (4/log(fr))^{1/n_D}$. The volume of the output of a note is adjusted by $v_{output} = p_v \times vol$ using a frequency-specific volume factor $p_v = 1 + 0.8 \times (log(fr) - 3)/5.5 \times cos(\pi/5.3(log(fr) - 3))$ [Woo04]. Jaffe and Smith [JS83] suggested using linear interpolation to obtain a fractional delay to generate a better result:

$$[ht!]y[n] = \gamma v_{output}(\omega \times y[n-D] + (1-\omega) \times y[n-D-1])/2, \tag{8.12}$$

where, $\omega \in (0, 1)$ is the weight factor. Overall, the Karplus-Strong algorithm is shown in algorithm. 2.

The resulting synthesizing function $\pi(x; \theta)$ is a differentiable function of part of the parameters θ value is a continuous function of the parameters: the frequency f and the

Algorithm 2: Karplus-Strong algorithm

```
1 Simulate the plucking phase of each note $i$ by initialize $y_i[0 : D-1] \sim \mathcal{N}(0, \beta)$;
2 for $i = 1, ..., L$ do
      for $n = D, ..., d_i$ do
          $y_i[n] = \gamma v_{output}(\omega \times y_i[n-D] + (1-\omega) \times y_i[n-D-1])/2$;
3 Return $y$;
```

volume *vol* of each note (We fix the note duration d). Therefore, we can implement the perturbation model within an automatic differentiation toolkit (we implement it in the PyTorch library), a feature that will be exploited to both fit the parametric perturbation model to real data and to construct real-world adversarial attacks.

8.6.3 Psychoacoustic Effect

Our ultimate task is to deceive the voice assistant with minimal impact on human hearing, and it is natural to leverage the psychoacoustic effect of human hearing. The principles of the psychoacoustic model are similar to what is used in the compression process of audio files, e.g. compress the loose-less file format "wav" to the loose file format "mp3". In this process, the information carried by the audio file is actually altered while the human ear could not tell the differences between these two sounds. Specifically, a louder signal (the “masker") can make other signals at nearby frequencies (the “maskees") imperceptible [Mit04]. In this Chapter, we adopted the same setup as [Qin+19]. When we add a perturbation δ, the normalized power spectral density (PSD) estimate of the perturbation $\bar{p}_\delta(k)$ is under the frequency masking threshold of the original audio $\eta_x(k)$,

$$\bar{p}_\delta(k) = 92 - \max_k p_x(k) + p_\delta(k) \tag{8.13}$$

where $p_\delta(k) = 10\log_{10}|\frac{1}{N}s_\delta(k)|^2$, $p_x(k) = 10\log_{10}|\frac{1}{N}s_x(k)|^2$ are power spectral density estimation of the perturbation and the original audio input. $s_x(k)$ is the k_{th} bin of the spectrum of frame x. This results in the loss function term:

$$\mathcal{L}_\eta(x,\delta) = \frac{1}{|\frac{N}{2}|+1}\sum_{k=0}^{|\frac{N}{2}|}\max\{\bar{p}_\delta(k) - \eta_x(k), 0\} \tag{8.14}$$

where N is the predefined window size and $[x]$ outputs the greatest integer no larger than x.

8.6.4 Expectation Over Transform

When using the voice assistant in a room, the real sound caught by the microphone includes both the original sound spoken by the human and the reflected sound. The "room impulse response" function explained the transformation of the original audio and the audio caught by the microphone. Therefore, to make our adversarial attack effective in the physical domain, i.e. attack the voice assistant over the air, it is necessary to consider the room impulse response in our work. Here, we use the classic Image Source Method introduced in [AB79; SBD18] to create the room impulse response r based on the room configurations (dimension, source location, reverberation time): $t(x) = x \times r$, where x denotes clean audio and $\times$ denotes convolution operation. The transformation function t follows a chosen distribution $\mathcal{T}$ over different room configurations.

8.6.5 Projected Gradient Descent and Loss Formulation

Originally, the wake-word detection problem is formulated as a minimization of: $E_{x,y\sim D}[\mathcal{L}(f(x), y)]$ where $\mathcal{L}$ is the loss function, f is the classifier mapping from input x to label y, and D is the data distribution. We evaluate the quality of our classifier based on the loss, and a smaller loss usually indicates a better classifier. In this Chapter, since we are interested in attack, we form $E_{x,y\sim D}[\max_{x'} \mathcal{L}(f(x'), y)]$, where $x' = x + \delta$ is our perturbed audio. In a completely differentiable system, an immediately obvious initial approach to this would be to use gradient ascent in order to search for an x' that maximizes this loss. However, for this maximization to be interesting both practically and theoretically, we need x' to be close to the original datapoint x, according to some measure. It is thus common to define a perturbation set $C(x)$ that constrains x', such that the maximization problem becomes $\max_{x'\in C(x)} \mathcal{L}(f(x'), y)$. The set $C(x)$ is usually defined as a ball of small length (of either ℓ_∞, ℓ_2 or ℓ_1) around x.

Since we have to solve such a constrained optimization problem, we cannot simply apply the gradient descent method to maximize the loss, as this could take us out of the constrained region. One of the most common methods utilized to circumvent this issue is called Projected Gradient Descent (PGD). To actually implement gradient descent methods, we will invert the sign of the aforementioned problem to minimize the negative loss, *i.e.*, $\min_{x'\in C(x)} -\mathcal{L}(f(x'), y)$.

Overall, the loss function of the vanilla PGD is:

$$\max_{z\in C(x)} \mathcal{L}(x, y) = \mathcal{L}_{WW}(f(x')) - \alpha\|x' - z\|^2 \tag{8.15}$$

where L_{WW} is the original loss function for the wake-word model, and α is the balancing parameter. In our case, we attack the voice assistant via a parametric threat model and consider the psychoacoustic effect, as illustrated in Section 8.6.3 and Section 8.6.2. The loss function of attack thus can be rewritten as:

$$\max \mathcal{L}(x, \delta_\theta, y) = \mathcal{L}_{WW} f(x + \delta_\theta) - \alpha \cdot \mathcal{L}_\eta(x, \delta) \tag{8.16}$$

Here, δ_θ is our adversarial music defined in Section 8.6.2. In addition, we consider the room impulse response defined in Section 8.6.4, which would be the attack in the physical world over the air. We simulated our testing environments using the parameters shown in Table 8.5 to transform $t(x)$ our digital adversary to compensate for the room impulse responses. We want to optimize our δ by tuning θ to maximize this loss to synthesize our adversary:

$$\max \mathcal{L}(x, \delta_\theta, y) = \mathbb{E}_{t \in \mathcal{T}}[\mathcal{L}_{WW} f(t(x + \delta_\theta)) - \alpha \cdot \mathcal{L}_\eta(x, \delta)] \tag{8.17}$$

Here, y is the ground truth label of the audio, and $\mathcal{L}_{WW}$ is the original loss of the emulated model for wake-word detection.

8.7 Audio Experiments and Results

Datasets

We collected 100 positive speech samples (speaking "Alexa") from 10 people (4 males and 6 females; 4 native speakers of English, 6 non-native speakers of English). Each person provided 10 utterances, under the requirement of varying their tone and pitch as much as possible. We further augmented the data to 20x by varying the speed, tempo, and volume of the utterance, resulting in 2000 samples. We used the LJ speech dataset [Ito17] for background noise and negative speech examples (speak anything but "Alexa"). We created a synthetic data set by randomly adding positive and negative speech examples onto a 10s background noise and created binary labels accordingly. While "hearing" positive speech examples, we set label values as 1.

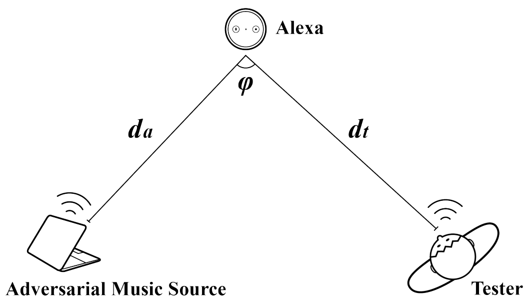


Figure 8.12: Physical Testing Illustration

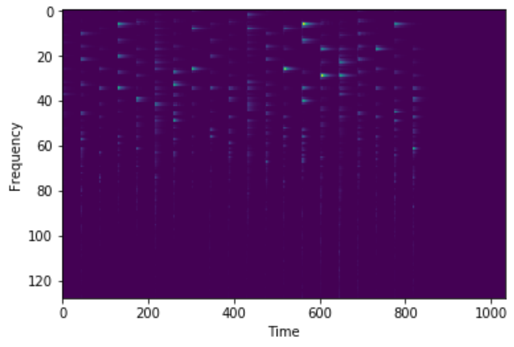


Figure 8.13: A sample adversarial music

Training and Testing the Adversarial Music

We followed the methodology described in Section 8.6.5, using the loss function defined by Eq. 8.17 and the parametric method illustrated in Section 8.6.2, we optimized the parameters of the music to maximize our attack. We used the emulated model developed in Section 8.6.1 to estimate the gradient and maximize the classification loss following Eq. 8.17. When using PGD to train, we restricted the frequency to 27.5Hz $\sim$ 4186Hz in the 88 notes space and restricted the volume from 0 dB $\sim$ 100 dB. Other parameters are defined in the code, we fixed some parameters to speed up the training.[1] The trained perturbation are directly added on top of the clean signal to perform our digital evaluation. Our physical experiments are conducted at a home environment which can be seen in the video attached in the supplementary material, and the setup is shown in Figure 8.12. The adversarial music is played by a MacBook Pro (15-inch, 2018) speaker. The tester stands d_t away from the Amazon Echo or the computer running our emulated model, and the adversary is placed d_a away from the Echo. The angle between the two lines is ϕ. We tested against the model for 100 samples with and without the audio adversary. We used a decibel meter to ensure that our adversary is never louder than the wake-word command. Our human spoken wake-word command is measured to be 70dB on average. We experimented with adversaries being played at 60dB and 70dB (measured by the decibel meter), and we also experimented with 2 different testers (a male and a female). [2]

The performance metrics of the emulated model on adversary examples are shown in Table 8.4. An example of a modified adversarial attack example is shown in Figure 8.13. As we can see, our attack is effective against both the emulated model and the real Alexa model in both digital and physical tests. Especially, the precision score takes a heavier hit by the adversary compared with the recall score. We also noticed that False Positives are relatively uncommon, this might be due to the fact that we are running an un-targeted attack against the wake-word detection, and our adversaries are not encouraging the model to predict a specific target label. Overall, our experiments showcased that audio adversaries masked as a piece of music can be played over the air, and disable a commercial-grade wake-word detection system.

In our physical experiments against the real Alexa, we measured the performance of our audio perturbation under several different physical settings shown in Table 8.5. As we can observe, our model successfully attacked Alexa in most cases. The distance d_a between Alexa and the attacker is critical for a successful attack: a shorter distance d_a generally leads to a more effective adversary, while the number of successful attacks declines when distance d_t is shorter. The adversary also tended to be more effective when being played with a louder volume. We observed it would not be effective being played under 40dB (which is a quiet room setting). Our adversaries successfully demonstrated to be effective at various physical locations and angles ϕ referenced to Alexa and the human

[1]We will release our code upon acceptance to encourage reproducibility
[2]please refer to the demo video in the supplementary material for more graphical information

tester, the larger ϕ the less effective our adversary would be. This is exciting since Alexa's 7-microphone array is supposedly to be very robust at source separation. This can prove that our consideration of the room impulse responses was useful. When we remove the room impulse transform, the adversary loses its effect.

Last and most importantly, we verified that a random piece of music being played at the same volume as our adversarial music cannot affect Alexa. We picked 10 different top-ranking guitar music on Youtube and conducted the same test as Table 8.5. We verified that Alexa's wake-word detection was almost not affected by the presence of these random musics.[3]

Models	Attack	Digi/Phys	Precision	Recall	F1 Score	#Samples
Emulated Model	No	Digital	0.97	0.94	0.955	4000
Emulated Model	No	Physical	0.96	0.91	0.934	100
Alexa	No	Physical	0.93	0.92	0.925	100
Emulated Model	Yes	Digital	0.14	0.11	0.117	4000
Emulated Model	Yes	Physical	0.12	0.09	0.110	100
Alexa	Yes	Physical	0.11	0.10	0.110	100

Table 8.4: Performance of the models with and without attacks in digital and physical testing environments given the number of testing samples

Test Against Alexa	$\phi = 0°$			$\phi = 90°$			$\phi = 180°$		
$d_t =$	$4.2ft$	$7.2ft$	$10.2ft$	$4.2ft$	$7.2ft$	$10.2ft$	$4.2ft$	$7.2ft$	$10.2ft$
$d_a = 4.7ft, 70dB$	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10
$d_a = 6.2ft, 70dB$	1/10	0/10	0/10	1/10	0/10	0/10	1/10	2/10	1/10
$d_a = 7.7ft, 70dB$	2/10	0/10	0/10	3/10	1/10	1/10	3/10	3/10	1/10
$d_a = 4.7ft, 60dB$	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10
$d_a = 6.2ft, 60dB$	1/10	1/10	0/10	3/10	1/10	0/10	2/10	2/10	0/10
$d_a = 7.7ft, 60dB$	2/10	1/10	0/10	3/10	2/10	1/10	4/10	3/10	1/10

Table 8.5: Times of the real Amazon Alexa being able to respond to the wake-word under the influence of our adversarial music with different settings.
(The female and male tester each tests 5 utterances.) The testing set up is illustrated in Figure. 8.12, and it is also showed in the demo video. In this context, dB indicates dB SPL.

[3]This is shown in our demo video

8.8 Conclusion

In this chapter, we demonstrated that it is possible to adversarially fool deep image classifiers in the real world, not by modifying an object of interest itself, but by modifying the camera observing the objects. The optics involved with such modifications greatly limits the types of attacks that can be physically realized, but by developing a reasonable threat model and then fitting the parameters of this model to data, we can accurately capture the allowable set of perturbations. By then optimizing over this class to construct (universal) adversarial examples, we show that it is possible to perform targeted adversarial attacks on real objects, using a single adversarial sticker to misclassify an object in multiple different orientations and scales.

In the audio domain, we deactivated the wake-word detection system on Amazon Alexa with real-world inconspicuous adversarial background music. We first created an emulated model for the wake-word detection on Amazon Alexa following the implementation details published in Guo et al. [Guo+18]. The model is the basis to design our white-box attacks. We collected, augmented, and synthesized a data set for training and testing. We implemented our threat model which disguises our attack in a sequence of inconspicuous background music notes in PyTorch. Our experiments verified that our adversarial music is not only effective against our emulated model digitally, but also can effectively disable Amazon Alexa's wake-word detection function over the air. Our work shows the potential of adversarial attacks in the audio domain, specifically, against the widely applied wake-word detection systems in voice assistants. Overall, this suggests a real concern of attack against commercial-grade machine learning algorithms, highlighting the importance of adversarial robustness from a practical, security-based point of view.

Chapter 9

Robust Multi-Modal Fusion

9.1 Introduction

The objective of this chapter is to extend our discussion from Chapter 5, focusing on an empirical investigation into the optimal fusion strategy that can both maximize performance and maintain robustness. We study this via the task of audio-visual text retrieval: to retrieve the correlated text description given a random video, and vice versa, to retrieve the matching videos provided with text descriptions (See Fig. 9.1). While several computer vision tasks (e.g., image classification [He+16b; MPRC16; Hua+17], object detection [Ren+15; Red+16; Mit+18b]) are now reaching maturity, cross-modal retrieval between visual data and natural language description remains a very challenging problem [Zha+17b; Mit+18a] due to the gap and ambiguity between different modalities and availability of limited training data. Some recent works [Mit+18c; Kle+15; WLL16; KFF15; Fro+13] attempt to utilize cross-modal joint embeddings to address the gap. By projecting data from multiple modalities into the same joint space, the similarity of the resulting points would reflect the semantic closeness between their corresponding original inputs. In this Chapter, we focus on learning joint video-text embedding models and combining video cues for different purposes effectively for developing a robust video-text retrieval system.

The video-text retrieval task is one step further than the image-text retrieval task, which is a comparatively well-studied field. Most existing approaches for video-text retrieval are very similar to the image-text retrieval methods by design and focus mainly on the modification of loss functions [DLS16; Xu+15; TTS16; Ota+16; Pan+16a]. We observe that simple adaptation of a state-of-the-art image-text embedding method [Fag+18] by mean-pooling features from video frames generates a better result than existing video-text retrieval approaches [DLS16; Ota+16]. However, such methods ignore lots of contextual information in video sequences such as temporal activities or specific scene entities, and thus they often can only retrieve some generic responses related to the appearance of a

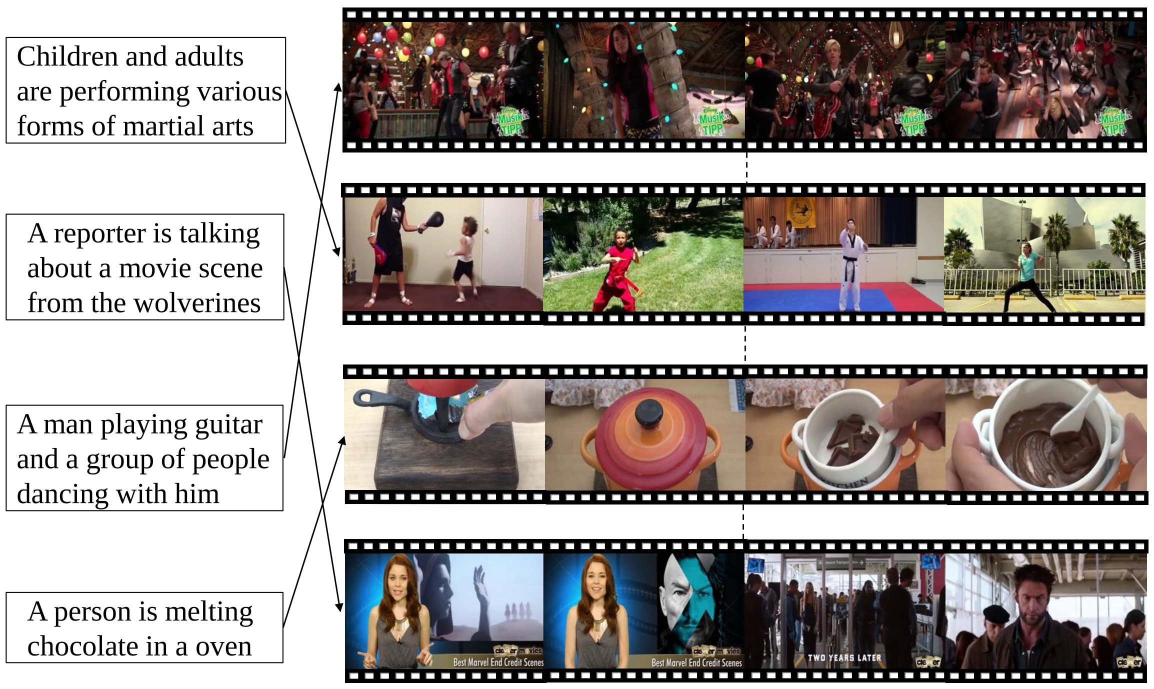

Children and adults are performing various forms of martial arts
A reporter is talking about a movie scene from the wolverines
A man playing guitar and a group of people dancing with him
A person is melting chocolate in a oven
Best Marvel End Credit Scenes
TWO YEARS LATER

Figure 9.1: Illustration of Video-Text retrieval task: given a text query, retrieve and rank videos from the database based on how well they depict the text, and vice versa.

static frame. They may fail to retrieve the most relevant information in many cases to understand important questions for efficient retrieval such as ‘What happened in the video’, or ‘Where did the video take place’. This greatly undermines the robustness of the systems; for instance, it is very difficult to distinguish a video with the caption “a dog is barking" apart from another “a dog is playing" based only on visual appearance (See Fig. 9.2). Associating video motion content and the environmental scene can give supplementary cues in this scenario and improve the chance of correct prediction. Similarly, understanding a video described by a “gunshot broke out at the concert" may require analysis of different visual (e.g., appearance, motion, environment) and audio cues simultaneously. On the other hand, a lot of videos may contain redundant or identical contents, and hence, an efficient video-text retrieval should utilize the most distinct cues in the content to resolve ambiguities in retrieval.

While developing a system without considering most available cues in the video content is unlikely to be comprehensive, an inappropriate fusion of complementary features could adversely increase ambiguity and degrade performance. Additionally, existing hand-labeled video-text datasets are very small and very restrictive considering the amount of rich descriptions that a human can compose and the enormous amount of diversity in the visual world. This makes it extremely difficult to train deep models to understand videos

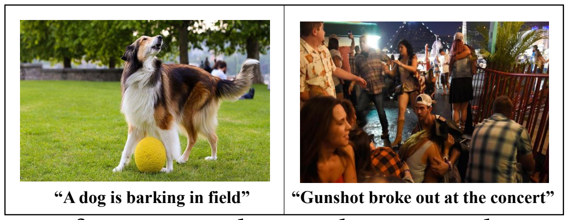

"A dog is barking in field"
"Gunshot broke out at the concert"

Figure 9.2: Example frame from two videos and associated caption to illustrate the significance of utilizing supplementary cues from videos to improve the chance of correct retrieval.

in general to develop a successful video-text retrieval system. To ameliorate such cases, we analyze how to judiciously utilize different cues from videos. We propose a mixture of experts system, which is tailored towards achieving high performance in the task of cross-modal video-text retrieval. We believe focusing on three major facets (i.e., concepts for Who, What, and Where) from videos is crucial for efficient retrieval performance. In this regard, our framework utilizes three salient features (i.e., object, action, place) from videos (extracted using pre-trained deep neural networks) for learning joint video-text embeddings and uses an ensemble approach to fuse them. Furthermore, we propose a modified pairwise ranking loss for the task that emphasizes on hard negatives and relative ranking of positive labels. Our approach shows significant performance improvement compared to previous approaches and baselines.

Contributions: This chapter covers the following main points:

• The success of video-text retrieval depends on more robust video understanding. This paper studies how to achieve the goal by utilizing multimodal features from a video (different visual features and audio inputs). Our proposed framework uses action, object, place, text and audio features by a fusion strategy for efficient retrieval.

• We present a modified pairwise loss function to better learn the joint embedding which emphasizes hard negatives and applies a weight-based penalty on the loss based on the relative ranking of the correct match in the retrieval.

• We conduct extensive experiments and demonstrate a clear improvement over the state-of-the-art methods in the video-to-text retrieval tasks on the MSR-VTT dataset [Xu+16] and MSVD dataset [CD11].

This Chapter is an extended version of our work [Mit+18a] with significantly more insights and detailed discussions about the proposed framework. The main extension in our pipeline is adding scene cues from videos, along with object and activity cues for learning joint embeddings to develop a more comprehensive video-text retrieval system. The previous version utilized object-text and activity-text embeddings which focused mainly on resolving ambiguities arising related to concepts for Who and What. We add a place-text embedding network in our framework to make it more robust which will help us resolve ambiguities arising from concepts for Where. Experiments show that this change results in a significant improvement over the previous works in two benchmark datasets.

9.2 Related Work

Image-Text Retrieval. Recently, there has been significant interest in learning robust visual-semantic embeddings for image-text retrieval [Mit+18c; KJL14; HE17; Wan+17]. Based on a triplet of object, action and, scene, a method for projecting text and image to a joint space was proposed in early work [Far+10]. Canonical Correlation Analysis (CCA) and several extensions of it have been used in many previous works for learning joint embeddings for the cross-modal retrieval task [SFF10; HYH13; Gon+14; YM15; QNTLBC16; HSST04] which focuses on maximizing the correlation between the projections of the modalities. In [Gon+14], authors extended the classic two-view CCA approach with a third view coming from high-level semantics and proposed an unsupervised way to derive the third view from clustering the tags. In [QNTLBC16], authors proposed a method named MACC (Multimedia Aggregated Correlated Components) aiming to reduce the gap between cross-modal data in the joint space by embedding visual and textual features into a local context that reflects the data distribution in the joint space. Extension of CCA with deep neural networks named deep CCA (DCCA) has also been utilized to learn joint embeddings [YM15; And+13], which focus on learning two deep neural networks simultaneously to project two views that are maximally correlated. While CCA-based methods are popular, these methods have been reported to be unstable and incur a high memory cost due to the covariance matrix calculation with large-amount of data [Wan+18; MLF15]. Recently, there are also several works leveraging adversarial learning to train joint image-text embeddings for cross-modal retrieval [Wan+17; CP18].

Most recent works relating to text and image modality are trained with ranking loss [KSZ14; Fro+13; Wan+18; Fag+18; NHK17; Ven+16]. In [Fro+13], authors proposed a method for projecting words and visual content to a joint space utilizing ranking loss that applies a penalty when a non-matching word is ranked higher than the matching one. A cross-modal image-text retrieval method has been presented in [KSZ14] that utilizes triplet ranking loss to project image feature and RNN-based sentence description to a common latent space. Several image-text retrieval methods have adopted a similar approach with slight modifications in input feature representations [NHK17], similarity score calculation [Wan+18], or loss function [Fag+18]. VSEPP model [Fag+18] modified the pair-wise ranking loss based on violations caused by the hard-negatives (i.e., non-matching query closest to each training query) and has been shown to be effective in the retrieval task. For image-sentence matching, a LSTM-based network is presented in [HWW17] that recurrently selects pairwise instances from image and sentence descriptions, and aggregates local similarity. In [NHK17], authors proposed a multimodal attention mechanism to attend to sentence fragments and image regions selectively for similarity calculation. Our method complements these works that learn joint image-text embedding using a ranking loss (e.g., [KSZ14; Ven+16; Fag+18]). The proposed retrieval framework can be applied to most of these approaches for improved video-text retrieval performance.

Video Hyperlinking. Video hyperlinking is also closely relevant to our work. Given an anchor video segment, the task is to focus on retrieving and ranking a list of target videos based on the likelihood of being relevant to the content of the anchor [Awa+17; Boi+17]. Multimodal representations have been utilized widely in video hyperlinking approaches in recent years [BDG18; VRG18; Awa+17]. Most of these approaches rely heavily on multimodal autoencoders for jointly embedding multimodal data [VRG17; FWL14; CGK15]. Bidirectional deep neural network (BiDNN) based representations have also been shown to be very effective in video hyperlinking benchmarks [VRG18; VRG16]. BiDNN is also a variation of multimodal autoencoder, which performs multimodal fusion using a cross-modal translation with two interlocked deep neural networks [VRG17; VRG16]. Considering the input data, video-text retrieval is dealing with the same multimodal input as video hyperlinking in many cases. However, video-text retrieval task is more challenging than hyperlinking since it requires to distinctively retrieve matching data from a different modality, which requires effective utilization of the correlations in between cross-modal cues.

Video-Text Retrieval. Most relevant to our work are the methods that relate video and language modalities. Two major tasks in computer vision related to connecting these two modalities are video-text retrieval and video captioning. In this work, we only focus on the retrieval task. Similar to image-text retrieval approaches, most video-text retrieval methods employ a shared subspace. In [Xu+15], authors vectorize each subject-verb-object triplet extracted from a given sentence by word2vec model [Mik+13] and then aggregate the Subject, Verb, Object (SVO) vector into a sentence level vector using RNN. The video feature vector is obtained by mean pooling over frame-level features. Then a joint embedding is trained using a least squares loss to project the sentence representation and the video representation into a joint space. Web image search results of input text have been exploited by [Ota+16], which focused on word disambiguation. In [Ven+15], a stacked GRU is utilized to associate sequence of video frames to a sequence of words. In [Pan+16a], authors propose an LSTM with visual-semantic embedding method that jointly minimizes a contextual loss to estimate relationships among the words in the sentence and a relevance loss to reflect the distance between video and sentence vectors in the shared space. A method named Word2VisualVec is proposed in [DLS16] for the video to sentence matching task that projects vectorized sentence into visual feature space using mean squared loss. A shared space across image, text and sound modality is proposed in [AVT17] utilizing ranking loss, which can also be applied to video-text retrieval task.

Utilizing multiple characteristics of video (e.g., activities, audio, locations, time) is evidently crucial for efficient retrieval [YYH04]. In the closely related task of video captioning, dynamic information from video along with static appearance features has been shown to be very effective [Zha+17c; Ram+16]. However, most of the existing video-text retrieval approaches depend on one visual cue for retrieval. In contrast to the existing works, our approach focuses on effectively utilizing different visual cues and audio (if available) concurrently for more efficient retrieval.

Ensemble Approaches. Our retrieval system is based on an ensemble framework [Pol06; Fra+12]. A strong psychological context of the ensemble approach can be found from its intrinsic connection in decision making in many daily life situations [Pol06]. Seeking the opinions of several experts, weighing them, and combining them to make an important decision is an innate behavior of humans. The ensemble methods hinge on the same idea and utilize multiple models for making an optimized decision, as in our case diverse cues are available from videos and we would like to utilize multiple expert models which focus on different cues independently to obtain a stronger prediction model. Moreover, ensemble-based systems have been reported to be very useful when dealing with a lack of adequate training data [Pol06]. As the diversity of the models is crucial for the success of ensemble frameworks [Pol07], it is important for our case to choose a diverse set of video-text embeddings that are significantly different from one another.

9.3 Approach

In this section, we first provide an overview of our proposed framework (Section 9.3.1). Then, we describe the input feature representation for video and text (Section 9.3.2). Next, we describe the basic framework for learning visual-semantic embedding using pair-wise ranking loss (Section 9.3.3). After that, we present our modification on the loss function which improves the basic framework to achieve better recall (Section 9.3.4). Finally, we present the proposed fusion step for video-text matching (Section 9.3.5).

9.3.1 Overview of the Proposed Approach

In a typical cross-modal video-text retrieval system, an embedding network is learned to project video features and text features into the same joint space, and then retrieval is performed by searching the nearest neighbor in the latent space. Since in this work we are looking at videos in general, detecting most relevant information such as object, activities, and places could be very conducive for higher performance. Therefore, along with developing algorithms to train better joint visual-semantic embedding models, it is also very important to develop strategies to effectively utilize different available cues from videos for a more comprehensive retrieval system.

In this chapter, we study how to leverage the capability of neural networks to learn a deep representation first and fuse the video features in the latent spaces so that we can

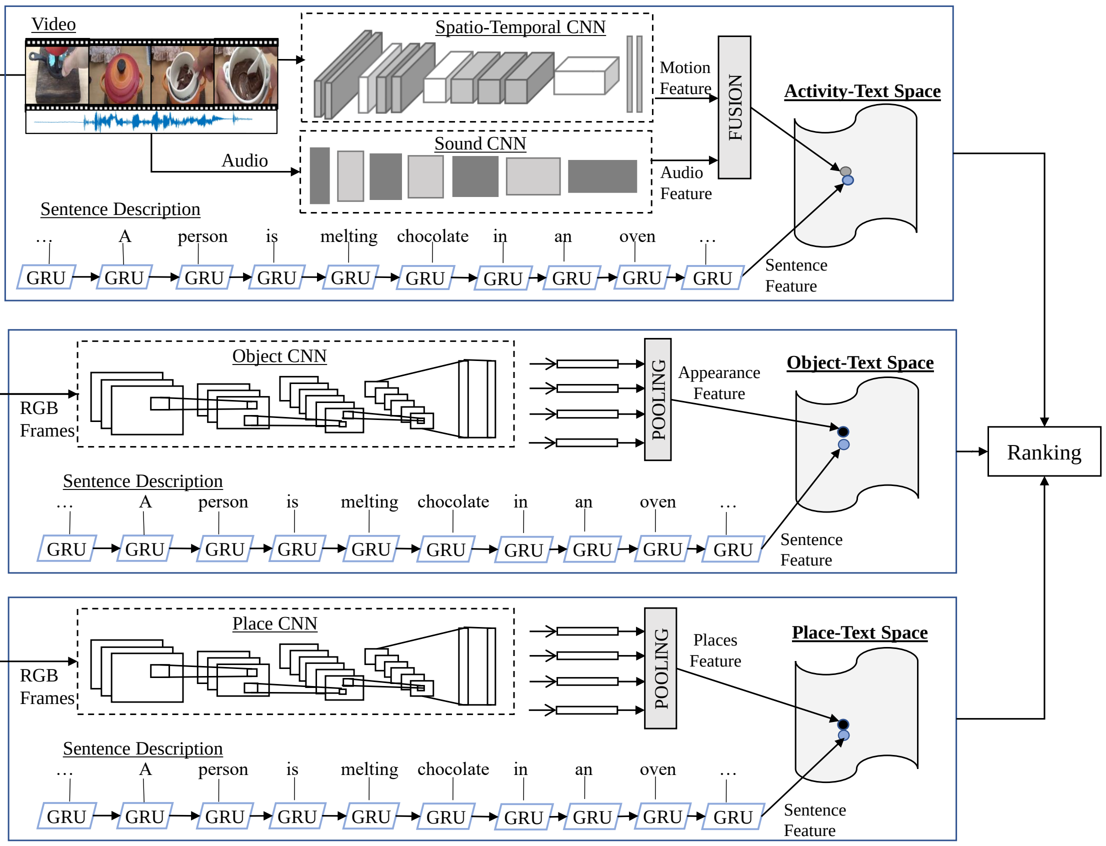


Figure 9.3: We propose to learn three joint video-text embedding networks as shown in Fig. 9.3. One model learns a joint space (Object-Text Space) between text features and visual object features. Another model learns a joint space (Activity-Text Space) between text features and activity features. Similarly, there is a third model which learns a joint space (Place-Text Space) between scene features and text features. Here, Object-Text space is the expert in solving ambiguity related to who is in the video, whereas Activity-Text space is the expert in retrieving what activity is happening and place-Text space is the expert in solving ambiguity regarding locations in the video. Given a query sentence, we calculate the sentence's similarity scores with each one of the videos in the entire dataset in all of the three embedding spaces and use a fusion of scores for the final retrieval result. Please see Section 9.3.1 for an overview and Section 9.3 for details.

develop expert networks focusing on specific subtasks (e.g. detecting activities, detecting objects). For analyzing videos, we use a model trained to detect objects, a second model trained to detect activities, and a third model focusing on understanding the place. These heterogeneous features may not be used together directly by simple concatenation to train a successful video-text model as intra-modal characteristics are likely to be suppressed in such an approach. However, an ensemble of video-text models can be used, where a video-text embedding is trained on each of the video features independently. The final retrieval is performed by combining the individual decisions of several experts [Pol06]. An overview of our proposed retrieval framework is shown in Fig. 9.3. We believe that such an ensemble approach will significantly reduce the chance of poor/wrong prediction.

We follow the network architecture proposed in [KSZ14] that learns the embedding model using a two-branch network using image-text pairs. One of the branches in this network takes text features as input and the other branch takes in a video feature. We propose a modified bi-directional pairwise ranking loss to train the embedding. Inspired by the success of ranking loss proposed in [Fag+18] in image-text retrieval task, we emphasize hard negatives. We also apply a weight-based penalty on the loss according to the relative ranking of the correct match in the retrieved result.

9.3.2 Input Feature Representation

Text Feature. For encoding sentences, we use Gated Recurrent Units (GRU) [Chu+14a]. We set the dimensionality of the joint embedding space, D, to 1024. The dimension of the word embeddings that are input to the GRU is 300. Note that the word embedding model and the GRU are trained end-to-end in this work.

Object Feature. For encoding image appearance, we adopt a deep pre-trained convolutional neural network (CNN) model trained on ImageNet dataset as the encoder. Specifically, we utilize state-of-the-art 152 layer ResNet model ResNet152 [He+16b]. We extract image features directly from the penultimate fully connected layer. We first rescale the image to 224x224 and feed it into CNN as inputs. The dimension of the image embedding is 2048.

Activity Feature. The ResNet CNN can efficiently capture visual concepts in static frames. However, an effective approach to learning temporal dynamics in videos was proposed by inflating a 2-D CNN to a deep 3-D CNN named I3D in [CZ17b]. We use I3D model to encode activities in videos. In this work, we utilize the pre-trained RGB-I3D model and extract 1024-dimensional features utilizing continuous 16 frames of video as the input.

Place Feature. For encoding video features focusing on scene/place, we utilize a deep pre-trained CNN model trained on the Places-365 dataset as the encoder [Zho+17]. Specifically, we utilize 50 layer model ResNet50 [He+16b]. We extract image features directly from the penultimate fully connected layer. We re-scale the image to 224x224 and feed it into CNN as inputs. The dimension of the image embedding is 2048.

Audio Feature. We believe that by associating audio, we can get important cues to real-life events, which would help us remove ambiguity in many cases. We extract audio features using state-of-the-art SoundNet CNN [AVT16], which provides 1024 dimensional features from input raw audio waveform. Note that, we only utilize the audio which is readily available with the videos.

9.3.3 Learning Joint Embedding

In this section, we describe the basic framework for learning joint embedding based on bi-directional ranking loss.

Given a video feature representation (i.e., appearance feature, or activity feature, or scene feature) $\overline{v}$ ($\overline{v} \in \mathbb{R}^V$), the projection for a video feature on the joint space can be derived as $v = W^{(v)}\overline{v}$ ($v \in \mathbb{R}^D$). In the same way, the projection of input text embedding $\overline{t}(\overline{t} \in \mathbb{R}^T)$ to joint space is $t = W^{(t)}\overline{t}(t \in \mathbb{R}^D)$. Here, $W^{(v)} \in \mathbb{R}^{D \times V}$ is the transformation matrix that projects the video content into the joint embedding space, and D denotes the dimension of the joint space. Similarly, $W^{(t)} \in \mathbb{R}^{D \times T}$ maps input sentence/caption embedding to the joint space. Given feature representation for words in a sentence, the sentence embedding $\overline{t}$ is found from the hidden state of the GRU. Here, given the feature representation of both videos and corresponding text, the goal is to learn a joint embedding characterized by θ (i.e., $W^{(v)}$, $W^{(t)}$ and GRU weights) such that the video content and semantic content are projected into the joint embedding space. We keep the image encoder (e.g., pre-trained CNN) fixed in this work, as the video-text datasets are small in size.

In the embedding space, it is expected that the similarity between a video and text pair to be more reflective of the semantic closeness between videos and their corresponding texts. Many prior approaches have utilized pairwise ranking loss for learning joint embedding between visual input and textual input. They minimize a hinge-based triplet ranking loss by combining bi-directional ranking terms, in order to maximize the similarity between a video embedding and the corresponding text embedding, and while at the same time, minimize the similarity to all other non-matching ones. The optimization problem can be written as,

$$\begin{aligned} \min_{\theta} \sum_{v} \sum_{t^-} & [\alpha - S(v,t) + S(v,t^-)]_+ \\ + \sum_{t} \sum_{v^-} & [\alpha - S(t,v) + S(t,v^-)]_+ \end{aligned} \tag{9.1}$$

where, $[f]_+ = max(0, f)$. t^- is a non-matching text embedding, and t is the matching text embedding for video embedding v. This is similar to text embedding t. α is the margin value for the pairwise ranking loss. The scoring function $S(v,t)$ is defined as the similarity function to measure the similarity between the videos and text in the joint embedded space. We use cosine similarity in this work, as it is easy to compute and has shown to be very effective in learning joint embeddings. [KSZ14; Fag+18].

In Eq. (9.1), in the first term, for each pair (v,t), the sum is taken over all non-matching text embedding t^-. It attempts to ensure that for each visual feature, corresponding/matching text features should be closer than non-matching ones in the joint space. Similarly, the second term attempts to ensure that text embedding that corresponds to the video embedding should be closer in the joint space to each other than non-matching video embeddings.

9.3.4 Proposed Ranking Loss

Recently, focusing on hard negatives has been shown to be effective in many embedding tasks [Fag+18; SKP15b; Man+17]. Inspired by this, we focus on hard negatives (i.e., the negative video and text sample closest to a positive/matching (v, t) pair) instead of summing over all negatives in our formulation. For a positive/matching pair (v, t), the hardest negative sample can be identified using $\hat{v} = \arg\max_{v^-} S(t, v^-)$ and $\hat{t} = \arg\max_{t^-} S(v, t^-)$. The optimization problem can be rewritten as following to focus on hard-negatives,

$$\begin{aligned} \min_{\theta} \sum_{v} & [\alpha - S(v, t) + S(v, \hat{t})]_+ \\ + \sum_{t} & [\alpha - S(t, v) + S(t, \hat{v})]_+ \end{aligned} \tag{9.2}$$

The loss in Eq. 9.2 is similar to the loss in Eq. 9.1 but it is specified in terms of the hardest negatives [Fag+18]. We start with the loss function in Eq. 9.2 and further modify the loss function following the idea of weighted ranking [UBG09] to weigh the loss based on the relative ranking of positive labels.

$$\begin{aligned} \min_{\theta} \sum_{v} & h(r_v)[\alpha - S(v, t) + S(v, \hat{t})]_+ \\ + \sum_{t} & h(r_t)[\alpha - S(t, v) + S(t, \hat{v})]_+ \end{aligned} \tag{9.3}$$

where $h(.)$ is a weighting function for different ranks. For a video embedding v, r_v is the rank of matching sentence t among all compared sentences. Similarly, for a text embedding t, r_t is the rank of matching video embedding v among all compared videos in the batch. We define the weighting function as $h(r) = (1 + \beta/(N - r + 1))$, where N is the number of compared videos and β is the weighting factor. Fig. 9.4 shows an example showing the significance of the proposed ranking loss.

It is very common, in practice, to only compare samples within a mini-batch at each iteration rather than comparing the entire training set for computational efficiency [Man+17; SKP15b; KFF15]. This is known as semi-hard negative mining [Man+17; SKP15b]. Moreover, selecting the hardest negatives in practice may often lead to a collapsed model and semi-hard negative mining helps to mitigate this issue [Man+17; SKP15b]. We utilize a batch size of 128 in our experiment.

It is evident from Eq. 9.3 that the loss applies a weight-based penalty based on the relative ranking of the correct match in the retrieved result. If a positive match is ranked

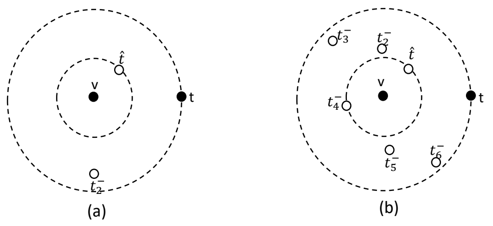


Figure 9.4: An example showing the significance of the proposed ranking loss. The idea is that if a large number of non-matching instances are ranked higher than the matching one given the current state of the model, then the model must be updated by a larger amount (Case: (b). However, the model needs to be updated by a smaller amount if the matching instance is already ranked higher than most non-matching ones. (Case: (a))

top in the list, then $h(.)$ will assign a small weight to the loss and will not cost the loss too much. However, if a positive match is not ranked top, $h(.)$ will assign a much larger weight to the loss, which will ultimately try to push the positive matching pair to the top of the rank.

9.3.5 Matching and Ranking

The video-text retrieval task focuses on returning for each query video, a ranked list of the most likely text description from a dataset and vice versa. We believe we need to understand three main aspects of each video: (1) Who: the salient objects of the video, (2) What: the action and events in the video and (3) Where: the place aspect of the video. To achieve this, we learn three expert joint video-text embedding spaces as shown in Fig. 9.3.

The Object-Text embedding space is the common space where both appearance features and text features are mapped to. Hence, this space can link video and sentences focusing on the objects. On the other hand, the Activity-Text embedding space focuses on linking video and language description which emphasizes more on the events in the video. Action features and audio features both provide important cues for understanding different events in a video. We fuse action and audio features (if available) by concatenation and map the concatenated feature and text feature into a common space, namely, the Activity-Text space. If the audio feature is absent from videos, we only use the action feature as the video representation for learning the Activity-Text space. The Place-Text embedding space is the common space where visual features focusing on scene/place aspect and text features are mapped to. Hence, this space can link video and sentences focusing on the entire scene. We utilize the same loss functions described in Sec. 9.3.4 for training these embedding models.

At the time of retrieval, given a query sentence, we compute the similarity score of the query sentence with each one of the videos in the dataset in three video-text embedding spaces and use a fusion of similarity scores for the final ranking. Conversely, given a

#	Method	Video-to-Text Retrieval					Text-to-Video Retrieval				
		R@1	R@5	R@10	MedR	MeanR	R@1	R@5	R@10	MedR	MeanR
1.1	VSE (Object-Text)	7.7	20.3	31.2	28.0	185.8	5.0	16.4	24.6	47.0	215.1
	VSEPP (Object-Text)	10.2	25.4	35.1	25	228.1	5.7	17.1	24.8	65	300.8
1.2	Ours (Object-Text)	10.5	26.7	35.9	25	266.6	5.8	17.6	25.2	61	296.6
	Ours (Audio-Text)	0.4	1.1	1.9	1051	2634.9	0.2	0.9	1.5	1292	1300
	Ours (Activity-Text)	8.4	22.2	32.3	30.3	229.9	4.6	15.3	22.7	71	303.7
	Ours (Place-Text)	7.1	19.8	28.7	38	275.1	4.3	14	21.1	77	309.6
1.3	CON(Object, Activity)-Text	9.1	24.6	36	23	181.4	5.5	17.6	25.9	51	243.4
	CON(Object, Activity, Audio)-Text	9.3	27.8	38	22	162.3	5.7	18.4	26.8	48	242.5
1.4	Joint Image-Text-Audio Embedding	8.7	22.4	32.1	31	225.8	4.8	15.3	22.9	73	313.6
1.5	Fusion [Object-Text, Activity (I3D)-Text]	12.3	31.3	42.9	16	145.4	6.8	20.7	29.5	39	224.7
	Fusion [Object-Text, Activity(I3d-Audio)-Text]	12.5	32.1	42.4	16	134	7	20.9	29.7	38	213.8
	Fusion [Object-Text, Place-Text]	11.8	30.1	40.8	18	172.1	6.5	19.9	28.5	43	234.1
	Fusion [Activity-Text, Place-Text]	11	28.4	39.3	20	152.1	5.9	18.6	27.4	44	224.7
1.6	**Fusion [Object-Text, Activity-Text, Place-Text]**	**13.8**	**33.5**	**44.3**	**14**	**119.2**	**7.3**	**21.7**	**30.9**	**34**	**196.1**
1.7	Rank Fusion [Object-Text, Activity-Text, Place-Text]	12.2	31.6	42.7	16	127.6	6.8	20.5	29.4	38	204.3

Table 9.1: Video-to-Text and Text-to-Video Retrieval Results on MSR-VTT Dataset.

query video, we calculate its similarity scores with all the sentences in the dataset in three embedding spaces and use a fusion of similarity scores for the final ranking.

$$S_{v-t}(v,t) = w_1 S_{o-t} + w_2 S_{a-t} + w_3 S_{p-t} \tag{9.4}$$

It may be desired to use a weighted sum when it is necessary in a task to put more emphasis on one of the facets of the video (objects or captions or scene). In this work, we empirically found putting comparatively higher importance to S_{o-t} (Object-Text) and S_{a-t} (Activity-Text), and slightly lower importance to S_{p-t} (Place-Text) works better in evaluated datasets than putting equal importance to all. We empirically choose $w_1 = 1$, $w_2 = 1$ and $w_3 = 0.5$ in our experiments based on our evaluation on the validation set.

9.4 Experiments

In this section, we first describe the datasets and evaluation metric (Section 4.1). Then, we describe the training details. Next, we provide quantitative results on MSR-VTT dataset (Section 4.3) and MSVD dataset (Section 4.4) to show the effectiveness of our proposed framework. Finally. we present some qualitative examples analyzing our success and failure cases (Section 4.5).

9.4.1 Datasets and Evaluation Metric

We present experiments on two standard benchmark datasets: Microsoft Research Video to Text (MSR-VTT) Dataset [Xu+16] and Microsoft Video Description dataset (MSVD) [CD11] to evaluate the performance of our proposed framework. We adopt rank-based metric for quantitative performance evaluation.

MSR-VTT. The MSR-VTT is a large-scale video description dataset. This dataset contains 10,000 video clips. The dataset is split into 6513 videos for training, 2990 videos for testing, and 497 videos for the validation set. Each video has 20-sentence descriptions. This is one of the largest video captioning dataset in terms of the number of sentences and the size of the vocabulary.

MSVD. The MSVD dataset contains 1970 Youtube clips, and each video is annotated with about 40 sentences. We use only the English descriptions. For a fair comparison, we used the same splits utilized in prior works [Ven+15], with 1200 videos for training, 100 videos for validation, and 670 videos for testing. The MSVD dataset is also used in [Ota+16] for video-text retrieval task, where they randomly chose 5 ground-truth sentences per video. We use the same setting when we compare with that approach.

Evaluation Metric. We use the standard evaluation criteria used in most prior work on image-text retrieval and video-text retrieval task [Ota+16; KSZ14; DLS16]. We measure rank-based performance by $R@K$, Median Rank($MedR$) and Mean Rank ($MeanR$). $R@K$ (Recall at K) calculates the percentage of test samples for which the correct result is found in the top-K retrieved points to the query sample. We report results for $R@1$, $R@5$ and $R@10$. Median Rank calculates the median of the ground-truth results in the ranking. Similarly, Mean Rank calculates the mean rank of all correct results.

9.4.2 Training Details

We used two Titan Xp GPUs for this work. We implemented the network using PyTorch following [Fag+18]. We start training with a learning rate of 0.002 and keep the learning rate fixed for 15 epochs. Then the learning rate is lowered by a factor of 10 and the training continued for another 15 epochs. We use a batch-size of 128 in all the experiments. The embedding networks are trained using ADAM optimizer [KB14b]. When the L2 norm of the gradients for the entire layer exceeds 2, gradients are clipped. We tried different values for margin α in training and found $0.1 \leq \alpha \leq 0.2$ works reasonably well. We empirically choose α as 0.2. The embedding model was evaluated on the validation set after every epoch. The model with the best sum of recalls on the validation set is chosen as the final model.

9.4.3 Results on MSR-VTT Dataset

We report the result on MSR-VTT dataset [Xu+16] in Table 9.1. We implement several baselines to analyze different components of the proposed approach. To understand the effect of different loss functions, features, the effect of feature concatenation, and the proposed fusion method, we divide the table into 7 rows (1.1-1.7). In row-1.1, we report the results of applying two different variants of pair-wise ranking loss. VSE[KSZ14] is based on the basic triplet ranking loss similar to Eq. 9.1 and VSEPP[Fag+18] is based on the loss function that emphasizes on hard-negatives as shown in Eq. 9.2. Note that, all other reported results in Table 9.1 is based on the modified pairwise ranking loss proposed in Eq. 9.3. In row-1.2, we provide the performance of different features in learning the embedding using the proposed loss. In row-1.3, we present results for the learned embedding utilizing a feature vector that is a direct concatenation of different video features. In row-1.4, we provide the result when a shared representation between image, text, and audio modality is learned using proposed loss following the idea in [AVT17] and used for video-text retrieval task. In row-1.5, we provide the result based on the proposed approach that employs two video-text joint embeddings for retrieval. In row-1.6, we provide the result based on the proposed ensemble approach that employs all three video-text joint embeddings for retrieval. Additionally, in row-1.7, we also provide the result for the case where the rank fusion has been considered in place of the proposed score fusion.

Loss Function. For evaluating the performance of different ranking loss functions in the task, we can compare results reported in row-1.1 and row-1.2. We can choose only results based on Object-Text spaces from these two rows for a fair comparison. We see that VSEPP loss function and proposed loss function perform significantly better than the traditional VSE loss function in $R@1$, $R@5$, $R@10$. However, VSE loss function has better performance in terms of the mean rank. This phenomenon is expected based on the characteristics of the loss functions. As higher $R@1$, $R@5$ and $R@10$ are more desirable for an efficient video-text retrieval system than the mean rank, we see that our proposed loss function performs better than other loss functions in this task. We observe similar performance improvement using our loss function in other video-text spaces too.

Video Features. We can compare the performance of different video features in learning the embedding using the proposed loss from row-1.2. We observe that the object feature and activity feature from the video performs reasonably well in learning a joint video-text space. The performance is very low when only the audio feature is used for learning the embedding. It can be expected that the natural sound associated with a video alone does not contain as much information as videos in most cases. However, utilizing audio along with the i3d feature as activity features provides a slight boost in performance as shown in row-1.3 and row-1.4.

Feature Concatenation for Representing Video. One could contend that instead of de-

Method	R@1	R@5	R@10	MedR	MeanR
Results Using Partition used by JMET and JMDV					
CCA					245.3
JMET					208.5
JMDV					224.1
W2VV-ResNet152	16.3		44.8	14	110.2
VSE (Object-Text)	15.8	30.2	41.4	12	84.8
VSEPP(Object-Text)	21.2	43.4	52.2	9	79.2
Ours(Object-Text)	23.4	45.4	53	8	75.9
Ours(Activity-Text)	21.3	43.7	53.3	9	72.2
Ours(Place-Text)	11.2	25.1	34.3	27	147.7
Ours-Fusion(O-T, P-T)	25.7	45.4	54	7	65.4
Ours-Fusion(A-T, P-T)	26	46.1	55.8	7	53.5
Ours-Fusion(O-T, A-T)	31.5	51	61.5	5	41.7
Ours-Fusion(O-T, A-T, P-T)	**33.3**	**52.5**	**62.5**	**5**	**40.2**
Rank-Fusion(O-T, A-T, P-T)	30	51.3	61.8	5	42.3
Results Using Partition used by LJRV					
ST	2.99	10.9	17.5	77	241
LJRV	9.85	27.1	38.4	19	75.2
W2VV(Object-Text)	17.9	-	49.4	11	57.6
Ours(Object-Text)	20.9	43.7	54.9	7	56.1
Ours(Activity-Text)	17.5	39.6	51.3	10	54.8
Ours(Place-Text)	8.5	23.3	32.7	26	99.3
Ours-Fusion(O-T, A-T)	25.5	51.3	61.9	5	32.5
Ours-Fusion(O-T, A-T, P-T)	**26.4**	**51.9**	**64.5**	**5**	**31.1**
Rank-Fusion(O-T, A-T, P-T)	24.3	49.3	62.4	6	34.6

Table 9.2: We highlight the proposed method. The methods which have 'Ours' keyword in the name are trained with the proposed loss.

veloping numerous video-semantic spaces, a more streamlined approach would be to amalgamate all available video features. This aggregated video feature can then be utilized to learn a singular video-text space.[DLS16; Xu+16]. However, we observe from row-1.3 that integrating complementary features by static concatenation-based fusion strategy fails to utilize the full potential of different video features for the task. Comparing row-1.2 and row-1.3, we observe that a concatenation of the object feature, activity feature, and Audio feature performs even worse than utilizing only the object feature in $R@1$. Although we see some improvement in other evaluation metrics, overall the improvement is very limited. We believe that both the appearance feature and action feature gets suppressed in such concatenation as they focus on representing different entities of a video.

Learning a Shared Space across Image, Text, and Audio.
Learning a shared space across the image, text and sound modality is proposed for cross-

modal retrieval task in [AVT17]. Following the idea, we trained a shared space across video-text-sound modality using the pairwise ranking loss by utilizing video-text and video-sound pairs. The result is reported in row-1.4. We observe that performance in video-text retrieval tasks degrades after training such an aligned representation across 3 modalities. Training such a shared representation gives the flexibility to transfer across multiple modalities. Nevertheless, we believe it is not tailored toward achieving high performance in a specific task. Moreover, aligning across 3 modalities is a more computationally difficult task and requires many more examples to train.

Proposed Fusion. The best result in Table. 9.1 is achieved by our proposed fusion approach as shown in row-1.6. We see that the proposed method achieves 31.43% improvement in $R@1$ for text retrieval and 25.86% improvement for video retrieval in $R@1$ compared to best performing Ours(Object-text) as shown in row-1.2, which is the best among the other methods which use a single embedding space for the retrieval task. In row-1.5, Fusion[Object-text & Activity(I3D-Audio)-text] differs from Fusion[Object-text & Activity(I3D)-text] in the feature used in learning the activity-text space. We see that utilizing audio in learning the embedding improves the result slightly. However, as the retrieval performance of individual audio features is very low (shown in row-1.2), we did not utilize audio-text space separately in fusion as we found it degraded the performance significantly.

Comparing row-1.6, row-1.5, and row-1.2, we find that the ensemble approach with score fusion results in significant improvement in performance, although there is no guarantee that the combination of multiple models will perform better than the individual models in the ensemble in every single case. However, the ensemble average consistently improves performance significantly.

Rank vs Similarity Score in Fusion. We provide the retrieval result based on the weighted rank aggregation of three video-text spaces in row-1.7. Comparing the effect of rank fusion in replacement of the score fusion from row-1.6 and row-1.7 in Table. 9.1, it is also evident that the proposed score fusion approach shows consistent performance improvement over the rank fusion approach. It is possible that exploiting similarity scores to combine multiple shreds of evidence may be less effective than using rank values in some cases, as the score fusion approach independently weights scores and does not consider overall performance in weighting [Lee97]. However, we empirically find that utilizing score fusion is more advantageous than rank fusion in our system in terms of retrieval effectiveness.

9.4.4 Results on MSVD Dataset

We report the results of video to text retrieval task on MSVD dataset [CD11] in Table 9.2 and the results for text-to-video retrieval in Table 9.3.

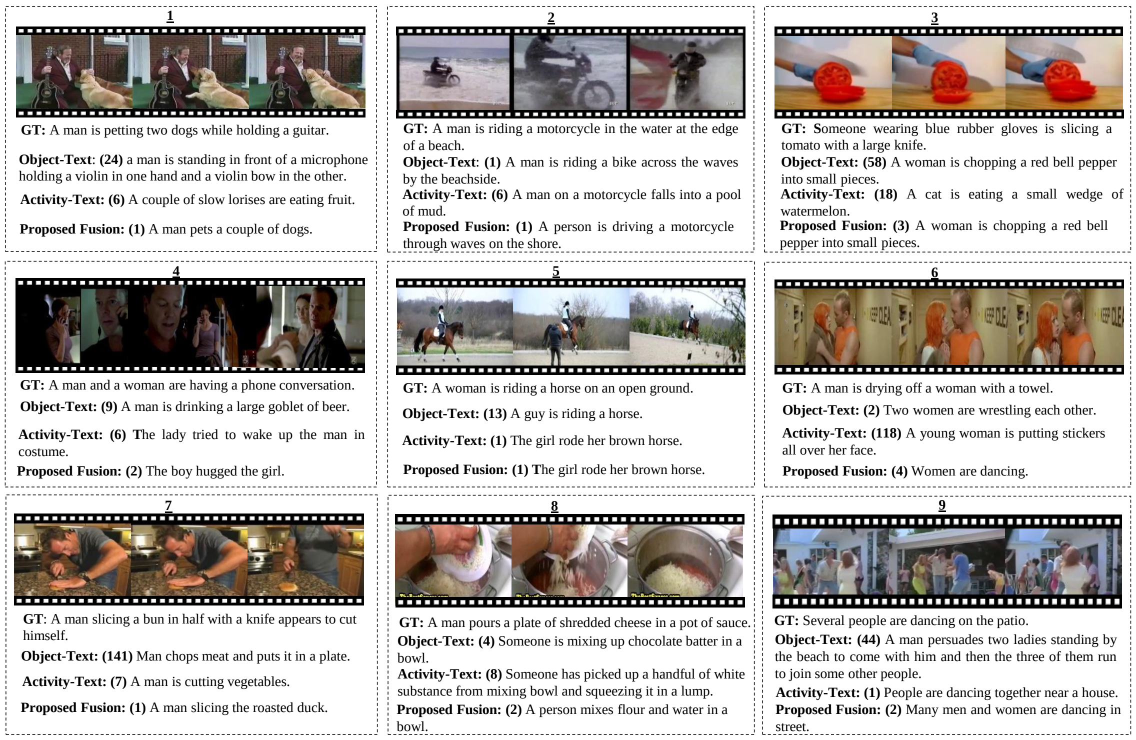
We compare our approach with existing video-text retrieval approaches, CCA[SFF10], ST [Kir+15], JMDV [Xu+15], LJRV [Ota+16], JMET [Pan+16a], and W2VV [DLS16]. For

Method	R@1	R@5	R@10	MedR	MeanR
Results Using Partition used by JMET and JMDV					
CCA					251.3
JMDV					236.3
VSE(Object-Text)	12.3	30.1	42.3	14	57.7
VSEPP(Object-Text)	15.4	39.6	53	9	43.8
Ours(Object-Text)	16.1	41.1	53.5	9	42.7
Ours(Activity-Text)	15.4	39.2	51.4	10	43.2
Ours(Place-Text)	7.9	24.5	36	21	64.6
Ours-Fusion(O-T, P-T)	17	42.2	56	8	36.5
Ours-Fusion(A-T, P-T)	17.2	42.6	55.6	8	34.1
Ours-Fusion(O-T, A-T)	20.3	47.8	61.1	6	28.3
Ours-Fusion(O-T, A-T, P-T)	**21.3**	**48.5**	**61.6**	**6**	**26.3**
Rank-Fusion(O-T, A-T, P-T)	19.4	45.8	59.4	7	29.2
Results Using Partition used by LJRV					
ST	2.6	11.6	19.3	51	106
LJRV	7.7	23.4	35	21	49.1
Ours(Object-Text)	15	40.2	51.9	9	45.3
Ours(Activity-Text)	14.6	38.9	51	10	45.1
Ours(Place-Text)	7.9	24.5	36	21	64.6
Ours-Fusion(O-T, A-T)	20.2	47.5	60.7	6	29
Ours-Fusion(O-T, A-T, P-T)	**20.7**	**47.8**	**61.9**	**6**	**26.8**
Rank-Fusion(O-T, A-T, P-T)	18.5	44.9	58.8	7	30.2

Table 9.3: We highlight the proposed method.

these approaches, we directly cite scores from respective papers when available. We report the score for JMET from [DLS16]. The score of CCA is reported from [Xu+15] and the score of ST is reported from [Ota+16]. If scores for multiple models are reported, we select the score of the best-performing method from the paper.

We also implement and compare results with state-of-the-art image-embedding approach VSE[KSZ14] and VSEPP[Fag+18] in the Object-Text(O-T) embedding space. Additionally, to show the impact of only using the proposed loss in retrieval, we also report results based on the Activity-Text(A-T) space and Place-Text(P-T) space in the tables. Our proposed fusion is named as Ours-Fusion(O-T,A-T,P-T) in the Table. 9.2 and Table. 9.3. The proposed fusion system utilizes the proposed loss and employs three video-text embedding spaces for calculating the similarity between video and text. As the audio is muted in this dataset, we train the Activity-Text space utilizing only I3D feature from videos. We also report results for our fusion approach using any two of the three video-text spaces in the tables. Additionally, we report the results of Rank-Fusion(O-T, A-T, P-T), which uses rank in place of similarity score in combining retrieval results of three video-text spaces in the



1

GT: A man is petting two dogs while holding a guitar.

Object-Text: **(24)** a man is standing in front of a microphone holding a violin in one hand and a violin bow in the other.

Activity-Text: (6) A couple of slow lorises are eating fruit.

Proposed Fusion: (1) A man pets a couple of dogs.

2

GT: A man is riding a motorcycle in the water at the edge of a beach.

Object-Text: **(1)** A man is riding a bike across the waves by the beachside.

Activity-Text: (6) A man on a motorcycle falls into a pool of mud.

Proposed Fusion: (1) A person is driving a motorcycle through waves on the shore.

3

GT: Someone wearing blue rubber gloves is slicing a tomato with a large knife.

Object-Text: (58) A woman is chopping a red bell pepper into small pieces.

Activity-Text: (18) A cat is eating a small wedge of watermelon.

Proposed Fusion: (3) A woman is chopping a red bell pepper into small pieces.

4

GT: A man and a woman are having a phone conversation.

Object-Text: (9) A man is drinking a large goblet of beer.

Activity-Text: (6) The lady tried to wake up the man in costume.

Proposed Fusion: (2) The boy hugged the girl.

5

GT: A woman is riding a horse on an open ground.

Object-Text: (13) A guy is riding a horse.

Activity-Text: (1) The girl rode her brown horse.

Proposed Fusion: (1) The girl rode her brown horse.

6

GT: A man is drying off a woman with a towel.

Object-Text: (2) Two women are wrestling each other.

Activity-Text: (118) A young woman is putting stickers all over her face.

Proposed Fusion: (4) Women are dancing.

7

GT: A man slicing a bun in half with a knife appears to cut himself.

Object-Text: (141) Man chops meat and puts it in a plate.

Activity-Text: (7) A man is cutting vegetables.

Proposed Fusion: (1) A man slicing the roasted duck.

8

GT: A man pours a plate of shredded cheese in a pot of sauce.

Object-Text: (4) Someone is mixing up chocolate batter in a bowl.

Activity-Text: (8) Someone has picked up a handful of white substance from mixing bowl and squeezing it in a lump.

Proposed Fusion: (2) A person mixes flour and water in a bowl.

9

GT: Several people are dancing on the patio.

Object-Text: (44) A man persuades two ladies standing by the beach to come with him and then the three of them run to join some other people.

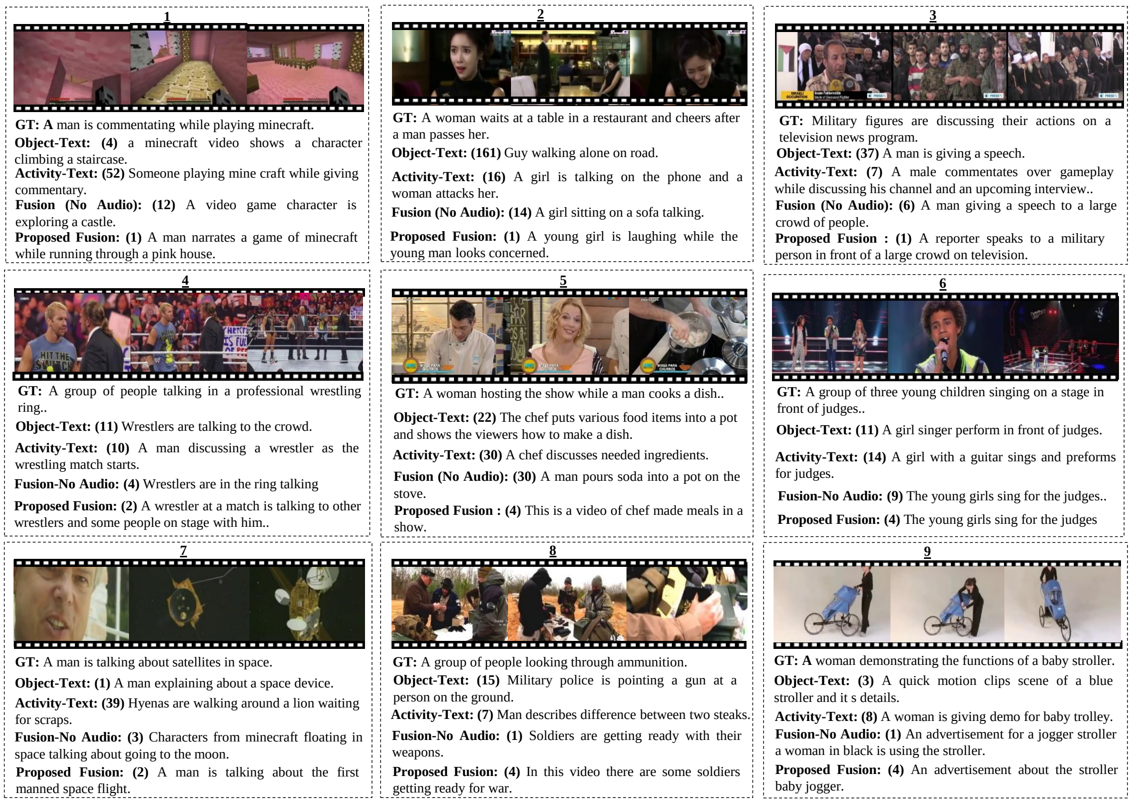
Activity-Text: (1) People are dancing together near a house.

Proposed Fusion: (2) Many men and women are dancing in street.

Figure 9.5: The value in brackets is the rank of the highest-ranked ground-truth caption. Ground Truth (GT) is a sample from the ground-truth captions. Among all the approaches, object-text (ResNet152 as video feature) and activity-text (I3D as video feature) are systems where a single video-text space is used for retrieval. We also report results for the fusion system where three video-text spaces (object-text, activity-text, and place-text) are used for retrieval.

fusion system.

From Table 9.2 and Table 9.3, it is evident that our proposed approach performs significantly better than existing ones. The result is improved significantly by utilizing the fusion proposed in this chapter that utilizes multiple video-text spaces in calculating the final ranking. Moreover, utilizing the proposed loss improves the result over previous state-of-the-art methods. It can also be identified that our loss function is not only useful for learning embedding independently, but also it is useful for the proposed fusion. We observe that utilizing the proposed loss function improves the result over previous state-of-the-art methods consistently, with a minimum improvement of 10.38% from the best existing method VSEPP(Object-Text) in Video-to-Text Retrieval and 4.55% in Text-to-Video Retrieval. The result is improved further by utilizing the proposed fusion framework in this chapter that utilizes multiple video-text spaces in an ensemble fusion approach in calculating the final ranking, with an improvement of 57.07% from the best existing method



1

GT: A man is commentating while playing minecraft.
Object-Text: (4) a minecraft video shows a character climbing a staircase.
Activity-Text: (52) Someone playing mine craft while giving commentary.
Fusion (No Audio): (12) A video game character is exploring a castle.
Proposed Fusion: (1) A man narrates a game of minecraft while running through a pink house.

2

GT: A woman waits at a table in a restaurant and cheers after a man passes her.
Object-Text: (161) Guy walking alone on road.
Activity-Text: (16) A girl is talking on the phone and a woman attacks her.
Fusion (No Audio): (14) A girl sitting on a sofa talking.
Proposed Fusion: (1) A young girl is laughing while the young man looks concerned.

3

GT: Military figures are discussing their actions on a television news program.
Object-Text: (37) A man is giving a speech.
Activity-Text: (7) A male commentates over gameplay while discussing his channel and an upcoming interview..
Fusion (No Audio): (6) A man giving a speech to a large crowd of people.
Proposed Fusion : (1) A reporter speaks to a military person in front of a large crowd on television.

4

GT: A group of people talking in a professional wrestling ring..
Object-Text: (11) Wrestlers are talking to the crowd.
Activity-Text: (10) A man discussing a wrestler as the wrestling match starts.
Fusion-No Audio: (4) Wrestlers are in the ring talking
Proposed Fusion: (2) A wrestler at a match is talking to other wrestlers and some people on stage with him..

5

GT: A woman hosting the show while a man cooks a dish..
Object-Text: (22) The chef puts various food items into a pot and shows the viewers how to make a dish.
Activity-Text: (30) A chef discusses needed ingredients.
Fusion (No Audio): (30) A man pours soda into a pot on the stove.
Proposed Fusion : (4) This is a video of chef made meals in a show.

6

GT: A group of three young children singing on a stage in front of judges..
Object-Text: (11) A girl singer perform in front of judges.
Activity-Text: (14) A girl with a guitar sings and preforms for judges.
Fusion-No Audio: (9) The young girls sing for the judges..
Proposed Fusion: (4) The young girls sing for the judges

7

GT: A man is talking about satellites in space.
Object-Text: (1) A man explaining about a space device.
Activity-Text: (39) Hyenas are walking around a lion waiting for scraps.
Fusion-No Audio: (3) Characters from minecraft floating in space talking about going to the moon.
Proposed Fusion: (2) A man is talking about the first manned space flight.

8

GT: A group of people looking through ammunition.
Object-Text: (15) Military police is pointing a gun at a person on the ground.
Activity-Text: (7) Man describes difference between two steaks.
Fusion-No Audio: (1) Soldiers are getting ready with their weapons.
Proposed Fusion: (4) In this video there are some soldiers getting ready for war.

9

GT: A woman demonstrating the functions of a baby stroller.
Object-Text: (3) A quick motion clips scene of a blue stroller and it s details.
Activity-Text: (8) A woman is giving demo for baby trolley.
Fusion-No Audio: (1) An advertisement for a jogger stroller a woman in black is using the stroller.
Proposed Fusion: (4) An advertisement about the stroller baby jogger.

Figure 9.6: The top 1 retrieved captions for four approaches based on the proposed loss function and the rank of the highest ranked ground-truth caption inside the bracket. Among the approaches, Object-Text is trained using ResNet feature as the video feature and Activity-Text is trained using the concatenated I3D feature and Audio feature as the video feature. We also report results for fusion approaches where three video-text spaces are used for retrieval. The fusion approaches use an object-text space trained with ResNet feature and place-text space trained with ResNet50(Place) feature, while in the proposed fusion, the activity-text space is trained using concatenated I3D and Audio feature. Fusion (No Audio) utilizes activity-text space trained with only the I3D feature.

in the video-to-text retrieval and 38.31% in the text to video retrieval. Among the video-text spaces, object-text and activity-text spaces show better performance in retrieval, compared to place-text space which indicates that the annotators focused more on object and activity aspects in annotating the videos. Similar to the results of MSR-VTT dataset, we observe that the proposed score fusion approach consistently shows superior performance than the rank fusion approach in both video-to-text and text-to-video retrieval.

9.4.5 Qualitative Results

We report the qualitative results on MSVD dataset in Fig. 9.5 and the results on MSR-VTT dataset in Fig. 9.6.

MSVD Dataset. In Fig. 9.5, we show examples of a few test videos from MSVD dataset and the top 1 retrieved captions for the proposed approach. We also show the retrieval result when only one of the embeddings is used for retrieval. Additionally, we report the rank of the highest-ranked ground-truth caption in the figure. We can observe from the figure that in most cases, utilizing cues from multiple video-text spaces helps to match the correct caption. We see from Fig. 9.5 that, among 9 videos, the retrieval performance is improved or higher recall is retained for 7 videos. Video-6 and video-9 show two failure cases, where utilizing multiple video-text spaces degrades the performance slightly than using object-text in Video-6 and activity-text space in Video-9.

MSR-VTT Dataset. Similar to Fig. 9.5, we also show qualitative results for a few test videos from MSR-VTT dataset in Fig. 9.6. Video 1-6 in Fig. 9.6 shows a few examples where utilizing cues from multiple video-text spaces helps to match the correct caption compared to using only one of the video-text spaces. Moreover, we also see the result was improved after utilizing audio in learning the second video-text space (Activity-text space). We observe this improvement for most of the videos, as we also observe from Table. 9.1. Video 7-9 shows some failure cases for our fusion approach in Fig. 9.6. Video 7 shows a case, where utilizing multiple video-text spaces for retrieval degrades the performance slightly compared to utilizing only one of the video-text spaces. For Video-8 and video-9 in Fig. 9.6, we observe that the performance improves after fusion overall, but the performance is better when the audio is not used in learning video-text space. On the other hand, video 1-6 includes cases where utilizing audio helped to improve the result.

9.4.6 Discussion

The experimental results are aligned with our rationale that utilizing multiple characteristics of a video is crucial for developing an efficient video-text retrieval system. Experiments also demonstrate that our proposed ranking loss function is effective in learning video-text embeddings better than existing ones. However, we observe that major improvement in experimental performance comes from our mixture of experts system which utilizes evidence from three complementary video-text spaces for retrieval. Our mixture of expert video-text models may not outperform the performance of a single video-text model in the ensemble in every single case, but it is evident from experiments that our system significantly reduces the overall risk of making a particularly poor decision.

From qualitative results, we observe it cannot be claimed in general that one video feature is consistently better than others for the task of video-text retrieval. It can be easily identified from the top-1 retrieved captions in Fig. 9.5 and Fig. 9.6 that the video-text embedding (Object-Text) learned utilizing object appearance feature (ResNet) as video feature is significantly different from the joint embedding (Activity-Text) learned using Activity feature (I3D) as video feature. The variation between the rank of the highest matching caption further strengthens this observation. Object-text space performs better than the activity-text space in retrieval for some videos. For other videos, the activity-text space achieves higher performance. However, it can be claimed that combining knowledge from multiple video-text embedding spaces consistently shows better performance than utilizing only one of them.

We observe from Fig. 9.6 that using audio is crucial in many cases where there is a deep semantic relation between visual content and audio (e.g., the audio is from the third person narration of the video, the audio is music or song) and it gives important cues in reducing description ambiguity (e.g., video-2, video-5, and video-6 in Fig. 9.6). We observe that the performance degrades in some cases when audio is utilized in the system (e.g., video-8 in Fig. 9.6). We see an overall improvement in the quantitative result (Table 9.1) which also supports our idea of using audio. Since we did not exploit the structure of the audio and analyze the structural alignment between audio and video, it is difficult to determine whether audio is always helpful. For instance, audio can come from different things (persons, animals, or objects) in a video, and it might shift our attention away from the main subject. Moreover, the captions are provided mostly based on visual aspects, which makes audio information very sparse. Hence, the overall improvement using audio was limited.

9.5 Conclusions

In this chapter, we study how to leverage diverse video features effectively for developing a robust cross-modal video-text retrieval system. Our proposed framework learns three expert video-text embedding models focusing on three salient video cues (i.e., object, activity, place) and uses a combination of these models for high-quality prediction. A modified pair-wise ranking loss function is also proposed for better learning the joint embeddings, which focuses on hard negatives and applies a weight-based penalty based on the relative ranking of the correct match. Extensive quantitative and qualitative evaluations of MSVD and MSR-VTT datasets demonstrate that our framework performs significantly better than baselines and state-of-the-art systems. This system exemplifies a typical methodology for robustly integrating information from various modalities and serves as a foundational element for subsequent contrastive learning approaches. Consistent with the robustness demonstrated in CLIP[Rad+19], this underlying concept of semi-hard negative mining is one of the key components of improving robustness at scale.

Chapter 10

Understand by Breaking Multimodal Fusion

10.1 Introduction

Increasingly, audio/visual event detection (AED) is gaining importance as its application in content filtering [Hua+18] and multimedia surveillance is soaring. These days, uploading a clip of video or audio to social media platforms such as Facebook or Youtube would involve going through its internal algorithmic content filtering mechanism in case of potential policy-violating content. This requires such systems to be both accurate and robust in order to sniff out(classify) malicious content, which could potentially reduce the need for human intervention.

From a multimodal learning perspective, the AED task is a perfect angle to understand the interactions between the audio and video modalities without worrying about the dependencies or senses in spoken language, as there is no language model involved [Wan18]. In this chapter, we base our experiments on the task of audio tagging, which aims to characterize the acoustic event of an audio stream by selecting a semantic label for it, and we perform analysis on the largest available dataset: Google AudioSet [Gem+17b]. The benchmark on the AudioSet first introduced by [Her+17b] has been constantly refreshing and growing from a mean average precision (mAP) 31% to 43.1% as reported by [Kon+19a; Wan18; WLM19; Kon+18].

Downstream applications of such AED models on home-security cameras detecting events such as glass-breaks, dog-barks or screaming have a strict requirement for both accuracy and robustness to achieve practicality. Despite the sheer volume of existing works on improving the benchmark of the AED task [MHV16a; Kon+19a] from the audio/visual research community, the reasoning for accuracy improvements and robustness of the models is still lacking. Most existing works on understanding audio classification either performed post-hoc analysis or concluded some intuitions from observations [Li+17; Kon+18; Alw+19]. Motivated by this, this work aim to explain neural networks' behavior on audio/visual data from an *adversarial perspective* to dissect the multi-modal learning process.

We follow the framework: Re-implement/improve SOTA models → Generating adversaries → Identify what breaks the model → Point out what makes the networks more robust → Understand how to distinguish robust features/non-robust features.

10.2 Background and Related Works

Audio/Visual Event Classification: In Chapter 6, we saw AED audio-only model's SOTA performance has leaped forward in the past few years, and other work such as [WTF19] has incorporated visual modalities to further boost performance. In this chapter, we mainly base our analysis on a Convolutional Self-attention Network (CSN) and provide a comprehensive comparison between our model with CRNN and ResNet on the AED task from the adversarial perspective.

Adversarial Examples: Fully explaining behaviors of the non-convex non-linear neural networks is notoriously difficult if not a mission impossible. We do not believe conventional wisdom could provide a "Panacea" at the moment. Inspired by [Ily+19] which demonstrated that adversarial examples are actually features in the vision domain, we envision that computing adversarial examples might help us identify what features different neural networks are learning from the audio-visual data, and whether they learn robust features. Adversarial examples were widely recognized to be a security concern for machine learning, and they are recently demonstrated to be equally effective in the audio domain [LSK19], in the meantime, there were attempts to study them in audio tasks [SBS19; Du+20]. We are not only focusing on the security aspect of adversarial examples in this chapter, we also believe by going in the opposite direction of gradient descent (adversarial attack) we could potentially gain more insights.

The main contributions of this chapter can be summarized as:

1. Leveraging audio-only adversarial examples, we showcase what are the robust features in frequency and time domain;

2. We provide a detailed analysis of the proposed multi-modal architecture through unimodal (audio-only) adversarial examples, and address the trade-off between early, middle, and late fusion on adversarial robustness;

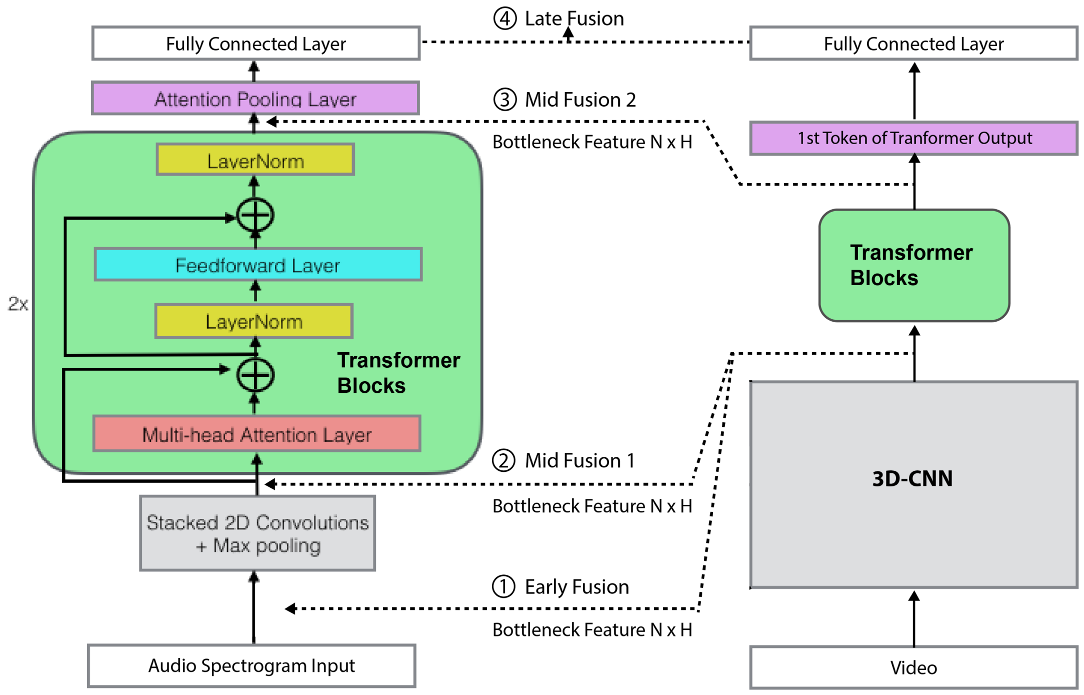


Figure 10.1: The overall architecture of Convolutional Self-attention Network and the different multimodal fusion strategies studied.

10.2.1 Audio Encoding Network

We first describe the audio encoding part of CSN. Following TALNet architecture proposed by [Wan18], we employ stacked convolutions and pooling layers to first extract high-dimensional features from raw frames. In particular, the audio encoder consists of 10 convolution layers of 3x3, and 5 pooling layers are inserted after every 2 convolution layers, which reduces the frame rate from 40 Hz to 10 Hz. The outputs of the convolution encoder are fed into 2 transformer blocks to further model the global interaction among frames. Each transformer block consists of 1 layer of multi-head scaled dot-product attention and 2 feed-forward layers which are connected by residual connections. Finally, we use attention pooling [Wan18] to make final predictions. The audio encoder of CSN is depicted in the left branch of Fig 10.1.

10.2.2 Video Encoding Network (3D-CNN)

For encoding and representing videos, our model employs the R(2+1)D block [Tra+19] which decomposes the 3D (spatial-temporal) CNN into a spatial 2D convolution followed by a temporal 1D convolution. Specifically, let h, w, k denote the height and width of k frames in a sliding window. For a video clip with M frames, there are $N = \lceil M/k \rceil$ windows in it, and the size of each window is denoted as $[k, w, h]$. For each k frame window, our 3D CNN employs a sequence of $[1, w, h]$ 2D convolutional kernel followed by a $[k, 1, 1]$ temporal convolutional kernel to encode the video clip into a size H vector. For the windows in the video, we utilize a 1D temporal CNN and a linear layer to extend its temporal receptive field and map the visual context for multi-modal fusion with the audio model. Essentially, the 3D CNN encodes the video into $\mathbf{v} \in \mathcal{R}^{H \times N}$, where H is the embedding size for multimodal fusion.

10.2.3 Multimodal Fusion

In order to understand the trade-offs between early, middle, and late fusion, we performed 4 different types of fusions in our study. As is shown on the right side of Fig. 10.1, we analyze the effects of fusing video bottle-neck features with the audio pipeline at different stages. Specifically, we investigated late fusion, two levels of middle fusion, and early fusion. For late fusion (maximum model parameters), we first train an audio-only model and a video-only model which contains stacked transformer blocks and a pooling layer on top of the 3D-CNN, as shown on the right side of Fig 10.1. Then we aggregate the output predictions from the full audio and visual models. For early fusion (minimum model parameters), we concatenate the visual bottleneck feature with the spectrogram. In middle fusion 1, we concatenate the visual bottleneck feature with the audio bottleneck features before the transformer blocks. In middle fusion 2, we aggregate the output from the audio and video transformer blocks.

10.3 Adversarial Perturbations

10.3.1 Projected Gradient Descent

Originally, single-modal AED models can be formulated as a minimization of:
$\mathbf{E}_{x,y \sim D}[\mathcal{L}(f(x), y)]$ where $\mathcal{L}$ is the loss function, f is the classifier mapping from input x to label y, and D is the data distribution. We evaluate the quality of our classifier based on the loss, and a smaller loss usually indicates a better classifier. In this chapter, since we are using adversarial noises to deactivate the models, we form $\max_{\delta}[\mathbf{E}_{x,y \sim D}[\mathcal{L}(f(x'), y)]]$, where $x' = x + \delta$ is our perturbed audio input. It is common to define a perturbation set $C(x)$ that constrains x', such that the maximization problem becomes $\max_{x' \in C(x)}[\mathbf{E}_{x,y \sim D}[\mathcal{L}(f(x'), y)]]$. The set $C(x)$ is usually defined as a ball of small radius of the perturbation size ϵ (of either ℓ_∞, ℓ_2 or ℓ_1) around x.

To solve such a constrained optimization problem, one of the most common methods utilized to circumvent the non-exact-solution issue is the Projected Gradient Descent (PGD) method:

$$\delta = \mathcal{P}_\epsilon \left(\delta - \alpha \frac{\nabla_\delta \mathcal{L}(f(x+\delta), y)}{\|\nabla_\delta \mathcal{L}(f(x+\delta), y)\|_p} \right) \tag{10.1}$$

where $\mathcal{P}_\epsilon$ is the projection onto the ℓ_p ball of radius ϵ, and α is the gradient step size. To practically implement gradient descent methods, we use a very small step size and iterate it by ϵ/α steps.

In this chapter, we are particularly interested in measuring the attacking potency through the benchmarks including ϵ in the ℓ_p norm ball, and step size α. Since our main focus is to maximally exploit the model's universal weakness, we only discuss untargeted attacks that drift the model's prediction away from the true label y. We do not discuss targeted attacks where we have to minimize the loss towards y_{target} in the meantime, since doing this will inevitably further constrain our optimization and make our attack less potent. Note that our δ is a universal perturbation trained on the entire training set, and it is tested against the entire evaluation set to measure its attack success rate.

10.3.2 Multi-Modal Adversarial Perturbations

In a multi-modal setting, the loss formulation is $\mathcal{L}_{multi} = \mathcal{L}(f(x_{m_1} \oplus x_{m_2} \oplus \cdots \oplus x_{m_k}), y)$, where x_{m_k} denotes inputs from multiple modalities. In this chapter, we are dealing with 2 modalities: x_{audio} and x_{video} We study the adversarial perturbation computed against the audio input δ_{audio}. Our optimization goal is: $\max\limits_{\delta_{audio}}[\mathbf{E}_{x,y\sim D}[\mathcal{L}(f((x_{audio}+\delta_{audio})\oplus x_{video}), y)]]$, and thus, our PGD step is:

$$\begin{aligned}\delta_{audio} &= \\ \mathcal{P}_\epsilon &\left(\delta_{audio} - \alpha \frac{\nabla_{\delta_{audio}} \mathcal{L}(f((x_{audio}+\delta_{audio}) \oplus x_{video}), y)}{\|\nabla_{\delta_{audio}} \mathcal{L}(f((x_{audio}+\delta_{audio}) \oplus x_{video}), y)\|_p} \right)\end{aligned} \tag{10.2}$$

Since we focus more on the robustness of the audio modality in this chapter, we expand the above definition to incorporate δ_{video} in Chapter 11, readers can refer to Section 11.3.1 for the full definition of multimodal adversarial perturbation.

10.4 Experiments

10.4.1 Dataset

We use the AudioSet train and test setup as described in Chapter 6. The input for the audio branch are matrices of Mel filter bank features, which have 400 frames(time domain) and 64

Mel-filter (frequency domain) bins. For the visual branch, we employ a 3D CNN backbone to encode the spatial-temporal context in videos as is mentioned in Section 10.2.2.

10.4.2 Training Details

3D-CNN: Our 3D CNN video encoding backbone is initialized with the network pre-trained on IG65M [GTM19]. For training on the videos in Google Audio Set, we freeze the clip-level 3D-CNN backbone and fine-tune the video-level 1D CNN and the transformer layer.
CSN: We used a batch size of 300 and train the model for 50K steps. The model is optimized with Adam with a learning rate of 4e-4. We used a dropout rate of 0.75 for transformer layers and we found this critical to achieve the best performance. Also, we did not use positional encoding as we found that the model would yield better performance without it.
CRNN and ResNet: We reimplemented CRNN according to TALNet [Wan18], and ResNet following the same setup as [For+19] as baselines for comparisons in Section 10.5.3.

10.5 Results and Discussion

10.5.1 Adversarial Examples against Fusion in different stages

As we can see from Table 10.1, late fusion prevails in terms of performance with and without the presence of the adversarial noise computed in the same way (full-frequency and time domain, the same size as input). However, late fusion is the most expensive model as it requires 2 full-sized branches of audio and visual models with a huge amount of parameters. Evidently, we can observe the trend that the deeper level of the fusion is, the more robust the model would be against the adversary.

10.5.2 Constrained Temporal/Frequency Adversarial Examples

SpecAugment [Par+19] introduced a simple but effective technique for data augmentation in speech recognition task, where they mask out certain frequency band or time steps in the original spectrogram input. Inspired by this, we modify the spectrograms to construct adversarial examples. In particular, we distinguish frequency bands that are robust features versus those that are non-robust. Our results also partially explain why [Par+19]'s approach achieved significant performance improvement, that is by masking out non-robust frequency bands, the model is forced to learn more robust features that are going to generalize better in test setting.

To identify the robust/non-robust regions in the audio features, we compute constrained adversarial noises as is shown in Fig. 10.2 (a,b,c). Against the best CSN audio-only model

Models	Attack	**mAP**	**AUC**	**d-prime**
Early Fusion	No	0.41	0.920	2.360
Early Fusion	Yes	0.07	0.811	1.172
Mid Fusion 1	No	0.41	0.931	2.460
Mid Fusion 1	Yes	0.19	0.832	1.268
Mid Fusion 2	No	0.42	0.972	2.704
Mid Fusion 2	Yes	0.20	0.834	1.333
Late Fusion	No	**0.44**	0.969	2.631
Late Fusion	Yes	**0.27**	0.910	1.871

Table 10.1: Performance of our best performing CSN models with different multimodal fusion strategies shown in Fig 10.1, and their performance against the same strength of adversarial perturbation ($\epsilon = 0.3$, ℓ_2 norm). Here, mAP is the mean average precision, AUC is the area under the false positive rate and true positive rate (recall) which reflects the influence of the true negatives. The d-prime can be calculated from AUC [Gem+17b].

(without fusion), we computed different permutations of adversarial perturbations that mask out different frequencies and temporal portions with different attack strengths. The results are shown in Table. 10.2 and Table. 10.3. As we can observe, adversarial noises present on lower frequencies tend to have higher attack potency. This suggests that the lower frequency bands contribute more to the overall performance of the model while the features there are not robust. Attacks on higher frequencies tend to be less effective. Not surprisingly, higher ϵ leads to more effective attacks. Interestingly, using the ℓ_∞ attack, attacks on different frequency lead to similar accuracy drop. This is because ℓ_∞ is too potent as is shown in Fig. 10.2 (d).[1]In the temporal domain, we can observe a nearly uniform drop in performance despite of the different temporal masks, suggesting the temporal domain contributes equally to robustness.

10.5.3 Adversarial Robustness of different Neural Architectures

We also compute the adversarial noise with the same ℓ_2 norm and $\epsilon = 0.3$ to attack different neural network models. We run 3 trials of every model with different seeds to report the average performance. The results are shown in table 10.4[2]. We observe

[1]Due to space constraint, we often discuss adversarial perturbation using ℓ_2 norm instead of ℓ_∞ norm in this chapter. We have performed the equivalent experiments using ℓ_∞ norm, and observed the same trend.

[2]To avoid repeating, refer to Table 10.1 for early/middle fusion result

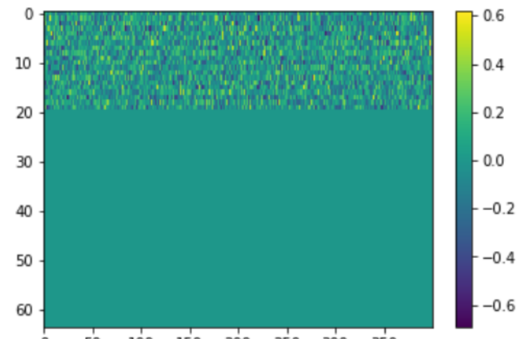
freq mask	ϵ	norm	α	**mAP**	**AUC**	**d-prime**
No	-	-	-	0.392	0.967	2.598
No	0.1	2	0.01	0.262	0.942	2.218
No	0.3	2	0.01	0.104	0.865	1.558
0-20	0.1	2	0.01	**0.192**	0.910	1.900
0-20	0.3	2	0.01	**0.077**	0.827	1.334
20-40	0.1	2	0.01	0.223	0.927	2.061
20-40	0.3	2	0.01	0.084	0.835	1.376
40-64	0.1	2	0.01	0.266	0.942	2.217
40-64	0.3	2	0.01	0.121	0.871	1.602

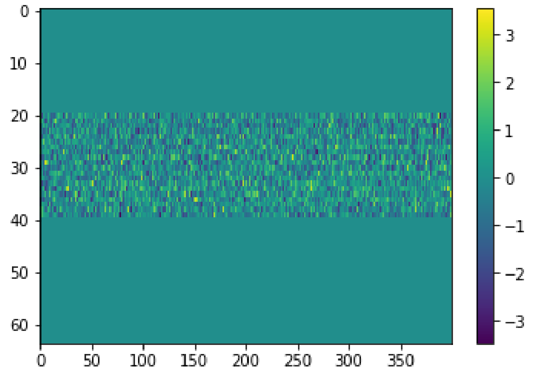
Table 10.2: Performance of our best performing CSN audio-only model under the attack of constrained adversarial perturbations in different *frequency* domains.

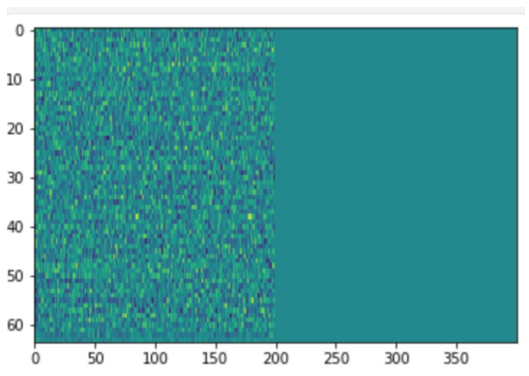
temporal mask	ϵ	norm	α	**mAP**	**AUC**	**d-prime**
No	-	-	-	0.392	0.967	2.598
0-200	0.1	2	0.01	0.274	0.945	2.266
0-200	0.3	2	0.01	0.171	0.902	1.830
200-400	0.1	2	0.01	0.272	0.945	2.266
200-400	0.3	2	0.01	0.169	0.903	1.835

Table 10.3: Performance of our best performing CSN audio-only model under the attack of constrained adversarial perturbations in different *temporal* domains.

that replacing the recurrent layer of CRNN with transformers greatly boosts the audio classification performance, suggesting that self-attention modules could lead to the best accuracy empirically. Under the influence of adversarial noise, our full late fusion model still has the highest performance, whereas ResNet seems to suffer the least in terms of performance drop. This is an interesting yet not fully understood phenomenon in that pure convolutional modules appear to be more robust than recurrent or self-attention layers. We conjecture that the residual links in ResNet could be obfuscating the gradients making them more robust against gradient-based attacks.


(a) Low Freq Noise(0-20)


(b) Higher Freq Noise (20-40)


(c) Temporal Noise (0-200dim)

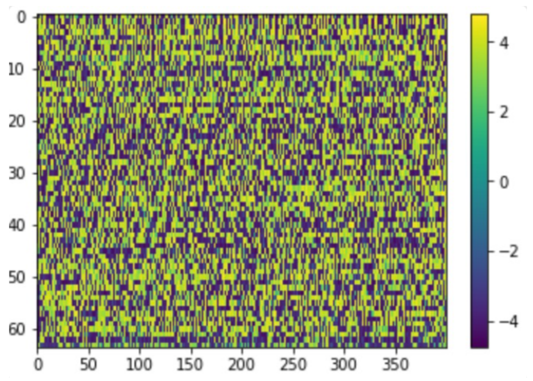

(d) ℓ_∞ Attack $\epsilon = 0.3$

Figure 10.2: Adversarial Noises constrained with different temporal/filter banks masks.

Model	**No attack**			**Attack**		
	mAP	AUC	d-prime	mAP	AUC	d-prime
ResNet(audio)	0.352	0.966	2.602	0.214	0.873	1.412
CRNN(audio)	0.356	0.966	2.572	0.183	0.810	1.123
CSN(audio)	0.392	0.970	2.650	0.221	0.872	1.213
CSN+3DCNN	0.441	0.969	2.631	0.271	0.910	1.871

Table 10.4: Different architectures against the same adversarial noise

Class	**CRNN**		**CSN**	
	mAP	mAP−adv	mAP	mAP−adv
Mosquito	54.1	6.2	61.2	11.4
Whale vocal	45.4	1.8	33.9	0.5
Light engine	37.6	1.2	49.4	0.3
Sewing machine	45.7	1.0	60.3	1.3
Sizzle	58.6	0.6	71.4	2.2
Reverberation	23.4	15.6	16.6	13.9
Static	39.5	0.6	44.2	1.8
Mains hum	37.8	0.4	24.1	0.4

Table 10.5: Class-wise mAP comparison between CRNN and CSN (audio) w/ and w/o attack.
The perturbation we used in this experiment is ℓ_2 with $\epsilon = 0.1$. mAP-adv is the performance under the influence of attack.

10.5.4 Class-wise Performance under attack

As a further step towards understanding what every model is learning and their reaction to the impact of adversarial noises. We further compare the performances of CSN (audio) and CRNN in different classes. Some of the classes that have large discrepancies are shown here. Although CRNN's overall performance is much lower than CSN, it outperforms CSN on several classes such as *Mains hum* by a large margin. From Table. 10.5, we can see that CSN has better adversarial robustness overall compared to CRNN. It is interesting to notice that classes trained with more data (the AudioSet is unbalanced) such as speech and music are relatively robust against attacks as is shown in Table 10.6, and the more well-defined structured sound such as sine wave does not get affected much by adversarial noise. Plus, the higher frequency classes' accuracy tends to drop significantly with adversarial noise.

10.6 Conclusion

In this chapter, we addressed several important yet unexplored questions about the multimodal adversarial robustness of neural networks in audio/visual event classification. Our work is based on the best state-of-the-art multimodal model on the Google AudioSet with 44.1 mAP. Our observations include:

Top 10 classes w/o attack			**Top 10 classes w/ attack**		
Attack	no	yes	**Attack**	yes	no
EmergVehicle	86.9	50.6	Speech	60.5	76.5
ChangeRinging	86.5	0.2	SmokeDetec	55.3	66.5
Crowing	86.3	19.6	Music	54.7	80.5
CivilSiren	85.1	4.3	Plop	54.0	71.6
Siren	84.9	41.6	EmergVehicle	50.6	86.9
Bagpipes	84.8	26.0	Ambulance	48.2	70.0
Music	80.5	54.7	PoliceCar	48.1	58.1
Croak	78.9	13.7	HeartSounds	45.4	59.0
Speech	76.5	60.5	SineWave	41.6	42.2

Table 10.6: Top 10 Class-wise mAP comparison on the entire test dataset w/ and w/o adversarial noise of the best-performing CSN (audio) model. The perturbation we used in this experiment is ℓ_2 with $\epsilon = 0.1$. "yes" indicates the performance under the influence of adversarial noise, and "no" indicates the original performance. Some classes can still stay on the top-10 list with the attack.

1. Later fusion has better adversarial robustness compared to early fusion
2. Self-attention layers offer more accuracy but not more robustness, while convolution layers are more robust
3. Lower-frequency features contribute more to the accuracy compared to high-frequency features, and yet they are not robust against adversarial noises.

Our findings provide some empirical evidence that could suggest better design in balancing the robustness/accuracy trade-off in multimodal machine learning. This Chapter focused on perturbations that only exist in the audio domain, in the next chapter, we will discuss when perturbation comes from multiple modalities.

Chapter 11

Unimodal *versus* Multimodal Robustness, Quantify Robustness

11.1 Introduction

As we have seen in the previous chapters, neural network models could be vulnerable to adversarial attacks, manipulations of the input to a classifier specifically crafted to be inconspicuous to humans, but which cause the classifier to predict incorrectly [MD+17; Car+19]. Concerns about potential adversarial examples have sparked a huge interest in the research community to study how can we train robust models that defend against potential perturbations [Car+19; Tsi+18]. However, building such adversarially robust models is challenging [Tsi+18]. A smaller but still substantial line of work has emerged to show that we could have formal verification of the robustness of neural network models [WK18; JLD19]. However, such methods are subjected to very tight constraints and are notoriously difficult to scale. So far, despite some successful large-scale empirical evaluations [CRK19; Mad+17] on image-only datasets, the robustness of multi-modal learning has not been fully understood. As illustrated in Fig. 11.1, the major challenge to analyzing multi-modal models is the high non-convexity and non-linearity of the decision boundaries in high dimensional latent spaces.

In this chapter, we discuss the robustness of multi-modal neural network models for classification tasks in a large-scale setting. We first measure *point-wise robustness* through the empirical maximum allowable perturbation in ℓ_p norm. Based on *point-wise robustness*, we show there exist counterexamples to the general claim that multi-modal models are more robust compared to the uni-modal models. Due to the limitation of point-wise robustness in terms of scalability and generalizability, class-wise robustness is a necessary and practical complement to tackle large-scale multi-modal robustness problems. Instead of measuring the accuracy drop caused by running universal adversarial perturbation in different magnitudes and ℓ_p norm [MD+17], we define the class-wise metric by using 1) the density of samples within the high-dimensional ball centered at the centroid of each class with a certain ℓ_p radius; 2) the convexity of samples in the high dimensional latent space. We evaluate our metric on the AudioSet [Gem+17b] and Kinetic-Sounds dataset [AZ17a]. The results indicate that multi-modal models are only more robust measured by class-wise metrics for a limited number of classes. We also observe the point-wise robustness of classification results vary greatly depending on the variance of the data with specific class labels.

Inspired by our observations, we propose a density-convexity-based mix-up fusion technique to smooth the decision boundary and add robustness to the fused model. Our proposed mixup could both improve class-wise robustness and point-wise robustness upon our baseline fusion model while increasing the accuracy compared to vanilla fusion methods. We also compare it to traditional adversarial training, where we also see competitive robust accuracy. These advances allow us to address adversarial robustness in large-scale multi-modal settings for the first time.

11.2 Related Work

Previous works such as [JLD19; WK18; WK20] focused on *point-wise* robustness, studying the maximum allowable radius of centered Chebyshev ball: *a ℓ_p ball centered at an input point, within which the output class of a given neural network with remains unchanged, treating the decision boundary of the model as a convex or non-convex polytope.* This formulation certainly provides a safe threshold to defend against adversarial attacks, whereas it involves expensive iterative computations on each anchor point, resulting in huge difficulty to scale [JLD19; WK18], most of them depend on their claims on small-scale datasets like MNIST or CIFAR-10. For large-scale multimodal datasets such as AudioSet, such a verification would hardly be feasible.

Some recent works are trying to study defense methods on a large scale, including adversarial training [Mad+17], randomized smoothing [CRK19], and model ensemble [SRR20]. Most of them measured robustness by point-wise accuracy or attack success rate for specific attack budget ϵ and number of iterations. Nonetheless, these metrics frequently lack the specificity to accurately portray the impact of adversarial perturbations on the classifier's performance. This motivates us to look into both class-wise accuracy changes along with point-wise accuracy changes in order to have a better chance of understanding potential reasons for label change caused by adversarial perturbations.

Audio-visual learning [BAM18] itself is more complicated than image classification, and the current focus still seems to be improving accuracy [WTF20]. The robustness of multimodal classification models involving large-scale video-audio datasets has never been studied rigorously. [WK20] considered the robustness of deep neural networks on videos and experimented on UCF101 dataset [SZS12] which is a relatively small dataset. They measured robustness by the maximum safe radius (point-wise), which computes the minimum distance from the optical flow sequence obtained from a given input to that of an adversarial example in the neighborhood of the input. Built on the previous Chapter 10, this chapter is a more comprehensive study of the robustness of multi-modals models both consisting of video and audio against adversarial perturbation causing changes to both modalities.

11.3 Background

As is covered in the previous Chapter 10, the definition of Unimodal adversarial perturbation can be seen in Section 10.3.1.

11.3.1 Multi-Modal Adversarial Perturbations

Under audio-visual multimodal learning setting, we formulate our loss as:
$L_{multi} = \mathcal{L}(f(g(x_A) \oplus h(x_V)), y)$, over the classification function f, which can however readily be expanded to additional modalities. $g(x_A)$ denotes the encoding of audio features into a bottleneck representation with audio encoding networks (CSN) depicted in Figure 11.2, while $h(x_V)$ similarly represents the (R2+1D)CNN [Tra+18] encoded video representation and $\oplus$ indicates concatenation of features. $\mathcal{D}_A$ and $\mathcal{D}_V$ indicating their individual dataset. This more complicated setting requires us to study the adversarial perturbation computed against both the audio input δ_A and the video input δ_V, and can be written out as:

$$\begin{aligned} &\mathbf{E}_{(x_A,y)\sim\mathcal{D}_{\mathcal{A}};(x_V,y)\sim\mathcal{D}_{\mathcal{V}}} \max_{\delta_A \in C(x_A), \delta_V \in C(x_V)} [\mathcal{L}(f(x'), y)] \\ &\text{subject to } C(x) = \{a \in \mathbb{R} : ||a - x||_p \leq \epsilon\} \end{aligned} \tag{11.1}$$

Here, $x' = g(x_A + \delta_A) \oplus h(x_V + \delta_V)$. The set $C(x)$ is usually defined as a ball of small radius of the perturbation size ϵ (of either ℓ_∞, ℓ_2 or ℓ_1) around x. Thus, to compute uni-modal audio perturbation to attack the multimodal model, our PGD step could be rewritten as:

$$\delta_A := \mathcal{P}_\epsilon\left(\delta_A - \alpha \frac{\nabla_{\delta_A}\mathcal{L}(f(g(x_A + \delta_A) \oplus h(x_V)), y)}{||\nabla_{\delta_A}\mathcal{L}(f(g(x_A + \delta_A) \oplus h(x_V)), y)||_p}\right) \tag{11.2}$$

Accordingly, to compute video perturbation against video classifier $g(x)$, we perform the following PGD step:

$$\delta_V := \mathcal{P}_\epsilon\left(\delta_V - \alpha \frac{\nabla_{\delta_V}\mathcal{L}(f(h(x_V + \delta_V) \oplus g(x_A)), y)}{||\nabla_{\delta_V}\mathcal{L}(f(h(x_V + \delta_V) \oplus g(x_A)), y)||_p}\right) \tag{11.3}$$

In the more complicated multi-modal case, we jointly optimize δ_A and δ_V, where:

$$\delta_A, \delta_V := \mathcal{P}_\epsilon(\delta_{(V,A)} - \alpha \frac{\nabla_{\delta_{(V,A)}} \mathcal{L}(f(h(x_V + \delta_V) \oplus g(x_A + \delta_A)), y)}{\|\nabla_{\delta_{(V,A)}} \mathcal{L}(f(h(x_V + \delta_V) \oplus g(x_A + \delta_A)), y)\|_p}) \tag{11.4}$$

11.4 Challenge Common Assumptions in Multimodal Learning

There is a vague notion that multimodal systems are generally more robust compared to unimodal models [BAM18]: "Having access to multiple modalities that observe the same phenomenon may allow for more robust predictions." From an information retrieval perspective, this statement is theoretically true since the same information was captured twice in different modalities, improving the robustness of multimodal models. However, this is not always true if we rigorously consider how adversarial perturbations affect the neural network model as follows.

Theorem 1. *There exists a sample $x_i \in \mathcal{D}$, and a unimodal sample-wise attack $\exists||\delta_{A,i}||_p \leq \epsilon_A$ or $\exists||\delta_{V,i}||_p \leq \epsilon_V$ that can break a multimodal fusion network $f((x_{V,i} \oplus x_{A,i}), y_i)$, changing its prediction label y_i.*

Here, $\mathcal{D}$ is the dataset, and ϵ_A and ϵ_V are the point-wise robustness threshold for each uni-modal of a sample x_i. Therefore, as a conjecture, a unimodal attack can break a multimodal model, which we empirically verified the existence of such cases in our experiments. The proof of Theorem 1 can be found in the appendix page.

11.5 Metrics for Class-wise robustness

Point-wise robustness is limited in terms of scalability and generalizability. Class-wise robustness metric is a more efficient for a large-scale dataset. Instead of exhaustively running universal adversarial perturbation we define two metrics to capture the main robustness property of each class.

11.5.1 Centroid-based density metric

We calculate the class-wise density of the class's high dimensional ℓ_p norm ball by a function of the number of samples n_c in the class c and the volume of the ℓ_p norm ball. In this chapter, n_c is the number of samples of each class in Audioset.

The centroid of a class $\odot_c$ is the mean of bottleneck features $l = g(x)$ of samples x in the class c: $\odot_c = \frac{\sum_{i=1}^{n_c} l_{i,c}}{n_c}$, where n_c is the number of samples in the class c. In our case, it is $l = g(x_A)$ for audio modality or $l = g(x_V)$ for video modality. For each class, we calculate

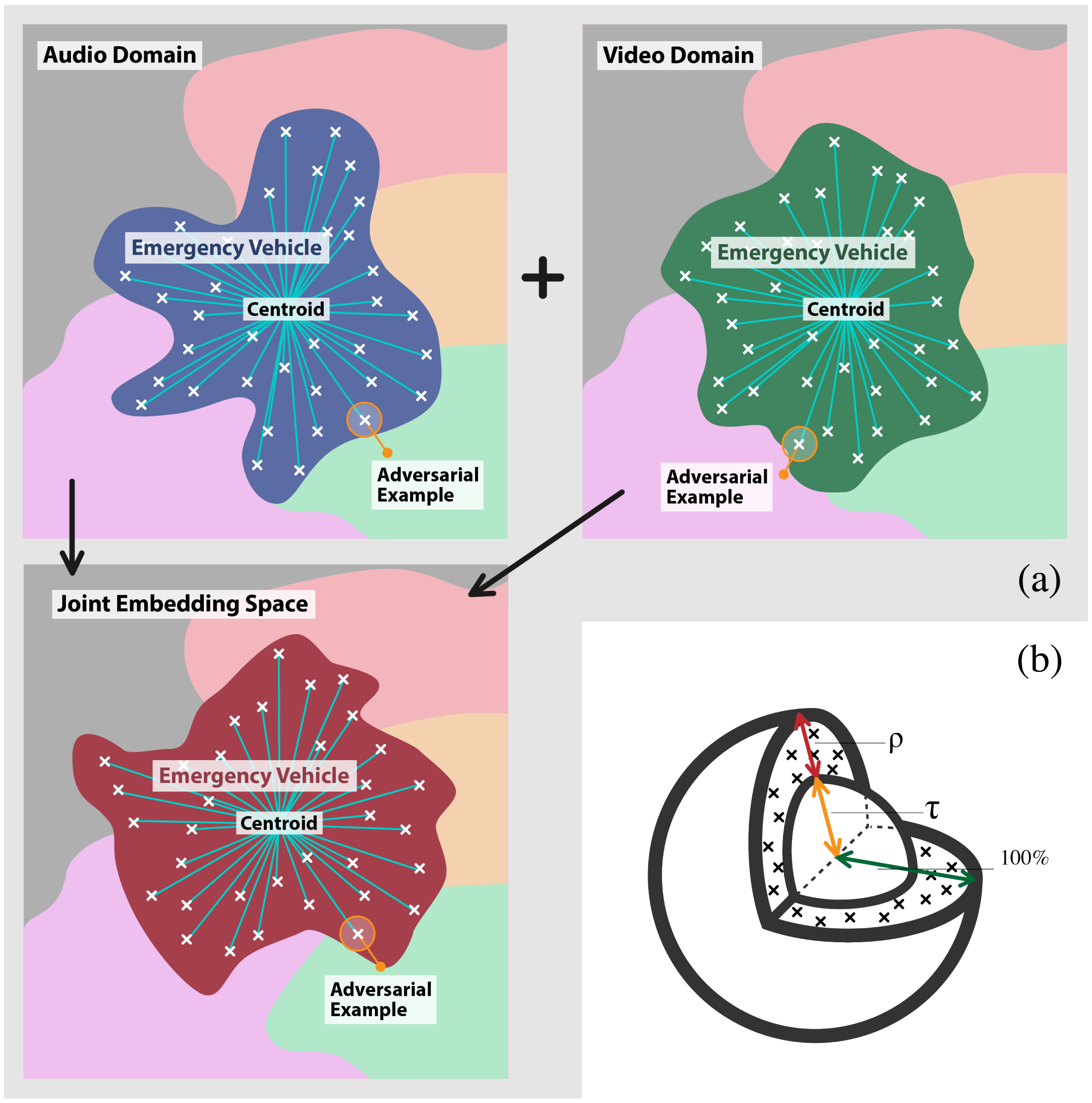


Figure 11.1: (a) An illustration of multi-modal fusion. (b) Illustration of the centroid based density metric $\rho_c^{R_\tau,p}$.

the distance of samples in c to the centroid $\odot_c$. The radius of a class $R_{p,c}$ on ℓ_p norm ball is the maximum distance of all samples in c to the centroid $\odot_c$: $R_{p,c} = \max\|l_{i,c} - \odot_c\|_{p,i=1,\dots,n_c}$. The radius of the first τ percentage of samples closing to the centroid is $R_{\tau,p,c}$, and the $n_{\tau,c} = \tau \times n_c$ is the number of τ percentage samples close to the centroid. According to [Jor], the volume $V_d^p(R)$ of the d-dimensional ℓ_p norm ball with a radius R is: $V_d^p(R) = \frac{(2\Gamma(\frac{1}{p}+1)R)^2}{\Gamma(\frac{d}{p}+1)}$, where, d is the dimension of the ball, R is the radius, p is the ℓ_p-norm, and Γ is the Gamma function[1]. Now, we formally define the robustness of a class c with regard to τ quantile of the class sample x_c's distance to the centroid $\odot_c$ of the class c by:

$$\rho_c^{R_{\tau},p,c} = \frac{n_c - n_{\tau,c}}{log(V_d^p(R_{p,c})) - log(V_d^p(R_{\tau,p,c}))} \tag{11.5}$$

where the numerator is the number of class samples whose ℓ_p distance to centroid larger than τ quantile of samples in c; $R_{\tau,p,c}$ is the τ quantile of all class sample's ℓ_p distance to the class's centroid. We perform the log operation on the volume to reduce the scale of Γ function for ease of computation. This can be intuitively interpreted as the density in the outer crust of a ball as is shown in Fig. 11.1(b). Generally, the higher the density of the crust, the more robust the samples within/below the crust are.

11.5.2 Convexity-based metric

The convex set C in geometry is defined as a set where given any two points $x_1, x_2 \in C$ in the set, the set contains the whole line segment $x = \theta x_1 + (1-\theta)x_2$, with $0 \leq \theta \leq 1$. Based on our observations and conjecture, we propose the convexity-based metric as one of the robustness measurements of the class. We construct the convex set $S = \{\hat{x}_s | \hat{x}_s = \theta x_1 + (1-\theta)x_2, \theta \sim U[0,1], \forall x_1, x_2 \in C\}$, and sample n points from it $\{\hat{x}_1, ..., \hat{x}_n | \hat{x}_i \in S\}$. The metric is defined as follows:

$$\kappa_c = \frac{\sum_{i=1}^{n} 1\{f(\hat{x}_i) = y_c\}}{n} \tag{11.6}$$

where y_c is the class label. The higher the κ_c is, the more convex the decision boundary of class c is. In this work, we set $n = 2000$. Conceptually, the higher number of instances where the midpoint of two points within the same class embedding continues to yield the same prediction from the model, the greater the degree of convexity we attribute to it. Therefore, we use this metric as a proxy to measure how convex the neural network is. We hope to see a positive correlation between convexity and robustness.

For both metrics, we use the bottleneck feature l to calculate the value of the metric. In later experiments, we empirically show the effectiveness of our metrics by contrasting them with the accuracy drop caused by universal perturbation [MD+17].

11.6 Density-Convexity based Mix-up

As is noted by [LLY20], mix-up techniques could potentially smoothen the decision boundary by generating virtual training samples by a weighted sum of existing training samples,

[1]https://wikipedia.org/wiki/Gamma_function

which improves generalizability. Inspired by our density-based and convexity-based metrics, we employ a simple adjustment to mix up:

$$\begin{aligned}\tilde{x}_A &= \alpha x_{Ai} + (1-\alpha)x_{Aj};\\ \tilde{x}_V &= \alpha x_{Vi} + (1-\alpha)x_{Vj};\\ \tilde{y} &= \alpha y_i + (1-\alpha)y_j;\end{aligned} \tag{11.7}$$

where $\alpha \in [0,1]$,(x_{Ai}, x_{Vi}, y_i) and (x_{Aj}, x_{Vj}, y_j) are two training samples, with both audio and video inputs drawn from 2 different classes y_i and y_j, subject to $\kappa_c < T$ (T is an empirical threshold on the convexity) and $\rho_c^{R_\tau,p} < D$ (D is an empirical threshold on the density), for both classes. Both T and D are dataset-dependent parameters, $T = 0.5, D = 8$ in this context. Remember, for a given sample x_i, the standard mixup method [Zha+18] selects x_j from a uniform distribution across the dataset. However, this method might not adequately accommodate the imbalanced distribution of the dataset. In contrast, we are augmenting the less convex classes of training data with more samples from the less dense samples which are farther to the center of its embedding space. Conducting such mixup is effectively adding more “difficulty" to the model during training, implicitly this training shares the same intuition as adversarial training. It is important to note in Figure 11.4, not all classes trace the linear correlation perfectly, and since our motivation is to capture performance across all classes to boost mAP, we do not specifically analyze those edge cases.

11.7 Experiments

11.7.1 Dataset and Model Setup

As is consistently mentioned throughout this thesis, *AudioSet* [Gem+17b] contains 2 Million 10-second YouTube video clips, summing up to 5,800 hours annotated with 527 types of sound events. We train and test the models according to the train (1998999 samples) and test split (20126 samples) described in [Her+17b]. The input for the audio branch are matrices of Mel filter bank features. For the visual branch, we employ an (R2+1D)CNN + transformer backbone [Tra+18] to encode the spatial-temporal context. This network is very similar to what we studied in Chapter 10, with only the video branch and fusion mechanism being slightly different. The clean performance (with no data augmentation) of our unimodal audio model and audio-visual model are listed in Table 11.1 (italic font). *Kinetics-Sounds* [AZ17a] is a subset of Kinetics [Kay+17] that contains 34 classes of audio-related events (22,107 train, 1,504 validation). We preprocess the Kinetic-sound dataset in a similar fashion. Our clean multimodal baseline: *86.5%*. We use Kinetic-sounds only for audio-visual performance since their audio-unimodal performance is low and thus not representative.

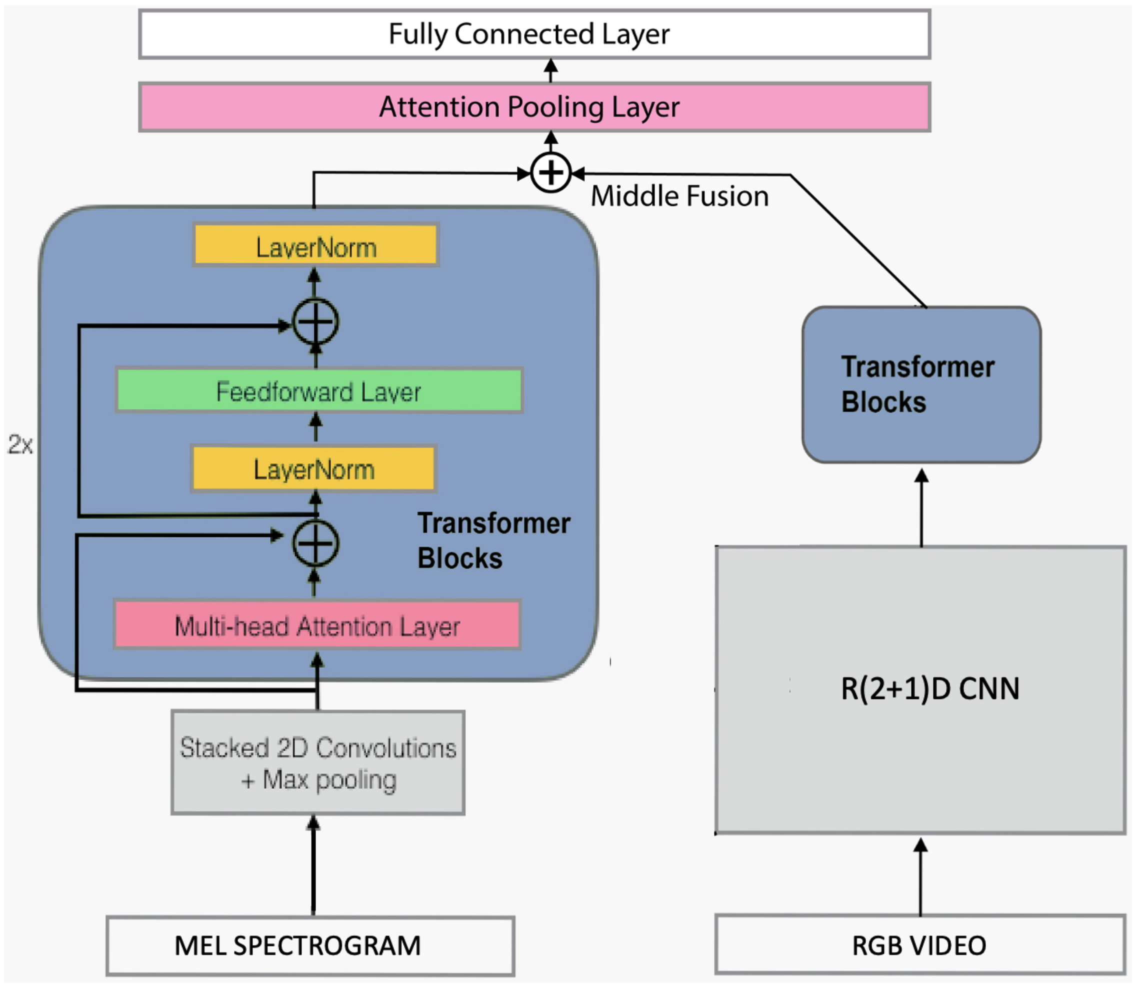


Figure 11.2: The overall architecture, the audio branch (left) uses Convolution self-attention architecture, video branch is on the right. Mid fusion involves the concatenation step described in §11.3.1.

11.7.2 Adversarial Perturbation & Adversarial Training

All of the adversarial perturbations in this chapter are computed with the PGD method [Mad+17], with $\epsilon = 0.1$, ℓ_2, trained over 20 iterations. Attack budget (ϵ and # of iteration) are constrained to be the same across unimodal, multimodal experiments. Adversarial Training is performed by the methods mentioned in [WRK20].

Models	Attack	**mAP**	**AUC**	**d-prime**
Audio UniModal (PANNS) [Kon+19a]	No	0.383	0.963	2.521
Audio UniModal	Yes	0.183	0.895	1.770
Mid Fusion (G-blend) [WTF20]	No	**0.427**	0.971	2.686
Mid Fusion	Yes A+V	0.182	0.889	1.836
Mid Fusion	Yes V-only	0.339	0.954	2.441
Mid Fusion	Yes A-only	0.310	0.940	2.276
Mid Fusion mixup	No	0.424	0.972	2.711
Mid Fusion mixup	Yes A+V	0.234	0.891	1.983
Mid Fusion *AT*	No	0.397	0.964	2.530
Mid Fusion *AT*	Yes A+V	0.199	0.900	1.861

Table 11.1: We use the overall architecture shown in Figure 11.2. Here, mAP is the mean average precision, AUC is the area under the false positive rate and true positive rate (recall). The d-prime can be calculated from AUC [Gem+17b]. AT denotes adversarial training. A red text color indicates the most potent perturbation against the model.

Models	Attack	**Acc**	**mAP**	**AUC**
Mid Fusion	No	**86.5%**	0.853	0.987
Mid Fusion	Yes A+V	5.0%	0.513	0.824
Mid Fusion	Yes V-only	6.6%	0.541	0.848
Mid Fusion	Yes A-only	85.6%	0.851	0.987
Mid Fusion mixup	No	85.5%	0.854	0.987
Mid Fusion mixup	Yes A+V	15.2%	0.734	0.623

Table 11.2: Performance of CSN models on Kinetics-Sounds.

11.7.3 Results and Discussion

As we can see from Tables 11.1 and 11.2, we find empirical evidence supporting Theorem 1, as uni-modal attacks (Video-only, Audio-only) can successfully break the multimodal network. Multimodal attacks are however more potent than unimodal attacks given the same attack budget, with the same ϵ and #iterations causing more accuracy drop compared

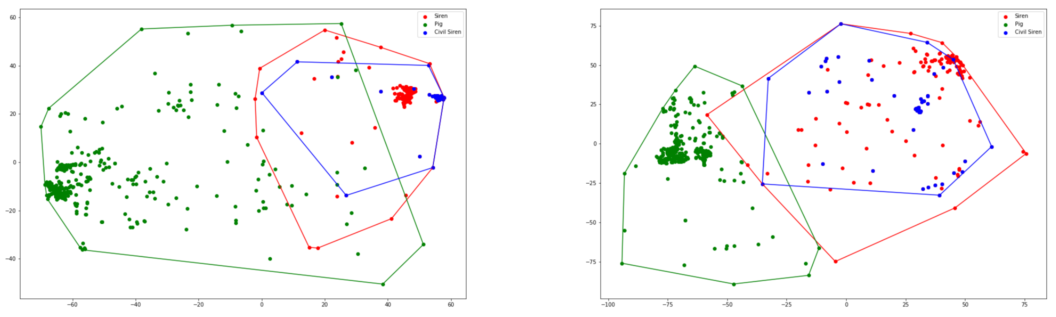


Figure 11.3: Left: TSNE of the vanilla fusion feature. Right: TSNE of the mixup fusion feature. Three classes: Red $\longrightarrow$ Siren; Blue $\longrightarrow$ Civil Siren (subclass of Siren); Green $\longrightarrow$ Pig.

to a unimodal attack.

We could also see the benefit of our proposed mixup, which could help minimize the accuracy loss significantly on both AudioSet and Kinetics-Sounds, compared to vanilla mid-fusion and traditional adversarial training (AT). Interestingly, among AudioSet where audio event classification should be the dominant task, audio modality showed on-par robustness compared to video. While in the Kinetics-Sounds dataset where video event classification is the dominant task, audio attacks did not affect the classifier at all, video modality dominantly decides the robustness. We pay the cost of sampling for measuring the convexity and density, whereas adversarial Training needs to pay the cost of both computing the adversarial examples and retraining with them, our method is a lot cheaper in terms of computational cost.

In Fig. 11.4, Performance Drop Rate:= $\frac{\text{clean performance}-\text{performance under attack}}{\text{clean performance}}$. If we plot using density as the x-axis, we could observe a similar correlation trend, but a bit noisier compared to Fig. 11.4. The density $\rho_c^{R,p}$ (see §11.5.1) is measured at 60 and 80 quantiles of the class in ℓ_2 norm. We can observe a negative correlation between the drop rate and the convexity of the class, suggesting a positive correlation between robustness and convexity. We could also see higher density would correlate to more robustness for an audio class, some denser classes tend to be more robust despite having low convexity. Intriguingly, we see the pattern that simpler waveforms such as siren (highlighted), bell sounds, child crying are more robust than complex signals like traffic/mechanics sounds. Our empirical findings verify our conjecture that both the density and the convexity of the data are simple factors that could severely affect its robustness. Fig. 11.3 shows the convex hull of the bottleneck feature of the mixup fusion model and vanilla mid-fusion model, where we observe that the mixup model clearly pushes samples of a class to form a denser outer crust and encourages a clearer boundary between classes resulting in a higher convexity within selected classes.

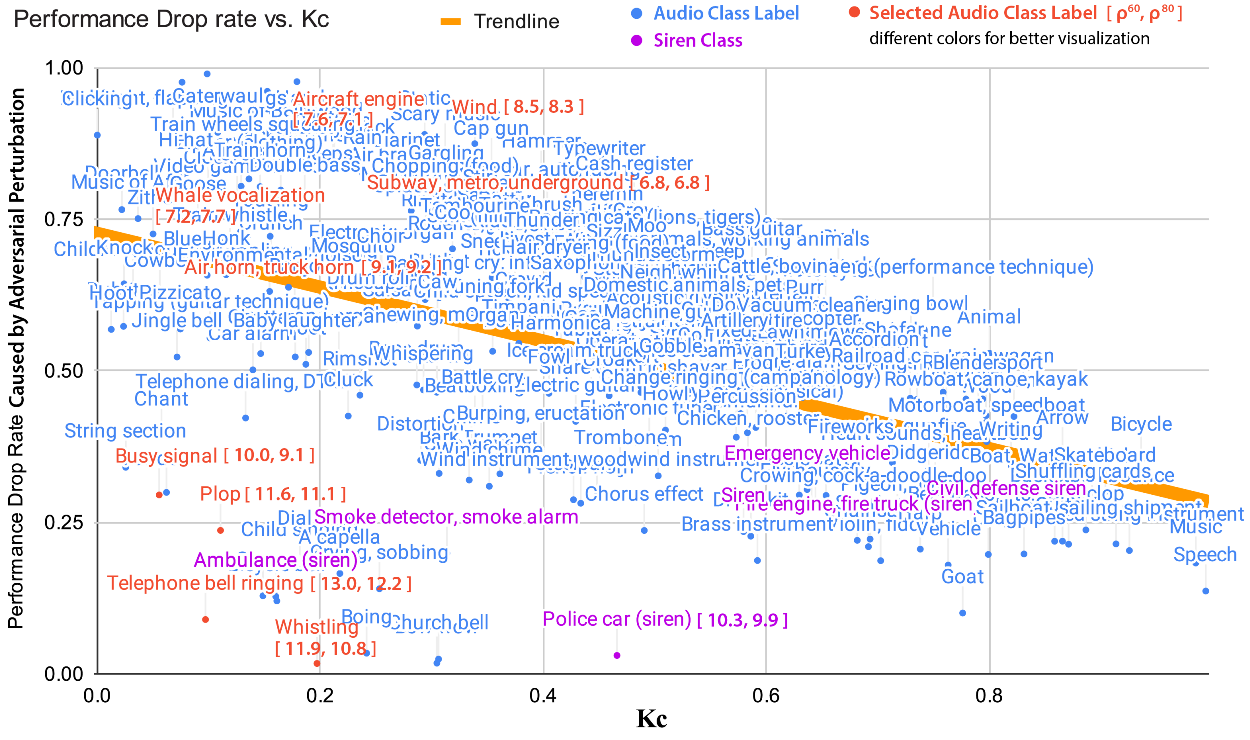


Figure 11.4: Performance Drop Rate% VS Convexity (κ_c) for all audio classes in AudioSet. Density $\rho_c^{R,p}$ partially shown due to space limit.

11.8 Conclusion

In contrast to some common notions, our work shows that multi-modal systems are not always more robust to adversarial attacks. In this chapter, we propose alternative metrics to understand the adversarial robustness of large-scale multi-modal models, and we empirically show the effectiveness of our density and convexity metric. We further propose a novel multi-modal mix-up method that selectively augments the denser samples in less convex classes to compensate for the robustness, and which outperforms simple adversarial training with respect to robustness. Our experiments on AudioSet [Gem+17b] and Kinetic-Sounds dataset [AZ17a] verify our hypothesis and the effectiveness of the mix-up strategy.

Chapter 12

Noise Robustness under Common Corruptions *versus* Adversarial Robustness

12.1 Introduction

In previous chapters, including Chapter 8, 9, and 11, we have delved into the topic of adversarial robustness. Given the high cost of adversarial training (AT), it would be hardly practical to pay the cost of AT before deploying larger and larger models trained on more and more data. Therefore, understanding different models' clean performance is important for us to understand each of their vanilla robustness against noise.

The significance of *noise robustness* of audio recognition (AR) systems is highlighted with the success of deploying Automatic Speech Recognition (ASR), and acoustic event detection (AED) techniques in various indoor and outdoor setups like mobile devices and vehicles [LSK19]. Previous AR pipelines rely on preprocessing or enhancement modules to filter out noises in the audio input. However, those approaches do not guarantee to eliminate noises completely [Mic+21; WC18], which still cascades to downstream modules of the pipelines. Therefore, measurements of the effects of these noises help to identify the robustness of the AR systems. In this chapter, we focus on the robustness of models without noise-filtering preprocessing steps in AED: predicting the most likely semantic label for the acoustic event in an audio stream. General environmental sounds, which are naturally noisy, do not include as many contextual dependencies or senses as their human speech subset [Li+22a]. Besides, AED does not usually involve language models, avoiding extra complexity for error analysis. Meanwhile, our findings about different neural models on the AED task could inform modeling choices for all other audio-related tasks such as ASR, speaker identification, etc.

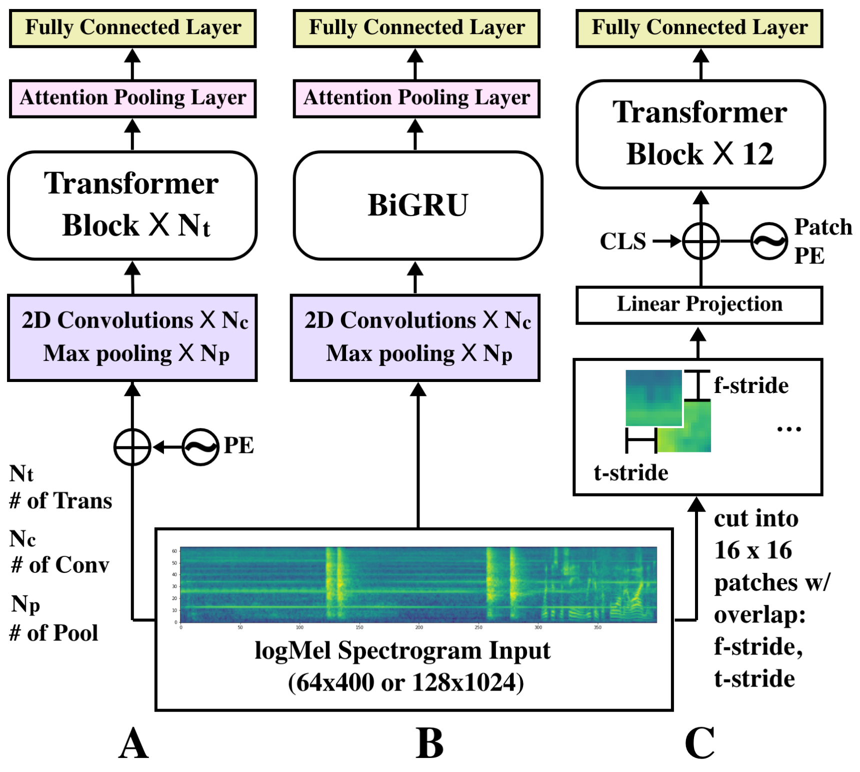


Figure 12.1: (A): CNN+Transformer, (B): CRNN (C): ViT. PE: positional encoding; $N_t = 2$, $N_c = 10$, $N_p = 5$, f-stride=t-stride= 8

There is a recent rise of Vision Transformer (ViT) [GCG21a] in AED over the previous CNN variants models [Kon+19a; Li+22a]. However, CNN-based models still have advantages in efficiencies and training efforts [Li+22a]. Those two categories of models have several major differences: CNN has good efficiency properties because of its effective input downsampling and discretizing the mathematical convolution into matrix multiplication, excelling at capturing local features; whereas ViTs capture global patch-wise correlation from self-attention and bear little inductive bias. Such distinctions may lead to their differences in properties in terms of robustness in audio classification. Recent studies from the vision community [PC21; Sha+21; Bho+21; Nas+21] show that ViTs are more robust learners compared to their CNN-based counterparts. For example, the model robustness increases with more layers of transformer blocks, and convolution does the opposite [Sha+21], and CNN-based models could improve their robustness if adopting ViT's training recipes [Zho+22]. However, all those findings are empirically collected on vision tasks, whether similar property stands in the audio domain remains unclear.

We therefore thoroughly investigate the difference between CNN-based and ViT-based models on AED in terms of noise robustness. We reproduced 4 competitive checkpoints from a recent study [Li+22a](CNN+Transformer, ViT, ResNet and CRNN) for comparison and conducted experiments on AudioSet. In our study, we start with the model's "clean" performance on the AudioSet test set without any input perturbation and report performance changes when the model is under the influence of various types of noises. We first mask out a portion of the audio to test the change in the performance of different models. Then, we model the intermittent noise and continuous noise as *occlusion* and *colored noise* respectively. Additionally, we conduct a pressure test by investigating the performance under adversarial perturbations. Lastly, we investigate the connection between the loss landscape and robustness of models by measuring model sharpness [Kes+16]. Our major takeaways are three-fold:

1. By conducting a thorough comparison, we have observed a substantial disparity in the resistance of various neural architectures to different forms of noise.

2. Through stress testing different neural architectures using noise, we conclude that ViT (attention architectures) exhibits greater overall robustness than other architectures.

3. Robustness-aware trained models tend to have flat loss surfaces, manifesting in terms of low sharpness score, however, currently, there is no fair method to compare sharpness across different architecture, nor there does measuring the sharpness of standard-trained model indicate its robustness.

Dataset and Setup: We follow the same setup as Chapter 6 for AudioSet. Table 12.1 first 4 rows list state-of-the-art *single models* and for the AED task trained on the *full* AudioSet and test on the *eval* set.[1] To limit variability, we do not include ensembled models, weight-averaged, multi-modal models. Following [Li+22a], we trained 4 models with competitive performance with comparable numbers of parameters, using the same training recipe, so that comparison could be carried out fairly. The training procedure details and parameters are identical to Chapter 6. (LR: $4E-4$, step decay, Adam optimizer, batch size 448, 10 epochs) The architecture of each model is shown in Figure 12.1, ResNet50's architecture is omitted due to its prevalence. We use the 64x400 feature input models in the experiments of this work for better efficiency

[1]The waveform is downsampled to 16 kHz; frames of 1,024 samples (64 ms) are taken with a hop of 400 samples (25 ms); each frame is Hanning windowed and padded to 4,096 samples before taking the Fourier transform; the filterbank of 64 triangle filters spans a frequency range from 0 Hz to 8 kHz.

Model	**#Param**	mAP	sharpness	λ_{max}
AST/ViT-S	22M	**0.41**	9.47	659.8
CNN+Trans	12.1M	**0.41**	3.75	252.2
ResNet50	25.6M	**0.38**	8.81	150.2
CRNN	10.5M	**0.41**	6.64	112.2
AST/ViT-S-SAM	22M	**0.43**	0.91	18.9
AST/ViT-S-AT	22M	**0.33**	0.96	26.5

Table 12.1: we adopted 64×400 logMel spectrogram features. Sharpness is computed following [Meh+21], the higher the score the sharper the minima. Here, first 4 rows are models trained with standard training. ViT-SAM indicates ViT trained with SAM [For+20] using $\rho = 0.2$, and ViT-AT is ViT trained with Adversarial training using the universal adversarial examples computed in § 12.2.4
.

12.2 Common Corruptions and Adversarial Perturbations

12.2.1 Temporal Masking/Occlusions

We leverage masking over temporal frames as the probing tool for analyzing the robustness difference between Transformer-based and CNN-based architectures. We employ the blank temporal mask, which stands for silence on different parts of 10-second audio snippets on all AudioSet *eval set* samples, shown in Fig 12.2 (a-g)[2] to collect the performance changes under different masking strategies. Fig 12.3 demonstrates that less information exposure to the model (e.g. longer masking, loss of context) results in lower performance. Fig 12.2 (a-b) mask out a 50% consecutive chunk of the audio snippet, which results in 8% mAP decreases in Fig 12.3, and the position of the mask has negligible effects. However, larger negative impacts are introduced as the mask interval decreases, despite that the total number of frames being masked remains the same (50%).

Surprisingly, we could observe the existence of a threshold masking interval which significantly decreases the model performance. For instance, masking every 0.25s for

[2]We apply masks directly on logMel filterbank features, for the readers who are interested in masking WAV files, Gibbs phenomenon will cause undesirable distortions of the spectrogram. Naturally, one would ask what is the effect of masking different frequency bands, but we believe such cases do not pose a realistic threat in reality

AST/ViT and CRNN almost nullified the model's capability to recognize. Inspired by George Miller's psychological study [Cow15], neural networks resemble human memory in some ways, showing there's a "magic number" period of information machine perception could remember. Taking a closer look into the class-wise performance, under $0.125s - 0.25s$ masking, there are only a handful of classes that the models could still predict with above 0.3 mAP: Speech, Music, Whistling, whispering, smoke detector, and fire alarm. Even the classes previously found [Li+22b] to be robust to high-frequency adversarial noise, e.g. Police cars, emergency vehicles, and siren classes, are beyond recognizing for our model.

Counter-intuitively, if we assemble the 0.125s fragments which were not masked to form a new 5s audio, the performance would recover dramatically, despite that we can clearly see the cutoff effect from the spectrograms. (see Fig 12.2(h,i), Fig 12.3(0.125s-1s Concat columns)) Interestingly, the performance of reassembled spectrogram would "recover" to the equivalent value as if the mask was a consecutive 5s mask on the spectrogram. Evidently, ViT/AST architectures and CRNNs are comparatively more robust architectures under occlusion, with their performance comparatively outperforming all other models under different types of occlusions. If noise spans longer than the convolution receptive field, we would expect the performance of CNNs to tank more. This is less observable in Figure 12.3 as compared to in Figure 12.4. The aforementioned observations suggest that a model's resistance to occlusion is largely influenced by the global correlation of its base learning unit or resolution. Specifically, Transformers learn from patches, and the more intricate the details these patches capture, the greater their robustness. On the other hand, CNNs learn via kernels. The wider the global correlation these kernels can perceive, the greater their resilience to occlusion. In practice, we conjecture it is much easier for Transformers to readily build the global correlation than for CNNs to learn using huge-size kernels, therefore, Transformers appear to be more robust empirically. This specific behavior could be leveraged to infer neural architecture under a black box setting. This could be a potent threat model for adversarial attacks in the audio domain.

12.2.2 Strong label masking

With the release of new strong labels from AudioSet [Her+21], we have knowledge of when a specific event occurs exactly for 12,091 samples among the entire 20,123 *eval set*, where the strong labels match with the weak labels. We compare the following settings:
Context-only We mask out the strong labels of each test sample and compare the performance of different models (Fig 12.2 (j)).
Strong label-only Conversely, we also expose the strong label portion to the model. 'Masking Context' indicates masking the complement of the strong label portions (Fig 12.2 (k)).
Concatenations Either the strong label portion (Fig 12.2 (l)) or the context portion (Fig 12.2 (m)) are concatenated.

Results in Fig 12.5 show that the strong labels are crucial to models' performance, and context-masking leads to moderate performance drop. However, concatenations bring marginal boost.

12.2.3 Colored Noise

It is a common practice to use colored noise as the framework to model continuous noise in speech. We investigate the effects of two white (Gaussian) noise setups. 1) Adding Gaussian noise $\sim \mathcal{N}(0, 0.002 \sim 0.1)$ onto the logMel spectrogram features (Fig 12.2(n)). 2) Adding Gaussian noise $\sim \mathcal{N}(0, 0.001 \sim 0.1)$ to the audio files and let the noise cascade into the logMel filters through the FFT step and the Mel Filters (Fig 12.2(o)). The former 2D noise is strong in magnitude compared to the latter 1D noise. And the results in Fig 12.6a and Fig 12.6b show that ViT is more robust to both noises than CNNS. Meanwhile, we find that those models have similar responses to pink, brown, and blue noises. The results for other colored noises are not included but can be found in our open-source repository. It's noteworthy that hybrid models like CRNN and CNN-Transformer exhibited specific vulnerabilities to colored noise introduced into 2D features, but not to noise introduced into 1D waveforms. Their performance even deteriorates below that of ResNet in this scenario, implying that the 2D Gaussian noise is more effectively disrupting their base learning unit than it is confounding the other two types of models. It's important to mention that determining the base learning unit of hybrid models is challenging, as their features undergo several non-linear, non-convex convolution layers before reaching the stage of building global correlation. At present, we lack the necessary tools for a precise examination of hybrid models. This would be interesting for further analysis in the future.

12.2.4 Adversarial Perturbation

Following [Li+22b] we also compute white-box universal adversarial perturbations (ℓ_∞ or ℓ_2 norm) against all the selected models and add them to the input logMel feature. The upgrade we made here is we use $APGD_{DLR}$ [CH20] instead of PGD attack, for ϵ from 0.005 to 0.05. This update was for the sake of completeness, the results did not differ much from PGD, and thus we do not repeat the result of PGD in the graph. Fig 12.2 (p, q) show examples in examples of two perturbations with $\epsilon = 0.005$, where the ℓ_∞ norm attack generates sparse lacunae in the logMel spectrogram. From the results in Fig 12.6c and Fig 12.6d, we can see that ViT is generally more robust than other models under adversarial perturbations, the performance for ℓ_2 does not differ much from ℓ_∞, and in fact ℓ_∞ could be thought of as a scaled-version of ℓ_2 attack. The performance of ResNet is cratered, which indicates its challenges to defend adversarial perturbations without specific temporal dependency modeling modules, which is also found in [Zho+22]. Notably, CRNN is less robust than ViT under adversarial perturbations despite its similar robustness compared to ViT under occlusions.

12.3 Through the Lens of Confusion Matrix

The confusion matrix represents true positive, false positive, true negative, and false negative predictions made by the model. This notion of a confusion matrix is primarily for binary or multi-class classification problems, where the definition of a confusion matrix is readily defined and very easy to implement. e.g. common libraries have existing functions.

However, in the case of multilabel classification, things are a bit more complex because each label is a binary decision, but the labels are not mutually exclusive. Following the canonical definition of confusion matrix, we would draw a 2×2 confusion matrix for each class, we would have to relax multilabel classification to multi-class classification and then draw the confusion matrix accordingly. As is shown in Figure 12.8, this results in a very sparse confusion matrix only showing a few bad-performing classes including *73 Domestic animals, pets; 138 Musical instruments; 72 Animal; 137 Music*, which we were pretty familiar with.

Although this problem might look deceiving simple, we realize relaxing multilabel problems into multi-class classification could be suboptimal. This urges us to trace back to the core definition of the metrics of multilabel classification as we discussed in Chapter 6.8and Chapter 4.2.2, where the mean Average Precision is defined as:

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n \tag{12.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{12.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{12.3}$$

where P_n and R_n are the precision and recall at the n_{th} threshold. This definition, in most cases, proves effective as it provides a conservative measure of a classifier's performance. By employing macro-averaging across various thresholds of the ROC curve, it distills the overall performance into a single numeric value. This complements other metrics such as AUC and d-prime.

However, there's a trade-off for its simplicity and interpretability. While it shows how well a classifier performs on individual classes by capturing True Positives (TP), False Positives (FP), and False Negatives (FN) in a binary setting, it doesn't provide insights into the specific classes the classifier is misidentifying. In other words, we cannot discern the specifics of misclassification as it doesn't reveal what the classifier is confusing one class with.

In fact, the theoretical guarantee behind multilabel classification is much more involved than the multi-class setting, this is in line with the derivation in another recent study [JQG22]. To derive the equivalent certified robustness as [CRK19], [JQG22] used a much more involved variant of Neyman-Pearson Lemma[NP33] that is applicable to multiple functions, which correspond to multiple labels predicted by the base multi-label classifier. Our certified radius on the AudioSet case could be obtained following the same setup as [JQG22], however, in our case, the certified radius is all below mAP of under 10%, which carries little practical value. Plus, since this thesis is more concerned with the real-world implication of robustness, we do not expand the discussion of certified radius here. Inspired by [JQG22], we redefined confusion matrixes for multilabel settings with top-k settings, specifically, we calculate the True Positives (TP) by finding the common indices between actual and top-k predicted labels. For each True Positive, we increment the corresponding cell in the confusion matrix. We then calculate the False Negatives (FN) by identifying the actual labels that are not among the top-k predicted labels, and the False Positives (FP) by identifying the top-k predicted labels that are not actual labels. For each False Negative and False Positive pair, we increment the corresponding cell in the confusion matrix. The detailed implementation could be found in the same repository as Chapter 6. This results in a much better overview of the classifier's capability as shown in Figure12.9. As we can see, clearly, the confusion areas are 0-75 (human-sound category), 75- 136 (animal), 137-282 (music), and 283-336 (sound of things), the rest are ambiguous and random sounds. This shows a well-performing classifier such as AST demonstrates a strong capability to learn the ontology of AudioSet automatically, and we observe very similar patterns for other well-performing models.

Through the lens of the confusion matrix, we can observe that though different attacks would reduce the terminal mAP of a model to a very low level, they work very differently. As is shown in Figure 12.10, *versus* Figure 12.12, adversarial perturbations fool the models into overpredicting on music and noise categories, whereas Gaussian noises trick the classifiers more into various existing noise categories. This is more evident in Figure 12.13, as the classifier fed with occluded information mainly over-predicts close to 300-index labels which correspond to siren/alarm classes, which is strong evidence that the model under noise perturbation or attack is simply "nudging" its gradient to the closest distribution it understands.

12.4 Through the Lens of Sharpness:

Except for the metrics or confusion matrix of the results, we would hope loss surface could help us identify robust models before we empirically measure their robustness.
Sharpness Metric: We follow the ϵ-sharpness definition of [Kes+16] which defines sharpness as a maximum loss within a neighborhood bounded by ϵ. Concretely, let $\boldsymbol{w}$

denote the collection of all model weights; then $\max_{|\epsilon|_2<\rho} \mathcal{L}(\boldsymbol{w}+\epsilon) - \mathcal{L}(\boldsymbol{w})$ is small under a given radius, ρ. $\mathcal{L}(w)$ is the loss function with parameters w, ϵ is a tiny perturbation in the ℓ_p ball. In table 12.1 the sharpness score is computed following [Meh+21] using $\rho = 0.05$. Intuitively, this formulation represents the maximum loss in the local ball near the minima in the weight space. ϵ-sharpness scores of different models are listed in Table 12.1, where higher score indicates sharper minima. At first glance, it is difficult to see correlations between the ϵ-sharpness scores and the robustness as indicated in the empirical tests in Figure 12.7.

Upon reconsideration, we identified several concerns: 1) To ensure a fair comparison, we assigned a uniform $\rho = 0.05$ across architectures, however, each architecture has its optimal ρ value for SAM training (would be the same for the measurement, since [Meh+21] followed SAM to compute the sharpness score). 2) The varying loss scales across different models could potentially distort the ϵ-sharpness metric. As such, it may not be feasible for the ϵ-sharpness metric to reveal the distinctions between different models initially, nor might it be capable of indicating any related robustness.

However, we do observe both SAM-trained[For+20] and adversarial-trained[WRK20] models display way lower sharpness than standard-trained models as it is indicated in Table 12.1 for the AST/ViT case. Both the SAM-trained model and adversarial-trained model demonstrate superior robustness against adversarial perturbation as is shown in Figure 12.7a. Put simply, while robust models tend to have less sharp minima, it is not effective to use the ϵ-sharpness metric to compare across different architectures and it would not yield any insights about robustness.

On the other hand, since there exist different definitions of sharpness based on Hessian, although it is also not scale-invariant, for broader comparison, we computed the largest eigenvalue of the Hessian $\nabla^2\mathcal{L}(w)$ following [CHG22]'s setup using 10% of AudioSet balanced set via power iteration by using PyHessian[Yao+20], 100 iterations, and reported in table 12.1 as well. We observe very similar trend as the ϵ-sharpness metric.

Though our aim here was to draw observations about sharpness *versus* robustness, our findings suggest that using the current definition of sharpness might not support such a comparison. This calls for the future development of scale-invariant sharpness measurements across architectures.

Sharpness related discussion: There have been debates about whether flatter minima surfaces help with generalization, the positive evidence has been confirmed in many works [For+20; Meh+21] and etc, until [KCL23] [And+23] empirically found counter-evidence that sharper minima could generalize as well. A recent work [WML23] theoretically explains this seemingly contradictory phenomenon proving such contradictory indeed would exist.

12.5 Conclusion

In our comparative analysis, the AST/ViT[GCG21a] architecture displayed more resilience compared to the CNN family to both colored and adversarial noise and was much better than pure CNN in defending against occlusion. We postulate that AST/ViT benefited from the fine-grain learning base unit-(i.e. patches of spectrogram) compared to coarser kernels CNNs are based on. Unlike CNNs, transformers do not have a fixed-size receptive field, they instead use self-attention mechanisms that consider the entire input space, allowing for a global receptive field. Meanwhile, the empirical evidence also indicates that a model's ability to withstand adversarial attacks does not necessarily imply a similar level of resilience against noise disturbances, suggesting no substantial correlation between adversarial robustness and noise robustness. This separation of robustness capabilities underlines the complexity of designing models that are universally robust against different types of perturbations. In addition, despite being able to observe that robustly trained models display lower sharpness, we were yet able to find a way to leverage the current definition of sharpness to compare different architectures. It highlights the need for a comprehensive approach when building and assessing robustness in neural network architectures. All aspects including model design, training techniques, dataset distribution, and metrics need to go through careful consideration.

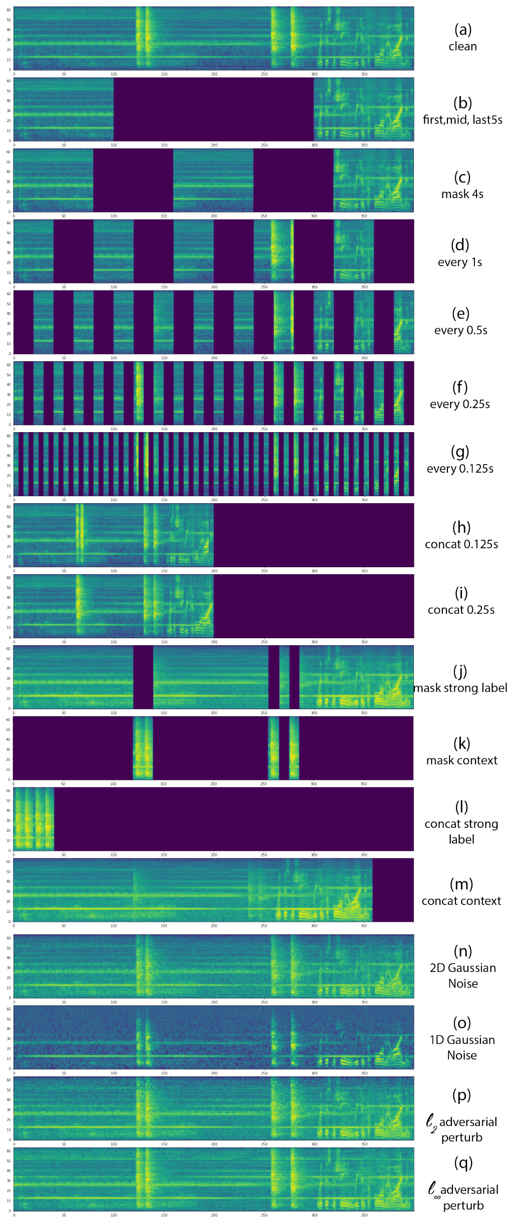


Figure 12.2: The sample sound here is a “hammer" class. Youtube id: –BfvyPmVMo. The 4 verticle lines are the actual hammering sounds (strong labels), and the dark part is the blank temporal mask. (b) - (c) refer to consecutive masking, and (d) - (g) refers to masking with different intervals. (h) - (i) concatenating the masked portions, equivalent to accelerating; (j) - (i) leaving out strongly labeled portion/context separately; (n) - (q) refer to adding Gaussian noises or perturbations.

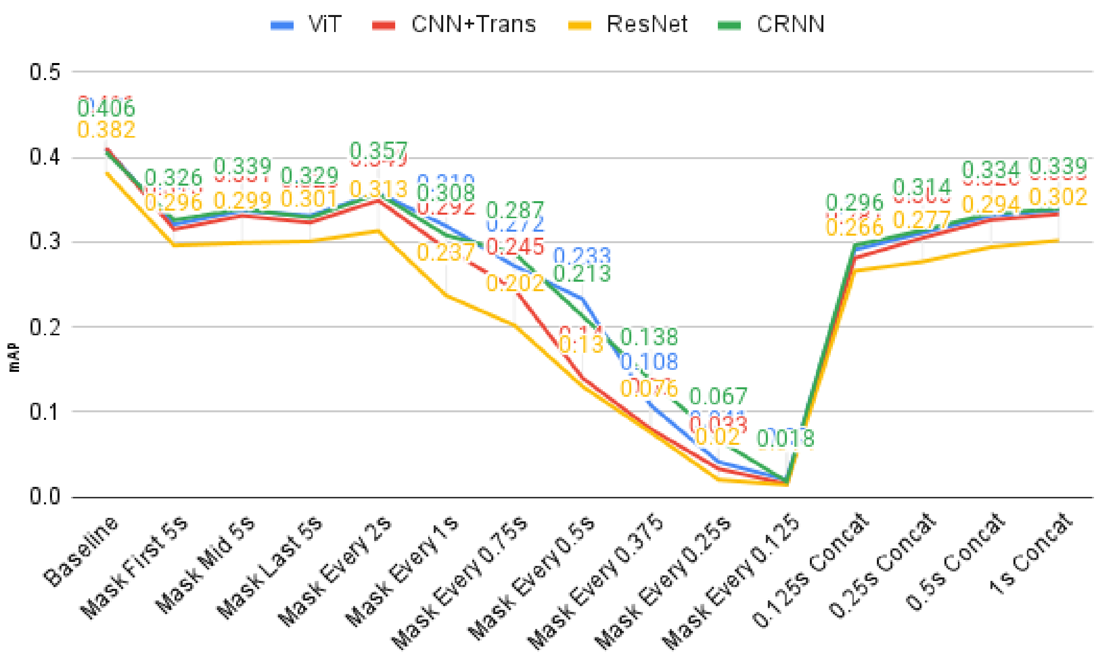


Figure 12.3: Following the occlusion patterns in Fig 12.2 (a-g), this is the performance of the 4 networks in comparison when getting occluded signals. Baseline means no masking (clean). When concatenating these fragments Fig 12.2 (h, i) together, we see the performance of all the networks "snap back". ViT/AST[GCG21a] showed are neck-to-neck with CRNN(TALNet)[WLM19].

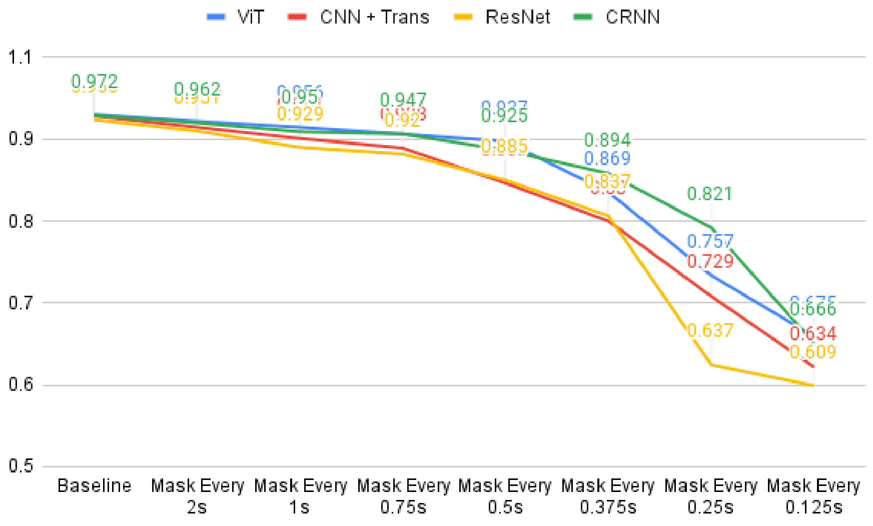


Figure 12.4: The kernel size is 7 and it corresponds to 0.175s of audio in the 10s audio (400-dim logMel). In Fig 12.3, we can see when masking every 0.25s, ResNet is the weakest performing one. This is more visible as AUC here. (All other networks are able to learn some longer-term correlation (global feature).) Note here, even at (mAP 0.018 in Fig 12.3) with finest-grain masking (every 0.125s), we are still performing way better than random guessing 527 classes. (Under such circumstances, only a few classes in the 527 classes would have non-zero mAP, the AUC value at this point is 0.609>0.5)

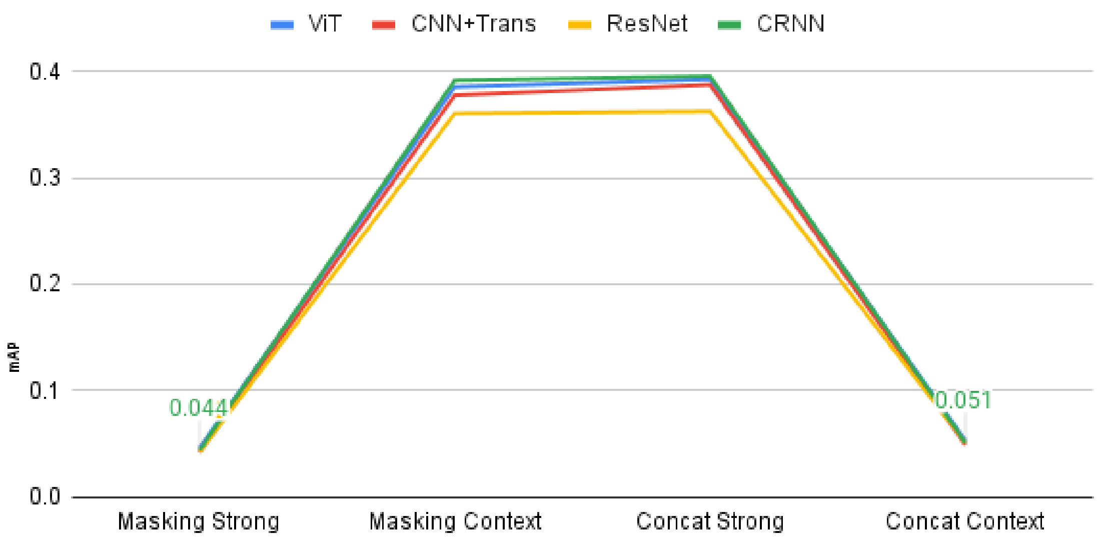
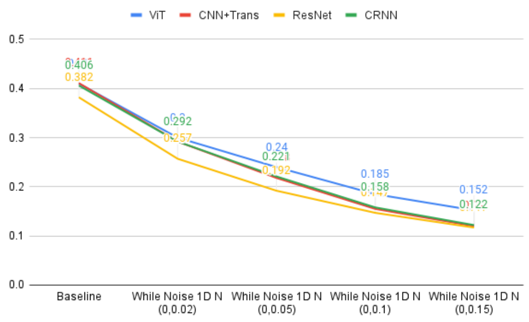
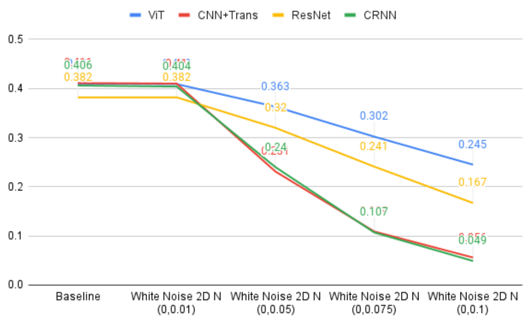


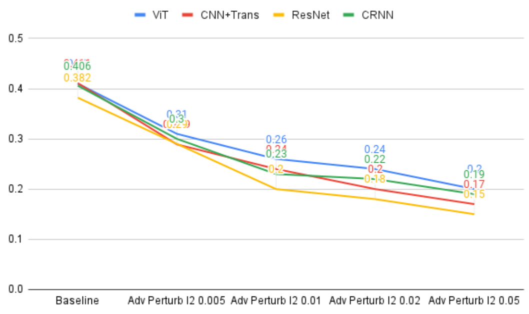
Figure 12.5: Performance when strongly-labeled portion of the signal is masked, *versus* the complement signal gets masked. See Fig 12.2 (j-m) for the effect.



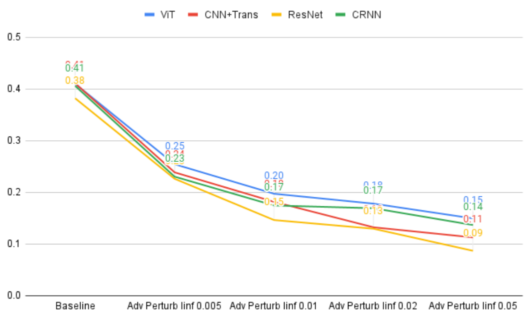
(a) Effect of different levels of Gaussian-noise added on waveform directly (1D)



(b) Effect of different levels of Gaussian-noise added on logMel (2D)

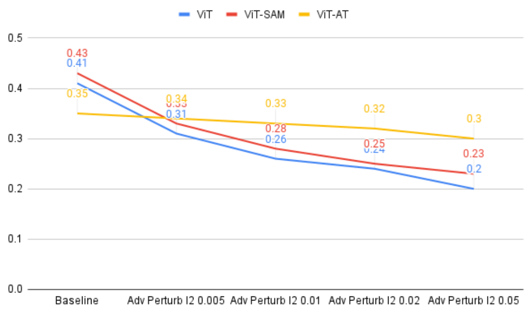


(c) Effect of different levels of ℓ_2 adversarial perturbation



(d) Effect of different levels of ℓ_∞ adversarial perturbation

Figure 12.6: Performance under the influence of Gaussian-noise and adversarial perturbations. The baseline is no masking (clean) performance.



(a) Effect of different levels of ℓ_2 adversarial perturbation on AST/ViT models trained by standard training, SAM, and Adversarial training (AT)

Figure 12.7: Performance under the influence of Gaussian-noise and adversarial perturbations. The baseline is no masking (clean) performance.

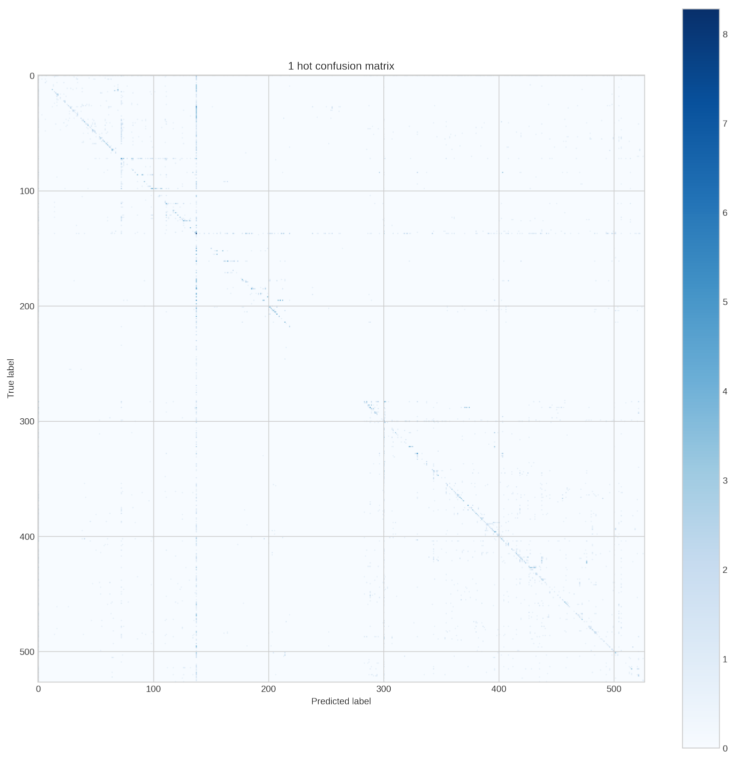


Figure 12.8: Confusion Matrix of AudioSet classification through relaxing multilabel to multi-class one-hot prediction (mAP 0.45 for AST model). 527 dimensions correspond to the 527 officially defined labels in the original order.Click here for label index

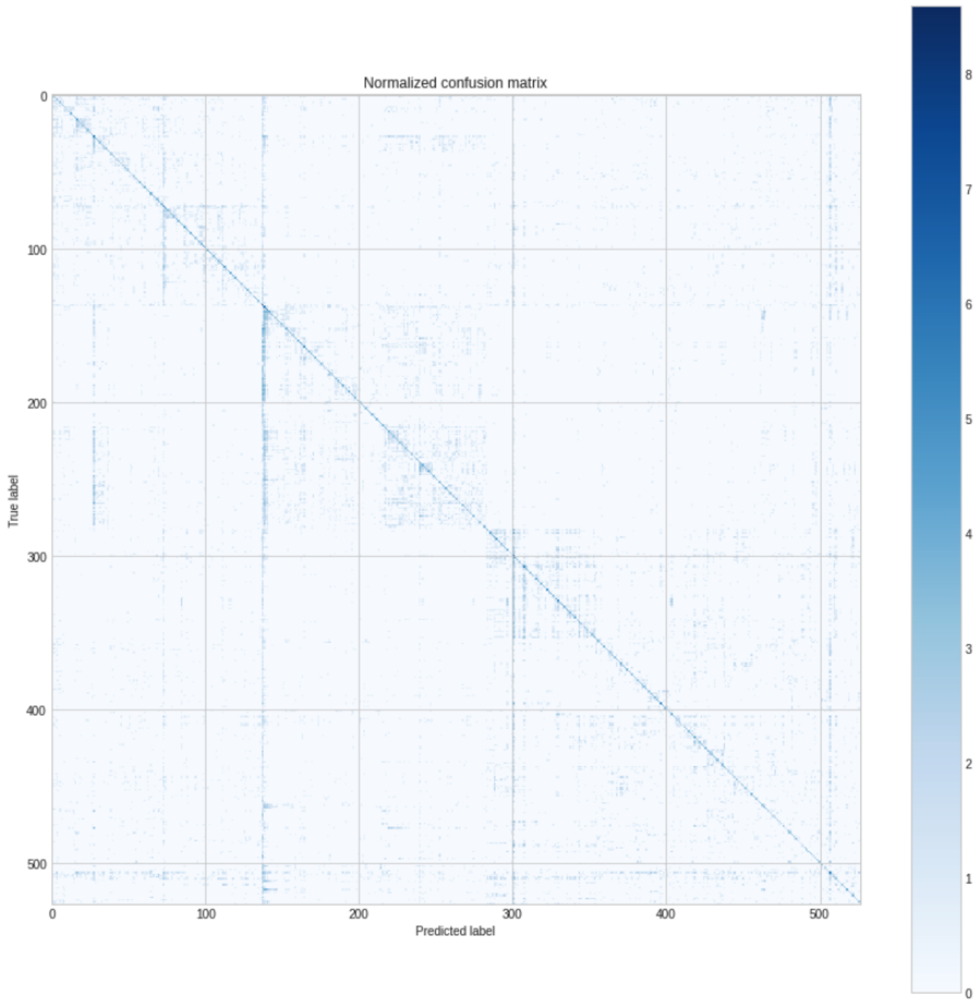


Figure 12.9: Confusion Matrix of AudioSet classification (mAP 0.45 for AST model). 527 dimensions correspond to the 527 officially defined labels in the original order.Click here for label index

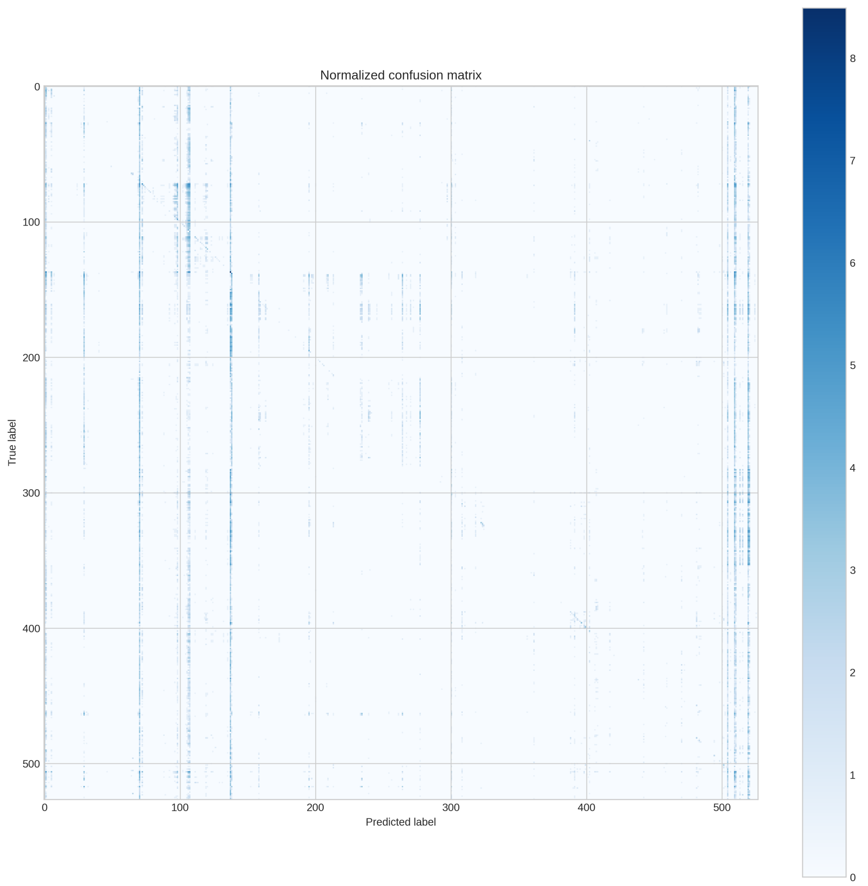


Figure 12.10: Confusion Matrix of AudioSet classification for Adversarial Perturbation ℓ_2 norm corresponding to Figure 12.3(p)(mAP 0.10 for AST model). 527 dimensions correspond to the 527 officially defined label in the original order. Click here for label index

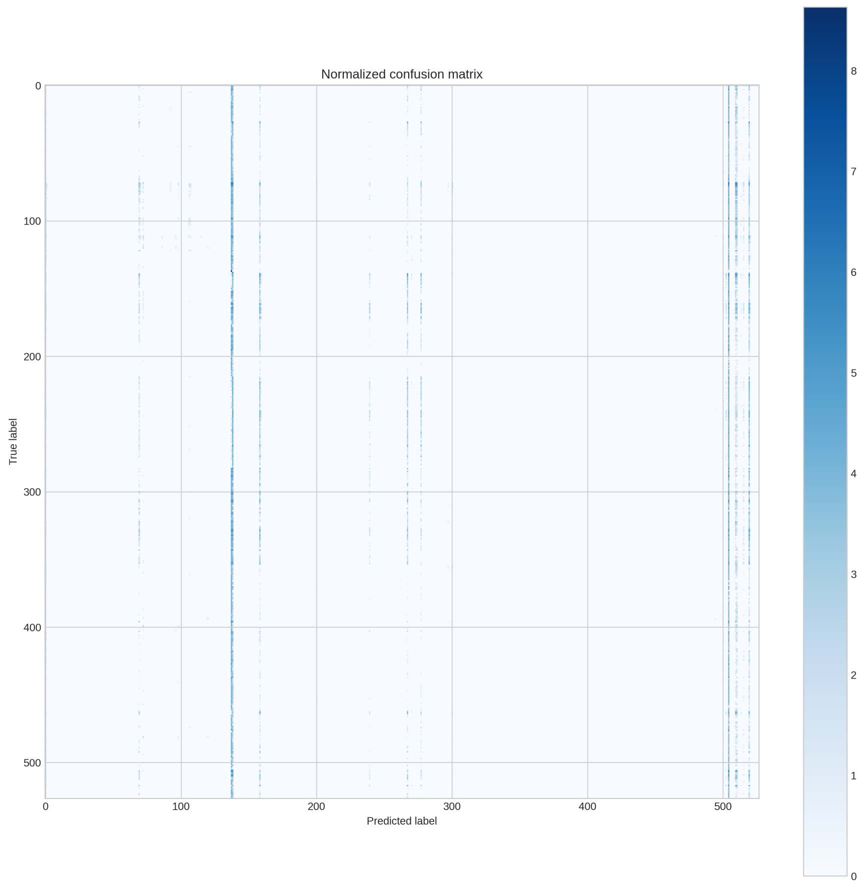


Figure 12.11: Confusion Matrix of AudioSet classification for Adversarial Perturbation ℓ_{inf} norm corresponding to Figure 12.3(q) (mAP 0.09 for AST model). 527 dimensions correspond to the 527 officially defined label in the original order. Click here for label index

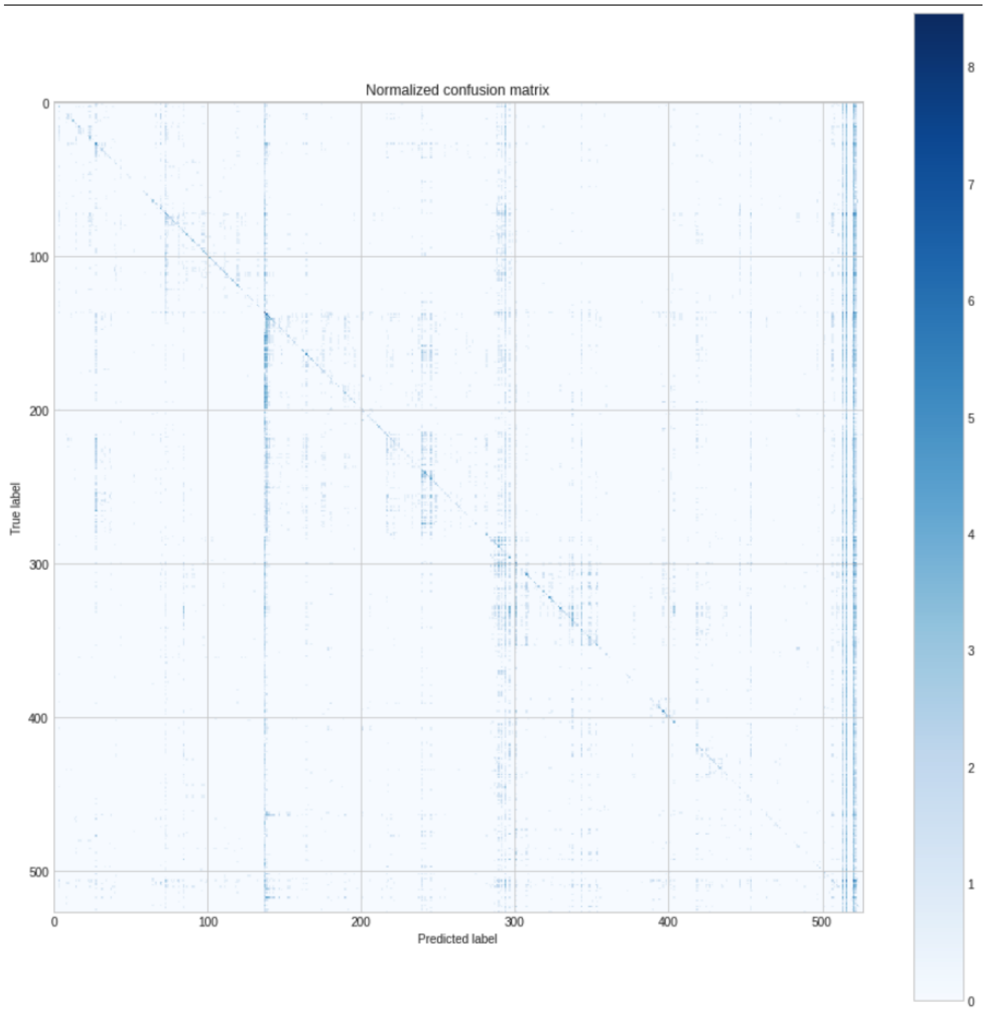


Figure 12.12: Confusion Matrix of AudioSet classification for Gaussian noise added on waveform corresponding to Figure 12.3(o), (mAP 0.09 for AST model). 527 dimensions correspond to the 527 officially defined label in the original order. Click here for label index

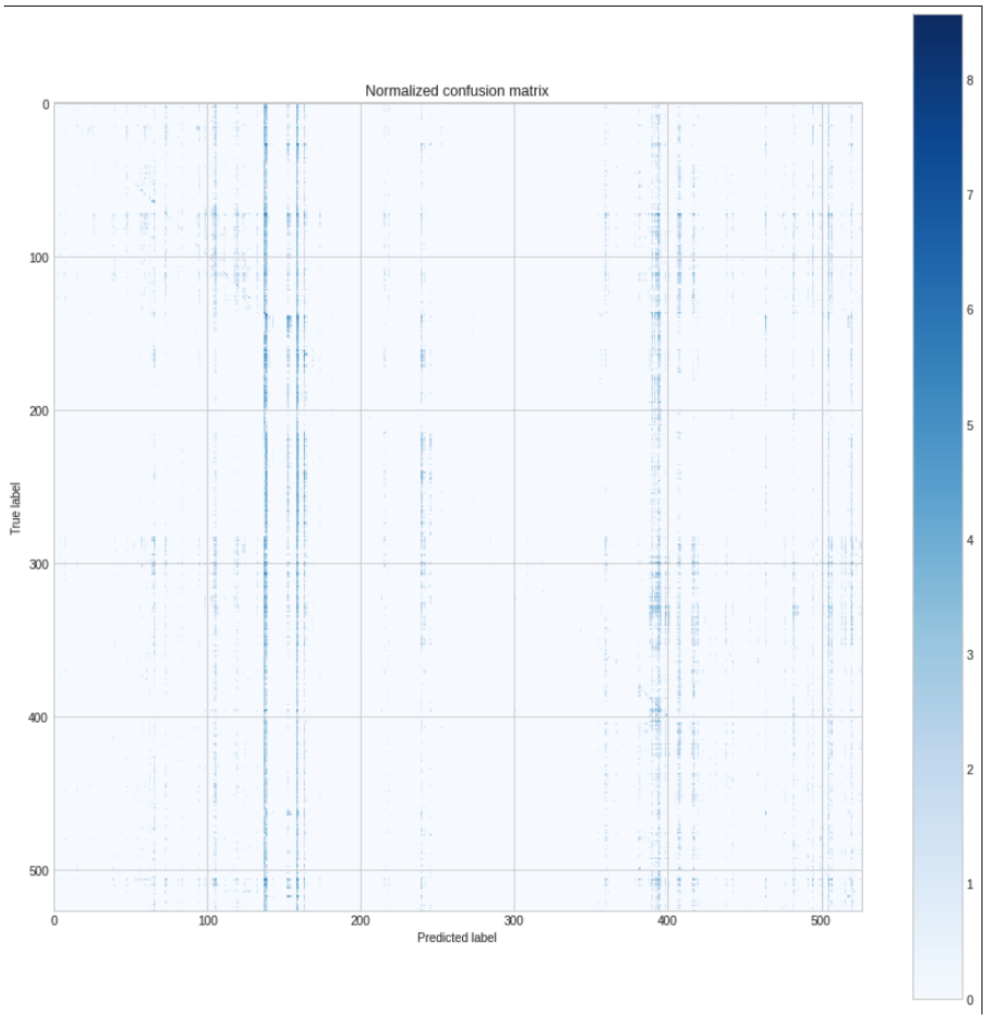


Figure 12.13: Confusion Matrix of AudioSet classification for occlusion corresponding to Figure 12.3(g), (mAP 0.03 for AST model). 527 dimensions correspond to the 527 officially defined label in the original order. Click here for label index

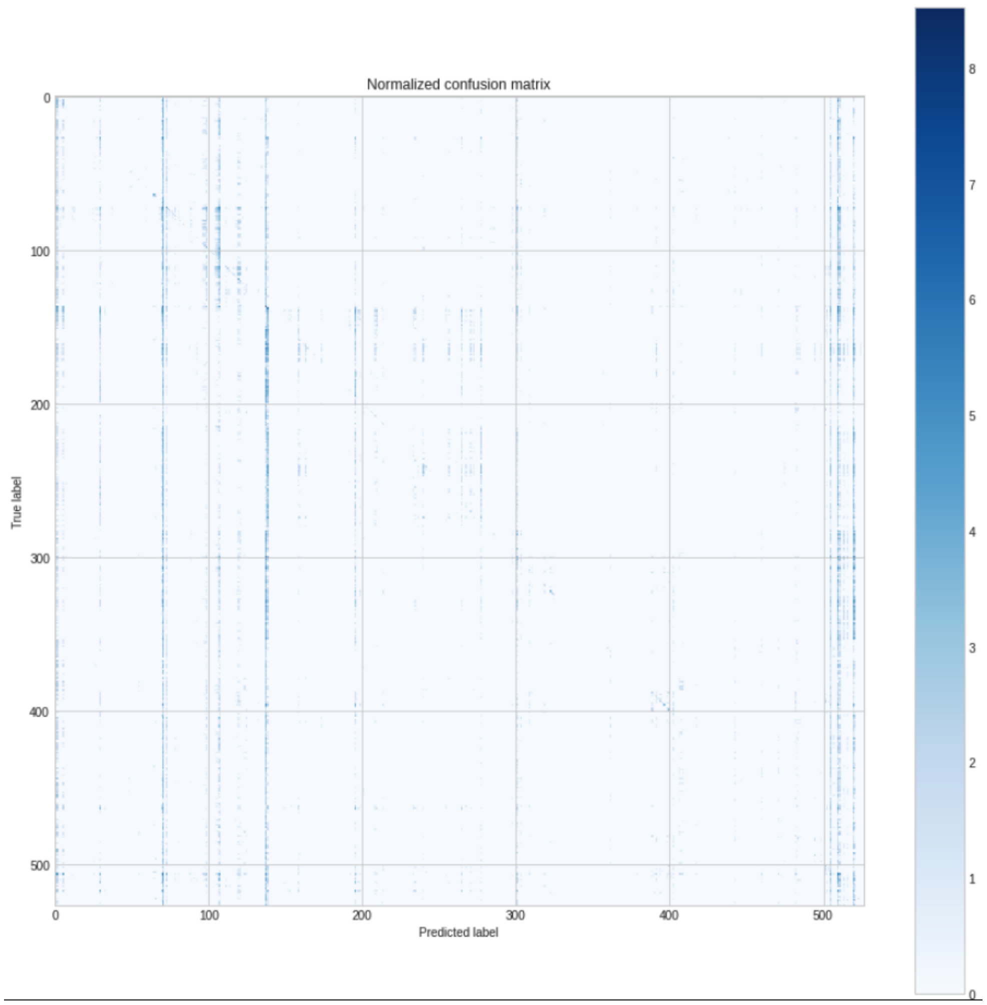


Figure 12.14: Confusion Matrix of AudioSet classification for Gaussian Noise added on feature corresponding to Figure 12.3(n)(mAP 0.06 for AST model). 527 dimensions correspond to the 527 officially defined label in the original order. Click here for label index

Part III

Towards Generalization and Robustness in Audio-Visual Event Recognition Systems

Realizing the aforementioned gap between adversarial robustness (toy problem) and real-world noise robustness, and realizing the limitation of existing empirical and theoretical robust methods, we identify the most practical path forward is to leverage the development of foundational models, denoising diffusion probabilistic models in our case, which could help us improve both generalization and robustness at the same time. In Chapter 13, also as the closing chapter of this thesis, first, we train a state-of-the-art conditional diffusion model. After demonstrating it could produce high-quality audio through generation metric, we leverage it to perform data augmentation through textual inversion and text-guided image-to-image generation, and we use the generated samples to mix with real data and demonstrate that this could be a promising way to augment the low resource data in the audio domain. Finally, we show that this diffusion model is inherently a perfect denoiser to fend off both noise and adversarial perturbation, which plugs into the randomized smoothing framework for defending against adversarial perturbation. In our experiments, diffusion denoising showed strong empirical evidence for defending against common corruptions in Chapter 12, and we also could achieve a robust certification.

Chapter 13

Audio-Journey: Diffusion to improve Generalization and Robustness

13.1 Introduction

In 2023, the field of machine learning for the audio domain, despite making significant strides, is arguably constrained by the scarcity of high-quality data. The largest datasets available, including AudioSet [Gem+17b], CLAP [Wu*+23], and VGGSound [Che+20a], comprise in total less than 3 million examples, falling orders of magnitude short of datasets in other domains, such as the Laion 5B Image-Text dataset [Sch+22]. Even for these datasets, the majority of annotations are collected through weak labeling, which may introduce noise and limit the definition of sound to a constrained definition.

This begs the question of how we should move forward to improve the generalizability and robustness of existing systems. Text-guided diffusion models have emerged as a promising solution for seamlessly integrating text and audio modalities. They hold potential not only for enhancing audio datasets, but also for denoising them. However, we face a classic chicken-and-egg conundrum, as we initially lack a substantial dataset that pairs text and audio to commence our work.([Kim+19] only has 50k samples) In our efforts to create text-audio pairs, we leverage Large Language Models (LLMs), which recent research has shown to possess the capability to extract substantial knowledge from billions of text inputs [Bro+20a; Tao+23]. The in-context generation capabilities of these models are so convincing as to raise concern about their ability, "hallucinating" false responses that mislead human users [Ben+21]. As part of a careful data augmentation strategy, however, these hallucinations can be used effectively to generate captions that can significantly enrich the existing weak labels that annotate audio datasets.

The obvious concern is that such hallucinations could introduce even greater noise, especially when the original weak labels are vague. For example, a 10-second audio clip containing a multitude of sounds might be annotated with a single weak label of “speech.” Replacing such a weak label with LLM output conditional on a single-word prompt might simply swap out one source of noise for another. However, datasets such as AudioSet are sourced from videos, with visual information that can complement the audio. Video-to-Text (VTT) models can efficiently extract and transcribe visual cues into text representation, and state-of-the-art (SOTA) models such as BLIP2 [Li+23c] have demonstrated robust performance at this task. We apply BLIP2 to frames sampled from the video to generate a set of possible captions that capture visual information.

We have generated audio captions by prompting an LLM with the previously-available weak labels. Independently, we have also generated video captions using a VTT model. We once again leverage LLMs by computing entailment scores between these audio and visual captions to find the LLM “hallucinations” that are most compatible with the visual information. Additionally, we prompt the LLM to generate a merged audio-visual caption that provides an enriched annotation for the audio clip. Throughout this process, we remain particularly vigilant about potential audio-visual false positives: some audio clips do not have any corresponding visual information, for example when the objects seen on video do not produce sound or when the sounds’ sources are off-camera. We cover our approach to this issue in § 13.3. Our methodology so far is visualized in the left half of Figure 13.1, starting with the Raw Audio and Raw Video and leading up to the Audio+Visual (A+V) Caption.

Altogether, applying our methodology to AudioSet [Gem+17b] produces a dataset of over 2 million audio clips with significantly-enriched captions. We perform a human evaluation to verify the quality of our generated captions. We find that our captions are quantitatively better than previously-generated captions [Mei+23] and qualitatively comparable to human annotations released by AudioCAPs [Kim+19].

Having constructed a substantially enriched audio-text dataset, we can learn a powerful generative model for audio. We encode each audio clip into a post-quantization embedding space [Déf+22], and train a score-based latent diffusion model to reconstruct the audio, conditional on a T5[Raf+20] encoding of our generated captions. We delve deeper into our modeling choices and motivations in Section§ 13.4. Our entire system is illustrated in Figure 13.1.

Our experimental results reveal that our diffusion model outperforms baseline models such as AudioLDM [Liu+23; Yan+23] in generating higher-quality outputs. Additionally, we prove that this model can replicate all the supplemental capabilities of Stable Diffusion [Rom+22], including features like ControlNet[ZA23] and Dreambooth[Rui+22], among others[1].

To further validate the quality of both our generated captions and the trained diffusion model, we generate a large dataset of new audio samples and use it to train a classifier from

[1]Refer to the Appendix for more details of these add-on capabilities.

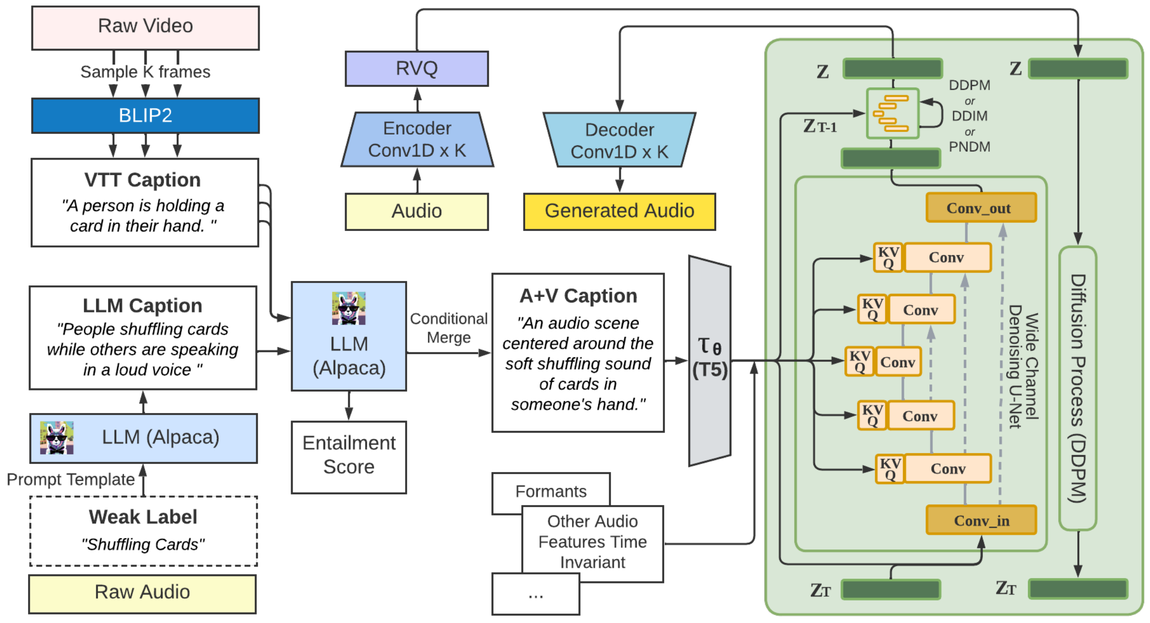


Figure 13.1: BLIP2:[Li+23c], T5:[Raf+20], Conv_in and Conv_out layers are modified to 128 channels. Audio Encoder, decoder and residual vector quantized (RVQ) layers are pre-trained by Encodec [Déf+22].

scratch. In this process, we identified Textual-inversion [Gal+22] as a much stronger tool compared to naive text-to-audio generation. Furthermore, we can also use our generated data to *supplement*, rather than replace, the existing AudioSet training data; we show that the combination of real and generated data results in improved SOTA classification accuracy.

At last, we show our diffusion model could be used as a natural denoiser, which perfectly fits into the randomized smoothing paradigm, and we show improved robustness using our diffusion as denoiser *versus* baseline randomized smoothing.

Our work, motivated by the need to augment low-resource audio datasets, makes the following contributions:

1. We release a new large-scale resource augmenting the existing AudioSet. This substantial increase not only enhances the volume of data available but also greatly diversifies it, providing a richer resource for future research and analysis.
2. We successfully train a diffusion model using a pre-trained latent encoder-decoder, bypassing the need to train a VAE and vocoder (e.g., HiFiGAN [KKB20]), which demonstrates excellent generation quality.
3. We showcase our diffusion model's ability to generate *useful* audio data. Classifiers trained with diffusion-generated data improve over those trained only on the original data.

4. We showcase our diffusion model's ability to denoise and improve the robustness of existing classifiers significantly.

13.2 Background & Related Works

Diffusion Models for Audio: Denoising Diffusion Probabilistic Models (Diffusion Models) [HJA20] are a class of score-based generative models to predict how a data point diffuses over set time steps. The motivation for these models is as follows: given an image and a known forward diffusion process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, defined in equation (13.1), model and predict the reverse diffusion process $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, defined in equation (13.2). Once trained, the reverse diffusion process can map random noise into new samples from the training data's distribution. While accurately modeling the proper probability density function (PDF) of a sufficiently complex dataset $P(X)$ is intractable, diffusion models instead model the gradient or stein score of the PDF: $\nabla_x \log P(X)$. Through integration, this score function conserves the information stored within the PDF without being intractable to compute [SE20], allowing for superior data coverage compared to other generative models. The forward process equation is as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \qquad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{13.1}$$

where β_t is a value from 0-1 retrieved from the noise scheduler. Diffusion models have excelled at tasks including image synthesis [DN21] and audio generation ([Liu+23; Yan+23]). In contrast to other generative models, diffusion models suffer from a significant drawback: the extended duration required for sampling. This happens because the iterative denoising process requires multiple steps instead of a single forward pass employed by GANs and VAEs for generation. Many modern diffusion models address this limitation by operating in the latent space of an autoencoder, significantly reducing the dimensionality required for generation [Rom+22]. This approach improves image quality while simultaneously lowering sampling and training time. The reverse diffusion equation is as follows:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{13.2}$$

Where $\mathbf{x}$ is the noised latent, and t is the timestep, p_θ is an unconditional denoising model to approximate conditional probabilities, this model is parameterized through a score estimator $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, which is equivalent to $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y = \emptyset)$. It's noteworthy that both p_θ and $\boldsymbol{\epsilon}_\theta$ can be effectively learned using a single neural network, in this work, we implement it with a high-channel UNet (more details in §13.4). Classifier-Free guidance is achieved by balancing the conditional and unconditional score estimator, where w is the tuning parameter, $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$:

$$\begin{aligned}\bar{\epsilon}_\theta(\mathbf{x}_t, t, y) &= \epsilon_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t}\; w\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\ &= (w+1)\epsilon_\theta(\mathbf{x}_t, t, y) - w\epsilon_\theta(\mathbf{x}_t, t)\end{aligned} \tag{13.3}$$

Several recent works have used latent diffusion models for audio generation. AudioLDM and DiffSound [Liu+23; Yan+23] generate audio by applying diffusion to spectrogram representations of sound. However, in addition to the denoising network, these approaches require training both a new VAE and an entirely separate vocoder (e.g., HiFi-GAN [KKB20]) to convert from the generated spectrograms back into waveforms. This requires significant engineering effort and may be difficult to reproduce or generalize to new domains (e.g., if FFT parameters for spectrograms are hard-coded). In this work, we dramatically reduce the engineering effort and GPU hours needed to train an audio diffusion model. Rather than training our own VAE and vocoder, we use Encodec [Déf+22], an off-the-shelf VQ-GAN model which has demonstrated competitive MUSHRA [Ser14] in high-fidelity audio generation. This allows us to focus all our training resources on the denoising U-Net. While using the pre-trained VQ-GAN prevents us from jointly learning the latent space and the diffusion model, our model is still able to adapt to the Encodec model's latent space. Our use of the Encodec model is similar to that of AudioGen [Kre+22], except they instead train an auto-regressive model. AudioGen also does not have public training code, making it a blackbox model and difficult to replicate.

Other Representation Learning Models: Directly modeling the score-based measure of the PDF allows diffusion models to significantly improve the diversity of their generation compared to other methods. For instance, GANs[KKB20] model the generative pipeline without expressly capturing the underlying data distribution; while this can be sufficient for high-quality generation, it cannot fully capture the diversity of the dataset. VAEs are better than GANs at capturing the diversity of the data distribution but often generate lower-quality samples [SE20]. Diffusion models generate samples that are of comparable or superior performance to GANs [DN21] while simultaneously producing better probability coverage of the underlying distribution [SE20]. Diffusion models' ability to generate diverse and high-quality samples makes them uniquely ideal for our goals of audio data augmentation.

Data Augmentation Methods: The conventional method of data augmentation in the audio domain involves utilizing basic transformations such as specAug [Par+19] and mixup [Zha+18] to create fresh audio by masking and mixing existing ones. However, these augmentations lack diversity in semantics. The primary benefit of using diffusion models for augmentation is that we can address the low-resource nature of audio data and generate diverse samples of similar quality to the original data, given the diffusion model's generation is high-quality and class-relevant. Recent work applying similar methods to the image domain suggests this is a scalable and accessible approach to data augmentation [Tra+23].

Automatic Caption Generation: Another common issue for audio datasets is the lack of

Classes	**LLM Generated Captions**
‘Singing’, ‘Yodeling’, ‘Speech’	‘A person singing and yodeling while talking.’
‘Pump (liquid)’, ‘Water’	‘The sound of a pump dispensing liquid and running water.’
‘Dog’, ‘Growling’, ‘Animal’	‘A dog growling and making animal sounds.’
’Frog’	’A frog is croaking in a dark, musty swamp.’

Table 13.1: Examples of conversions between class list and free text captions made to resemble image captions generated with an engineered LLM prompt.

high-quality captions. Other efforts, such as WavCaps [Mei+23] and AudioCaps [Kim+19], have taken various approaches to this challenge. AudioCaps employed human judges to create audio-text pairs for over 46 thousand samples taken from AudioSet, whereas WavCaps used ChatGPT to generate captions based on the weak labels resulting in a new dataset of approximately 400k samples. Both methods fail to scale effectively due to the often prohibitive cost of human judges and premium closed-source APIs.
Previous Audio Classification: Previous works[LQM+22] on audio classification achieve high accuracy by modeling $P(y|X)$ with a typical supervised learning approach. Self-supervised (SSL) methods such as AudioMAE [Poy+22] incorporate a model of $P(X)$ through pretraining, but such a model is inappropriate for the generation of new examples. Scored-based diffusion models model $P(X)$ directly and can use that understanding to directly perform classification. A properly trained diffusion model can function as a compressed knowledge base for the features provided in training [Li+23a].

13.3 Harnessing LLMs to generate Audio+Visual Captions: Prompt Engineering

Prompting given audio-only weak labels: We leverage the power of LLMs to increment the descriptiveness of the audio captions on datasets such as AudioSet[Gem+17b], which only contains weak labels without descriptive captions. We use Alpaca [Tao+23] and engineered prompts to generate a richer caption for every sample in AudioSet balanced and unbalanced sets, unifying the list of audio classes and introducing the relevant concepts. Table 13.1 shows specific examples of the class list to caption transformation. Alpaca is an open-source instruction-following model fine-tuned on the Llama-7b model [Tao+23]. Utilizing this model generates more grammatical captions that remain faithful to the original labels.

To generate text captions from class label lists, we used the following prompt: *"For each of these, summarize the sounds into a single sentence: \n describe a situation with all of these sounds together:"* followed by the clip's labels. A limitation of the Alpaca model is its tendency to add unnecessary details or ignore relevant labels when generating captions. By adding examples to the prompt, we leveraged the in-context learning ability of Alpaca to enrich our captions. Table A1 in the appendix covers more details.

Filter Hallucination and Obtain Visual Captions: Despite initial success with our approach, some captions contain Alpaca hallucinations, particularly in the cases of a single-class caption. For example, in Table 13.1 line 4, *"swamp"* is a hallucination, however plausible. To address this, we filter captions to replace single-class captions with a simpler non-LLM derived caption *"The sound of [CLASS]"*. This second pass does not invalidate the Alpaca-generated captions, which are far superior at capturing the complexity of audio samples with multiple classes. Recognizing the lack of detail in this single class captions, we utilized a SOTA Video-to-text model BLIP2 [Li+23c] to generate video-based captions for each video. These captions were derived from 3 sampled frames within the video at the $\frac{1}{3}$, $\frac{1}{2}$, and $\frac{3}{4}$ points of the 10-second clips. We again used Alpaca to combine these three captions into one with the following merging prompt: *"Create one sentence that summarizes these three simply:"*, allowing us to more effectively summarize the information of each video from the frames sampled.

Merging Audio and Visual Captions: These video captions were then combined with the audio captions from single-class labels with Alpaca to provide the needed visual context within each caption and combat the hallucination generated with single-label Alpaca audio captions. In our prompt, we specifically focused on the audio label while using the visual caption as auxiliary information: *"Summarize these two captions conditioned on the second caption, the second caption describes an audio class and is the main concept:"*. For all the prompts, we provided examples for Alpaca to better utilize the strength of in-context learning. The appendix shows more examples.

Audio-visual False Positive and Entailment Scoring: By observing the audio-visual captions generated while being aware of the audio-visual false-positive issue, we noticed that some of the captions in which the audio and video were not aligned resulted in noisy captions with details unrelated to audio in the actual clip. To measure the alignment between the visual caption and audio labels and test the capabilities of a decoder-only language model, we created an entailment score Alpaca prompt: *"on a scale from 0 to 1, output the probability that these two captions happen together in float format:"* In our in-context examples, we emphasized similar concepts even if the exact audio label was not referred to, such as an "ukelele" in audio and a "mandolin" in the video receiving a high entailment score. Nonetheless, we maintain the integrity of the video caption by ensuring it did not introduce sounds not present in the audio label, such as the implication of speech through the depiction of a person when the audio label did not include speech.

We sought to investigate the fidelity of our AV entailment score and overall usage of decoder-only LMs to calculate textual similarity and other metrics. In parallel, we use T5-sentence encoder [Raf+20] and the BERT sentence encoder and computed the embedding similarities between the audio and video captions. The Pearson correlation was then computed between the Alpaca entailment score and the scores from both T5 and BERT encoders to get 0.13 for T5 and 0.14 for BERT. We also calculated the (min, median, max) score for each metric with Alpaca having (0, 0.55, 1), T5 having (0.60, 0.75, 0.98), and BERT having (−0.20, 0.34, 0.98).

These scores are not correlated which implies that a decoder LM may be less reliable than an encoder LM when determining textual similarity and calculating metrics. Despite this, using a caption similarity metric could ensure that future audio-visual training data is less noisy and has clearer relationships between different modalities and be used to guide the merging of audio-visual captions to determine when the visual context is useful to include in our caption.

Evaluation of Caption Generation: AudioSet's label coarseness and class imbalance are mitigated by our application of multiple Alpaca LLM caption generations, yielding 2.2M detailed audio clip captions. [2] Previous efforts on creating audio captions focused on automatic audio captioning[Mei+23], but with the small training set size and imbalanced classes within the dataset § 13.7, the performance of their model is also limited. These are also typically end-to-end with WavCaps[Mei+23] using an HTSAT-BART model [Mei+23] which lacks the explainability and scalability compared to our human language-focused approach with weak labeling. Our approach better considers different modalities and introduces more flexibility in label generation due to the controlled hallucination that Alpaca has.

To assess the performance of our captions and the improvements due to additional context provided in their generation, we analyzed a subset of AudioCaps [Kim+19] captions (human-generated captions) and their corresponding audio clips against our generated captions, scoring each on a scale from 0 to 1 on the similarity to AudioCaps[Kim+19] while referencing the actual audio clip as shown in Table 13.2. Since most automatic metrics are based on n-gram similarity or LCS [3] and generally do not perform as well on individual sentence comparisons [Awa+22], we decided to use a human metric because of the large vocabulary variation as shown in the subset vocabulary size shown in Table 13.2. Additionally, the WavCaps [Mei+23] model is fine-tuned on AudioCaps which helps boost their automatic metric scores in comparison to our approach.

These results clearly display the qualitative improvements gained from in-context prompting for LLMs with clear examples to guide text generation as well as the addition

[2]We will provide more details in the appendix and will release all captions as open-source resources.

[3]Longest common subsequence (LCS) and comparison of n-gram overlaps perform poorly with vastly different vocabularies and description styles as it is not able to capture semantic similarity well or match with completely different caption structures and scenarios

Prompt Context	**Similarity**	**BLEU$_1$**	**BLEU$_4$**	**METEOR**	**ROUGE$_l$**	**CIDEr**	**Vocabulary**
Zero-Shot	0.474	0.128	0.006	0.079	0.128	0.094	-
One-Shot	0.605	0.178	0.014	0.100	0.182	0.193	-
Few-Shot	0.686	0.188	0.016	0.168	0.229	0.242	-
Audio-Visual Merge	**0.750**	0.165	0.013	0.109	0.178	0.183	**1480**
WavCaps[Mei+23]	0.667	**0.231**	**0.056**	**0.136**	**0.277**	**0.682**	509
AudioCaps[Kim+19]	N/A	N/A	N/A	N/A	N/A	N/A	871

Table 13.2: Average similarity scores ranging from 0.0 - 1.0 between captions generated with Alpaca prompts and ground truth captions (AudioCaps) where zero-shot means no examples given to Alpaca, one-shot is one example, and few-shot is several examples, automatic evaluation metrics compared to the ground truth, and vocabulary size for the sample captions generated.

of visual elements into the captions. One limitation is the absence of labels in the original AudioSet which the human judges mentioned in the AudioCaps captions. Typically this occurred with speech and wind sounds being present in AudioCaps but not AudioSet labels. However, these cases show the benefits of using LLMs for generation which was able to use context clues and natural language understanding to add these missing features. An example of this context-aware enrichment can be seen in Figure 13.1 where the caption "shuffling cards" is correctly extended to include a human in the scene. Similarly, we observed that Alpaca would hallucinate "at the park" or similar setting-specific details for audio samples weakly labeled with ducks, water, or speech. Such hallucinations are often correct, but even when false they encode relevant domain knowledge that helps improve the quality of our captions.

13.4 Text-guided Diffusion in Quantized Latent Space

Text Encoder τ_θ: We experimented with several text encoders for the prompt conditioned generation including CLIP [Rad+21], CLAP [Wu*+23], and T5 [Raf+20]. The CLIP encoder we used to fine-tune our U-Net models complicated the domain shift from image to audio. While we initially used CLAP for its textual-audio joint embedding, we found it performed worse than T5. T5 has a larger embedding space than CLAP or CLIP, requiring an additional linear projection to connect it to the U-Net. We found this detail crucial to changing the text encoder while preserving pretraining knowledge. The second strength of T5's larger embedding space is its ability to encode larger vocabularies. As displayed in Table 13.2, our LLM-generated captions' vocabulary is significantly larger than that of either WavCaps

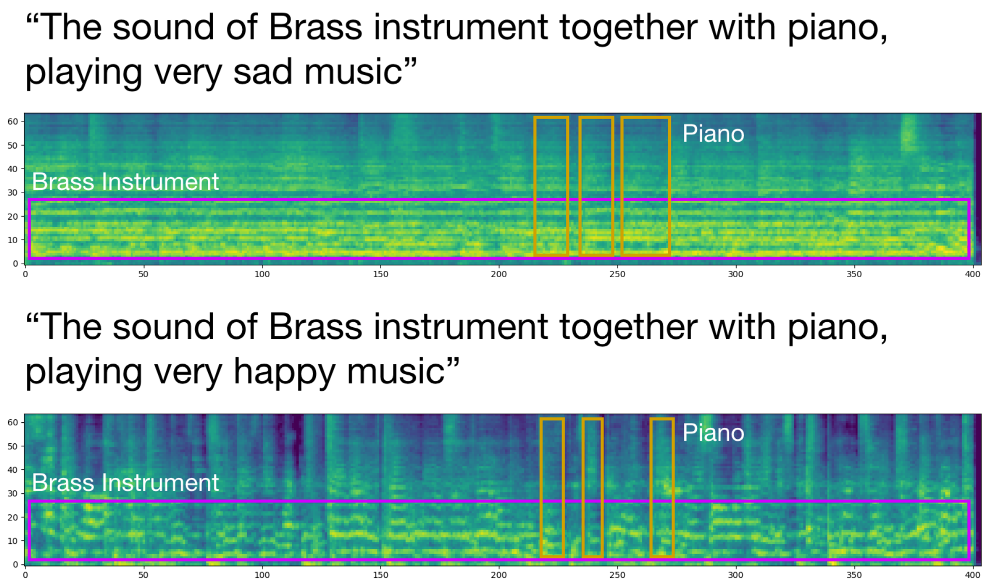


Figure 13.2: Spectrogram (only for visualization, never used) of sample generations showing tonal variance and harmonics.

or AudioCaps. To handle this more extensive caption vocabulary without loss, we chose T5 as our pipeline's text encoder. The final consideration for text encoding is using an attention mask on the text embedding. These encoders output a fixed-length embedding along with an attention mask usable to ignore invalid tokens. We experimented with and without attention masks with varying results. Experimentally, as shown in Table 13.4, masking had different effects on CLAP and T5-based models. Intuition would say that masking the T5 embedding would yield a more significant improvement as its fixed length is larger than CLAP; however, the addition of the linear projection layer between the text embedding and the U-Net functions as a type of masking resulting in inferior performance when combined with an attention mask, as was also discussed in [Rom+22] as “unmasked" expert model.

Noise schedule: The denoising process is trained across a fixed number T of DDPM [HJA20] steps. A naive approach would be to simply compute noise percent as a linear (scaled) interpolation from 0 - 1 across the timesteps to control the β_t in Eq. 13.1, as done in AudioLDM [Liu+23]. With recent work [Che23] showing the superior performance of non-linear schedule *versus* linear schedule, we adopt cosine noise scheduling without the need to tune the $\beta_{start}, \beta_{end}$ from $t = 0$ to $t = T$: $\beta_t = \text{Clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999)$ $\quad \bar{\alpha}_t = \frac{f(t)}{f(0)}$ where $f(t) = \cos\left(\frac{\frac{t}{T}+\delta}{1+\delta} \cdot \frac{\pi}{2}\right)^2$, where δ is a small offset. Figure 13.3 illustrates the process. During the sampling steps, we explored the usages of DDPM [HJA20], and accelerated approximating schedulers: DDIM[SME20], and PNDM [Liu+22]. Our empirical

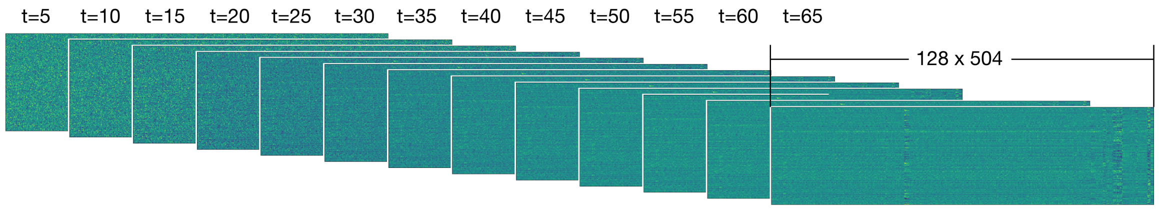


Figure 13.3: Latent space representations created throughout the denoising process. These images notably display the lack of interpretability in our generation space.

results show the best speed-quality tradeoff for PNDM [Liu+22], thus the results reported in the main paper are PNDM results. In the appendix, we include a detailed comparison of all noise schedulers.

U-Net ϵ_θ Design: We refer to our U-Net as a Wide Channel U-Net due to our choice to train and generate in a 128-channel latent space instead of the typical one or three channels used in SOTA audio generation. We had two main observations that informed this decision: first, the receptive field of the U-Net convolutional blocks could not fully explore the 128×504 latent space representations from the Encodec encoder; second, the latent encoding showed little variance within the 128 dimensions. We were able to leverage the second observation to correct the first by reshaping the latent vectors from a one-channel 128×512 image to a 128-channel 21×24 image. We then normalized each channel to a mean of zero and std of one representation to assist the U-Net in learning the noise: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With this new representation, the convolutional blocks are able to contain the entire image in their receptive field without losing resolution and result in higher fidelity audio. After generation, these transformations can be fully inverted to allow for decoding back into a waveform.

Another difference between our diffusion approach and that of past work (e.g., AudioLDM [Liu+23]) is our use of cross-attention instead of embedding adding. This change enables us to conserve text embedding features. In self-attention, the text embedding is first concatenated to the image embedding, subjecting it to modifications at each layer of the U-Net. For cross-attention, we instead use the unmodified text for attention at each layer of the U-Net, maintaining the text embedding fidelity throughout generation and improving class guidance. Equation 13.4 shows how this attention mechanism functions. In our cross-attention, the text conditioning y remains unchanged between diffusion steps; in self-attention, y would first be incorporated into the noised latent z_i before diffusion begins. In the self-attention setting, as the noisy latent passes through the U-Net the text embedding becomes increasingly interwoven into the noised latent and loses its reliability.[4]

[4]See the appendix for a more detailed explanation, including specific dimensions, and potential pitfalls

$$\begin{aligned}&\text{Cross-Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\cdot\mathbf{V}\\&\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)}\cdot\varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)}\cdot\tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)}\cdot\tau_\theta(y)\end{aligned} \tag{13.4}$$

Generation Latent Space: The Encodec model [Déf+22] we selected consists of an encoder, vector quantizer, and decoder stages. Initially, we attempted to directly learn the discrete "codebook" of RVQ as this has the highest degree of compression, at only 8×504, and could leverage the generative benefits of the Encodec codebook and decoder stage. However, during experimentation, we observed nearly 0 decreases in train loss over time, as diffusion is only suitable for continuous vector space [SE20], an AutoRegressive model might be better for this. We trained our next model on the decoder embedding, which is of larger size, at 128×504, but is continuous. This slight change improved training substantially and resulted in the model contributions we are providing. Despite the discrete representation's inferior performance, we pre-computed the entirety of AudioSet 2M into discrete vectors and saved these new compressed versions to disk for training. This slight change from reading raw audio files to reading compact discrete codes from disk substantially accelerated training for two reasons. First, the I/O read times became significantly shorter as the files consist of 8×504 features instead of $160,000 \times 1$, resulting in a complete copy of AudioSet 2M that only takes 63 GB compared to 1.4 TB before for a $> 95\%$ reduction. Second, without needing to store these large waveforms in memory, we increased our batch sizes significantly, greatly improving training time. These results reinforce the vision of the original LDM in that it shows diffusion models can learn and generate arbitrary latent spaces regardless of their human interpretability or structure.

13.5 Diffusion as Data Augmentation for better Generalization

Using a pre-trained text-to-audio diffusion model, following [Tra+23], we leveraged Textual Inversion [Gal+22] to generate over 80,000 new audio samples randomly divided among the 527 audio classes in the AudioSet-20K balanced set. A random value $k \in [1, 2, 3]$ was selected and mapped to k random classes from the AudioSet-20K class list for each sample. With these samples generated, we combined them into the datasets shown in Table 13.3. Due to the unbalanced distribution of AudioSet labels, we adopted the balanced sampling strategy as mentioned in Chapter 6. Note here, the weights for the lowest performing 100 classes are amplified as a control set for the experiment. When mixing real and synthetic data, we adopt the same strategy as [Tra+23]:

$$
\begin{gathered}
i \sim \mathcal{U}(\{1, \ldots, N\}), \quad j \sim \mathcal{U}(\{1, \ldots, M\}) \\
B_{l+1} \leftarrow B_l \cup \begin{cases} X_i & \text{with probability } (1-\alpha) \\ \tilde{X}_{ij} & \text{otherwise} \end{cases}
\end{gathered}
\tag{13.5}
$$

Note here $X \in \mathbb{R}^{N \times H \times W \times 1}$ denotes a dataset of N real audio latents, and $i \in \mathbb{Z}$ specifies the index of a particular latent X_i. For each latent, we generate M augmentations, resulting in a synthetic dataset $\tilde{X} \in \mathbb{R}^{N \times M \times H \times W \times 1}$ with $N \times M$ augmentations, where $\tilde{X}_{ij} \in \mathbb{R}^{H \times W \times 3}$ enumerates the jth augmentation for the ith latent in the dataset. Indices i and j are sampled uniformly from the available N real audio latents and their M augmented versions respectively. Given indices ij, with probability $(1-\alpha)$ a real audio latent X_i is added to the batch B, otherwise its augmented latent $\tilde{X}_{ij}$ is added.

To closely match the original AS-20K dataset, these new samples do not have LLM-generated captions, the naive approach is to generate them with a simple prompt formatted as "The sound of [LIST OF AUDIO CLASSES]." This configuration was used in [Liu+23] to convert labels to captions and provide prompts. We believe it is suboptimal for data augmentation, since everything needs to start generating from random Gaussian noise, and it is trading class accuracy off for diversity. If the class is not accurate, diversity in this case will be in vain. Therefore, we use Textual Inversion [Gal+22; Tra+23] to update the token of the class name of each of the AudioSet classes, and fine-tune their embeddings. At the generation step, we use the same prompt "The sound of [LIST OF AUDIO CLASSES]." but perform image-to-image diffusion, so that our denoising U-Net does not have to denoise all the way from pure Gaussian noise, but from a noisy version of ground-truth latent.

As shown in Table 13.3, we can see the classification models trained on augmented audio datasets improve with the growing size of the dataset. These results display the value of additional diffusion-generated samples as a form of data augmentation.

13.6 Diffusion As Denoiser for better robustness

Randomized smoothing is a certified defense that converts any given classifier f into a new smoothed classifier g that is characterized by a non-linear Lipschitz property [CRK19]. When queried at a point x, the smoothed classifier g outputs the class that is most likely to be returned by f under isotropic Gaussian perturbations of its inputs.

$$
g(x) = \arg\max_{c \in Y} \mathbb{P}[f(x+\delta) = c] \quad \text{where} \quad \delta \sim \mathcal{N}(0, \sigma^2 I). \tag{13.6}
$$

As we briefly covered in Chapter 12, Randomized Smoothing procedure is as follows: suppose that when the base classifier f classifies $N(x, \sigma^2 I)$, the class c_A is returned with probability $p_A = \mathbb{P}[f(x+\delta) = c_A]$, and the "runner-up" class c_B is returned with probability $p_B = \max_{c \neq c_A} \mathbb{P}[f(x+\delta) = c]$. The smoothed classifier g is robust around x within the radius

$$R = \frac{\sigma}{2}\left(\Phi^{-1}(p_A) - \Phi^{-1}(p_B)\right), \tag{13.7}$$

where Φ^{-1} is the inverse of the standard Gaussian CDF. When f is a deep neural network, computing p_A and p_B accurately is not practical. To alleviate this problem, [CRK19] used Monte Carlo sampling to estimate some $\underline{p_A}$ and $\overline{p}_B$ such that $p_A \leq \underline{p_A}$ and $p_B \geq \overline{p}_B$ with arbitrarily high probability. The certified radius is then computed by replacing p_A, p_B with $\underline{p_A}, \overline{p}_B$.

Denoised smoothing [Sal+20] is an instantiation of randomized smoothing [CRK19], where the base classifier f is composed of a denoiser denoise followed by a standard classifier f_{clf}:

$$f(x+\delta) := f_{\text{clf}}(\text{denoise}(x+\delta)). \tag{13.8}$$

Given a very good denoiser (i.e., $\text{denoise}(x+\delta) \approx x$ with high probability for $\delta \sim \mathcal{N}(0, \sigma^2 I)$), we can expect the base classifier's accuracy on noisy images to be similar to the clean accuracy of the standard classifier f_{clf}. [Sal+20] instantiate their denoised smoothing technique by training custom denoiser models with Gaussian noise augmentation, combined with off-the-shelf pre-trained classifiers.

Given these contexts, our audio-journey diffusion model can easily play the role of this good denoiser, similar to what was covered in [CTK+22]. The entire denoising smoothing as a defense for noise/ adversarial noise is shown in Figure 13.4. This way of leveraging pre-trained diffusion models are appearing to be a very promising strategy in speech enhancement [Kon+20; Lu+22], although without a robust certificate. Inspired by [Ban+22], we identified a striking similarity between the cold-diffusion paradigm *versus* our AudioMAE as covered in Chapter 7. Instead of using Gaussian noise, our AudioMAE uses a random masking strategy developed in Chapter 7, this is perfectly inline with the definition of Cold Diffusion [Ban+22], with AudioMAE being a special case of 1-stepped cold diffusion. We expect this masking to mimic signal packet drop in network communication.

$$\min_{\theta} \mathbb{E}_{x \sim X} \|R_\theta(D(x,t),t) - x\|_k \tag{13.9}$$

where x denotes a random image sampled from distribution X and $\|\cdot\|$ denotes a norm. R_θ is a restoration operator that inversely (or approximately inversely) applies D. This operator possesses the property that $R(x_t, t) \approx x_0$. Given an image $x_0 \in \mathbb{R}^N$, we denote the degradation of x_0 by operator D with severity t as $x_t = D(x_0, t)$. The output distribution $D(x_0, t)$ of the degradation should be a continuous function of t, and the operator should satisfy the condition $D(x_0, 0) = x_0$. Following this definition, we can see that our AudioMAE is a cold-diffusion special case where our $t = 1$.

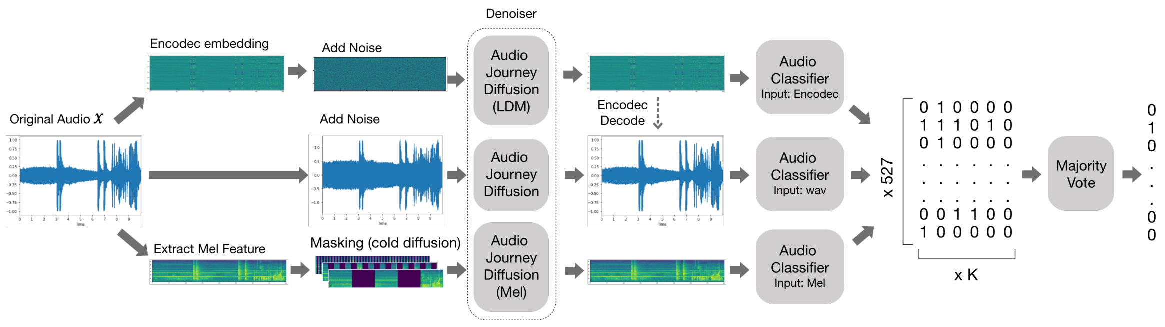


Figure 13.4: Diffusion as denoised smoothing, we leveraged 3 different types of our audio-journey model to defend against noises including adversarial perturbation, colored noise, and occlusion.

13.7 Experiment and Results

Datasets:

We follow the same setup of AudioSet as Chapter 12. The primary differentiating factor between our method and those of other SOTA audio generation papers is our limited dataset. We trained our model exclusively on AudioSet without Alpaca-generated captions. These results show the impressive capabilities of generation inside the latent space of the Encodec [Déf+22] model and our Alpaca captions.

As is noted in Chapter 6, we paid special attention to not over-sample majority classes by using a specialized cutMix [Yun+19] strategy sampling from the class distribution (multinomial). Since we could not merge captions with probability, we concatenate the captions using the keyword "followed by", and only allow mixing each weakly label with other strongly-labeled segments. This strategy significantly boosts data quality. On the other hand, we find that the naive mixup[Zha+18] strategy notably impairs generation performance. The process of mixing up samples creates noisy instances, leading the model to potentially learn unintelligible sounds.

Classification Accuracy after Diffusion Augmentation: Table 13.3 lists classifier performance when trained on our diffusion-augmented datasets showing a clear image that additional samples generated with diffusion measurably improve classifier accuracy. Due to the large number of parameters of AST, it struggles to train from scratch, diffusion augmentation visibly alleviated this training difficulty and complemented pretraining. The most significant improvement comes from augmenting AudioSet-20K with an extra 20K generated samples, and the benefits slowly attenuate with more examples. Nonetheless, it does yield measurable improvements when used as augmentation, especially when used in conjunction with other augmentation methods. As is shown in Figure 13.5. We see more clearly the class-wise mAP changes based on our CNNtrans model trained on a balanced AS-20k dataset. As we can see, diffusion data cannot entirely replace ground

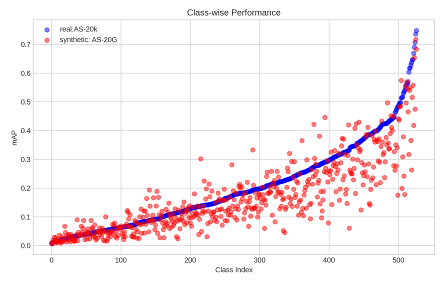
	Model	PT	Aug	AS-20kG	AS-20k	+20kG	+40kG	+60kG	+80kG	AS-2M
PANNs [Kon+19a]	CNN	-	-	-	22.1	-	-	-	-	37.5
PANNs [Kon+19a]	CNN	-	mx+sp	-	27.8	-	-	-	-	43.1
TALtrans [LQM+22]	CNN+T	-	-	-	22.4	-	-	-	-	38.3
TALtrans [LQM+22]	CNN+T	-	mx+sp	-	28.0	-	-	-	-	43.7
Our TALtrans	CNN+T	-	-	$10.1_{\pm.50}$	$22.4_{\pm.16}$	$25.8_{\pm.11}$	$27.0_{\pm.13}$	$28.1_{\pm.06}$	$30.1_{\pm.03}$	$38.3_{\pm.15}$
Our TALtrans	CNN+T	-	mx+sp	$11.2_{\pm.40}$	$28.0_{\pm.20}$	$29.4_{\pm.31}$	$29.5_{\pm.13}$	$30.7_{\pm.21}$	$32.3_{\pm.14}$	$43.7_{\pm.25}$
AST [GCG21a]	DeiT	-	-	-	14.8	-	-	-	-	36.6
AST [GCG21a]	DeiT	IM	mx+sp	-	34.7	-	-	-	-	45.9
Our AST	DeiT	-	-	$3.2_{\pm.20}$	$14.8_{\pm.17}$	$15.4_{\pm.22}$	$16.7_{\pm.14}$	$18.1_{\pm.01}$	$20.2_{\pm.18}$	$34.6_{\pm.20}$
Our AST	DeiT	-	mx+sp	$8.2_{\pm.61}$	$16.9_{\pm.12}$	$18.4_{\pm.32}$	$19.5_{\pm.12}$	$20.7_{\pm.22}$	$22.4_{\pm.31}$	$37.6_{\pm.10}$
Our AST	DeiT	IM	mx+sp	$13.5_{\pm.50}$	$34.7_{\pm.77}$	$35.1_{\pm.18}$	$36.1_{\pm.42}$	$36.9_{\pm.03}$	$37.5_{\pm.12}$	$45.4_{\pm.70}$

Table 13.3: All datasets, other than AS-20k and AS-2M, are based from AS-20k with diffusion augmentation to add samples, details in section § 13.4 "G" symbol indicates it's synthetic. The metric is mean average precision (mAP). For pretraining (PT) dataset, AS:AudioSet, and IM:ImageNet. For augmentation (aug), mx+sp:mixup[Zha+18] and SpecAug[Par+19]. Generation model and classification accuracy for each augmented dataset showing the improvements measured from diffusion as augmentation. Dataset AS-20kG consists exclusively of generated samples with 0 real samples from AudioSet. Our TALtrans Model (CNN+T: CNN+Transformer) has 12.1M params, and our AST model (DeiT/ViT-B) has 88M params.

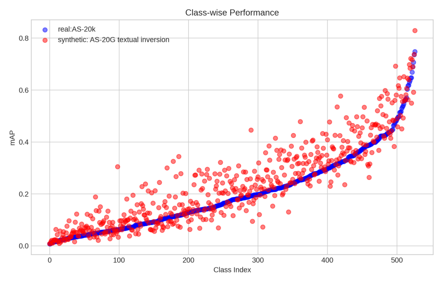
truth data, as demonstrated by the inferior scores for AS-20kG, and naive text-to-image, using Textual-inversion [Tra+23] completely closes and even surpasses the performance gap. We could also see a clear boost in class-wise mAP by mixing real data together with synthetic data. It is interesting to see that we could improve the accuracy of stubbornly low-performing classes, as is shown in Fig 13.5e, however, this is at the cost of trading off performance for other classes. This echoes our advocation for evaluation of class-wise robustness in chapter 10.

Generation Quality:

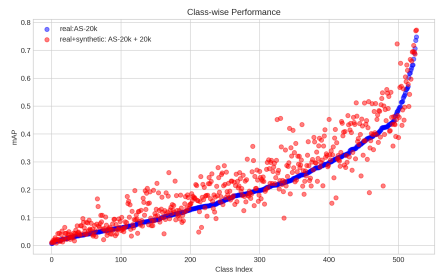
Table 13.4 presents interesting quantitative comparisons between our models and previous SOTA models. We performed our evaluation similarly to AudioLDM [Liu+23]; first, we extracted all captions from the AudioCaps [Kim+19] test set and generated samples based on each of these captions. We then compare FD scores against the ground truth audio from the AudioCaps [Kim+19] test set for each model, IS and KL scores are similarly



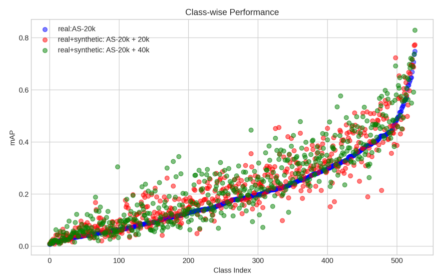
(a) AS20k real data VS 20k synthetic data (naive text-to-audio)



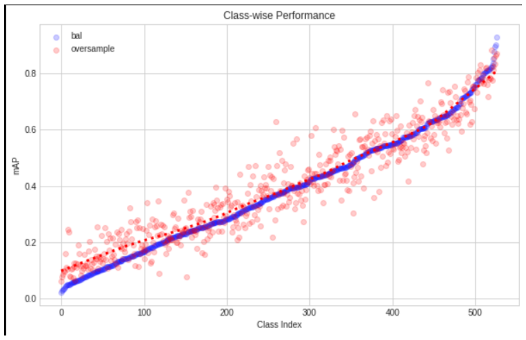
(b) AS20k VS 20k synthetic data using textual inversion[Tra+23]



(c) AS20k (real) VS AS20k+20k (real+synthetic)



(d) AS20k (real) VS AS20k+20k (real+synthetic) VS AS20k+40k (real+synthetic)



(e) AS20k (real) VS AS20k+20k (real+synthetic) oversampling lowest performing 100 classes

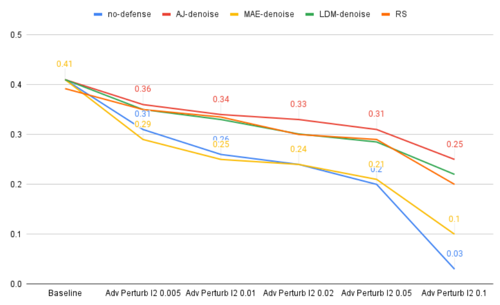
Figure 13.5: Blue line is the class-wise mAP of CNNtrans model trained on AS-20k balanced set, class-ids sorted by performance. *versus* classifiers trained on pure synthetic data or real+synthetic data.

Model	Datasets	Text	Params	FD	IS	KL
DiffSound [Yan+23]	AS+AC	✓	400M	47.68	4.01	7.76
AudioGen [Kre+22]	AS+AC+8	✓	285M	-	-	2.09
AudioLDM-L-Full [Liu+23]	AS+AC+2	×	739M	23.32	8.13	1.59
Audio Journey-CLAP	AS	✓	861M	67.6	1.63±.02	**0.127**
Audio Journey-CLAP-masked	AS	✓	861M	55.5	1.64±.02	0.134
Audio Journey-T5-masked	AS	✓	861M	13.14	1.64±.03	0.209
Audio Journey-T5	AS	✓	861M	**12.09**	**1.64**±.03	0.259

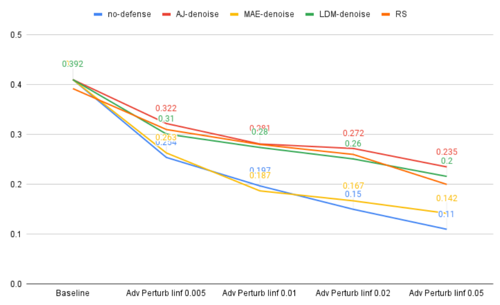
Table 13.4: These models are scored on frechet distance (FD), inception score (IS), and kullback–Leibler divergence (KL). Our scores are computed by comparison against Audio-Caps [Kim+19] test set explained further in § 13.7. Scores for DiffSound, AudioGen, and AudioLDM are from [Liu+23].

measured. This shows two noteworthy trends: first, our generative model holds up to current SOTA models despite training exclusively on AudioSet with Alpaca-generated captions, whereas previous SOTA works include multiple other datasets. Second, the inverse trends of our FD *versus* KL scores imply a trade-off between quality and diversity. This intuition is reflected in the models with said scores. CLAP model's superior KL scores are a reflection of the similarity between CLAP and CLIP, which these models were pretrained on. The T5-based model's superior FD scores imply T5 assists in generation more than CLAP despite lower variance.

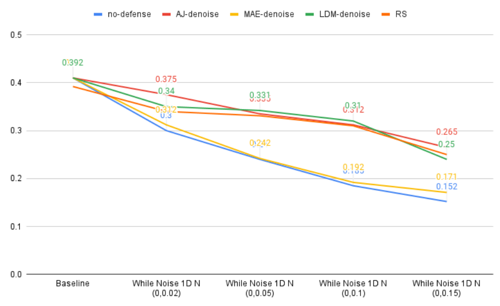
Classification Accuracy after Diffusion Denoising: Using the precise robustness metrics outlined in Chapter 12, we evaluate the impact of our diffusion-denoising process on robustness. Note here, we adopt the same one-shot denoising strategy as is used in [CTK+22], as we also observe degraded performance with more denoising steps as the diffusion model tends to hallucinate more with more denoising steps. We also trained a model with Gaussian noise added following Randomized smoothing [CRK19] as the comparison for defense. The specific results are shown in Figure 13.6. Overall, the different diffusion-denoising strategies offer a non-trivial amount of protection *versus* baseline with no defense. Notably, raw-wave diffusion offers the best protection against Gaussian noise and adversarial perturbation. MAE-denoising, despite offering not much protection under Gaussian noise or adversarial perturbation, shows a strong capability to defend against occlusion. LDM-denoising is much better than MAE-denoise in defending against Gaussian and adversarial noise but underperforms RS, showing us that the domain mismatching from Latent to Wav would decrease the denoising capability of diffusion models. Overall, these results show a clear trend that whatever noise the diffusion model is trained with, it would be best at defending.



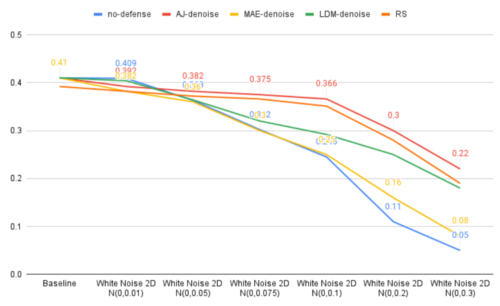
(a) Protection against ℓ_2 adversarial perturbation



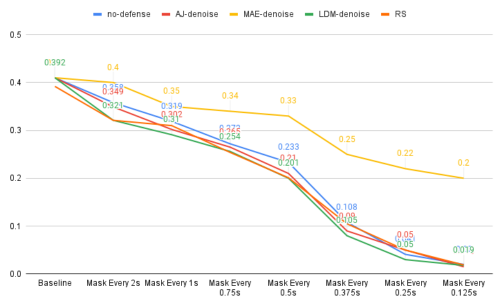
(b) Protection against ℓ_∞ adversarial perturbation



(c) Protection against Gaussian Noise added on Waveforms



(d) Protection against Gaussian Noise added on 2d features



(e) Protection against occlusion

Figure 13.6: Diffusion Denoising's protection against different kinds of common corruption as listed in Chapter 12. Here RS- Randomized smoothing baseline, AJ-denoise corresponds to Fig13.4's middle pipeline wav-to-wav denoise; MAE-denoise corresponds to Fig13.4's lower pipeline Mel-to-Mel denoise; LDM-denoise corresponds to Fig13.4's upper pipeline Latent-to-Latent denoise;

Chapter 14

Thesis Conclusion and Contribution

To sum up, Chapters 2,3,4 listed our findings about how to make CNN-family models perform well on the AED (audio-only) task:[1]

1. Deep learning performs better with larger features (9% gain) on the DCASE2016 dataset [MHV16b], as observed in Chapter 2.3;

2. Fully Convolution layers beat Fully Connected layers with more accuracy (+2%), and are more +95% efficient to train on the UrbanSound8k [SJB14a] dataset. This is covered in the discussion about Table 3.5 in Chapter 3;

3. Choosing the **proper kernel size** is essential for raw wave input. As evidenced in § 3.4, a deeper network is not always better;

4. Normalization is crucial for training deep CNNs, this is observed in Table.3.6;

5. Weak labels can be tackled with **pooling functions**, and max pooling tends to be the least ideal one, as is evident in a comparison of several pooling functions in § 4.2.2 using AudioSet and DCASE2017[WLM18] subset.

Our findings on smaller datasets (DCASE 2016,2017) were published at ICASSP 2017 and InterSpeech 2018: [Li+17; Dai+17; Tse+18] The study done on a larger-scale dataset was published at ICASSP 2019 [WLM19].

Chapter 6,7 documented our comprehensive studies on the Transformer-family models on the AED (audio-only) tasks, and how they perform against CNNs in performance. All our findings in these 2 chapters are observed for models trained on AudioSet [Gem+17b]:

[1]Key findings are emphasized with bold fonts for ease of identification. For enhanced readability, the list of publications, which was initially presented in Chapter 1, is reiterated here in small font.

1. Smaller features only lose 2% VS. large features, but are **6x more efficient** to train. This efficiency is clearly demonstrated through our extensive empirical comparison detailed in §6.4, which could help researchers/practitioners speed up their test cycle significantly;

2. Models with **local+global** information strike a balance between accuracy and efficiency. This is demonstrated by the competitive performance shown by CNN+Transformers, and CRNNs in Table 6.2 with fewer parameters. Also, it is evident in the discussion around Figure 7.3 showing how SwinTransformer [Liu+21](local+global attention) is less prone to create “ghost samples" compared to ViT(global attention).

3. Transformer family models have more number of parameters VS. CNN family models, and are more difficult/computationally expensive/ to train, and they are more sensitive to hyper-parameters, as is observed in results in Table 6.2.

4. **Pretraining** is **not always** necessary, its effect is mainly to warm start the training of a model. It is only **necessary for heavily-parameterized models** in our experiments. This is discussed in §6.5;

5. **Modern training techniques** e.g. (large batch size, LR decay, etc) very crucial for high accuracy, Learning Rate is the most crucial hyper-parameter, as is detailed in §6.6;

6. **Data augmentation and balancing** techniques are shown to be always helpful which on average brings about 2% performance gain, as is discussed in §6.7;

7. **Self-supervised learning** is demonstrating its potential to replace supervised learning, particularly in its promising ability to scale to larger, unlabeled datasets. In the context of audio self-supervised learning (SSL), reconstruction has empirically proven to be an effective pretext task, showcasing strong performance. As is shown in Table 7.2, AudioMAE is the best-performing model currently.

8. Since CNNs have limitations like 0-mask error, whereas Transformers are more versatile, we are seeing that the Transformer family is eclipsing CNNs in the SSL paradigm, as is discussed in §7.3.

Our findings about the Transformer families were published at InterSpeech 2022 [LQM+22] and NeurIPS 2022 [Poy+22].

Chapter 5 conducted on DCASE2017[Li+18] dataset addresses the question of whether incorporating visual cues could enhance the accuracy of an audio-only pipeline. The findings indicate that the inclusion of visual information doesn’t invariably contribute to improved performance. Rather, its usefulness is primarily seen in visually dominant tasks, and in certain cases, it might even negatively impact performance.

These research findings were published in ICASSP 2018 [Li+18].

All the chapters above provide a comprehensive study of **the best practice** for building an AED system that performs well focusing on mostly audio-only systems, and thus fulfills **the first goal** of the thesis, concluding Part I.

Chapter 8 showed real-world threats from adversarial perturbation in both audio and visual modalities without the need of tampering with the subject of interest.

These findings were published at ICML 2019[LSK19] and NeurIPS 2019[Li+19].

The aforementioned factors significantly inspired **the second objective** of this thesis, which seeks to delve into the **robustness** of machine learning systems within both unimodal and multimodal contexts.

Chapter 9 showed how fusing audio could enable us to build a more robust joint-embedding space (3% gain VS. no-audio), as is evident in Table 9.1 for the VTT model on the MSR-VTT dataset [Xu+16]. This chapter also shows how we could improve multimodal fusion by modifying the pairwise ranking loss (1% on MSRVTT and 30% gain on MSVD). §9.4.3.

Our findings were published at ICMR 2018 (Best Paper Award)[Mit+18a] and IJMIR 2019[Mit+19].

Chapter 10,11 is based on multimodal systems (audio+visual), and delves into finding the key factors of robustness for multimodal systems, our findings include:

1. **Multimodal** Models are **not necessarily more robust**, as is theoretically proved in § B.1, and is also empirically shown in §11.7.3.

2. **Prominent factors** that affect adversarial robustness: Late fusion is more robust than Early Fusion (§10.5.1); More training data makes the trained model more robust, and more structures in the data(sound) class makes the class more robust;(§10.5.4); lower frequency bands contribute more to robustness in sounds (§10.5.2);

3. Our proposed **density and convexity** metric could effectively measure the robustness of models on a certain class of data, as is evident in the linear correlation in Figure 11.4 and the discussion around it. While the majority of existing research primarily concentrates on point-wise robustness, our work trailblazes the exploration into classwise robustness;

4. We showed density and convexity-driven **mixup as a cheaper and effective alternative** to adversarial training (AT), as it can offer comparable protection against attack and is cheaper than AT, as detailed in §11.7.3.

These findings were published at ICASSP 2021 [Li+21] and ICASSP 2022 [Li+22b] (Best Student Paper Award).

Chapter 12 discussed the relationship between models' noise robustness under common corruption *versus* adversarial robustness, where we find:

1. **ViT architecture is comparatively more robust** compared to 3 other CNN-based architectures against both colored and adversarial noises. This is shown in § 12.2, as is discussed in § 12.2.1 This is possibly due to their finer-grain learning units and global receptive field versus CNN's fixed and limited receptive field size.

2. Our sharpness measurements indicate that while possessing flatter sharp minima tends to be a common trait among robust models, it isn't a sufficient condition for robustness. It is currently **not feasible** to make a fair comparison across architectures based on ϵ-sharpness or λ_{max}, as detailed in § 12.4.

3. Robustness against different types of noises might vary for the same architecture, **adversarial robustness does not guarantee** overall robustness against all types of noise, suggesting that we need to account for the most pragmatic threat model in practice. (§ 12.3)

The preprint of this work is at [LQM22], and will be submitted to incoming conferences in 2023.

Realizing the current limitation of the definition of robustness and the limitation of existing robustness methods, in Part III, Chapter 13 illustrates the potential of diffusion models as a **data augmentation** and **denoising** tool to strengthen off-the-shelf classifiers' robustness against noise.

1. We demonstrated that when classifiers are trained on diffusion-synthesized data, their performance improves. The **enhanced performance** of classifiers trained on data generated by the diffusion model confirmed the effectiveness of these models as beneficial tools for boosting audio classifier performance, as is evidenced in § 13.5 where we use AudioSet trained Diffusion model to augment audio dataset. This validation could stimulate further efforts toward augmenting audio datasets.

2. We also showed that diffusion models, as off-the-shelf tools to denoise noisy audio, would serve as a strong defense against different kinds of common corruptions in audio, the **protection level varies depending on the type of noise** the diffusion model was trained on. Results in § 13.6 showed 3 different types of diffusion models trained with different noise and their varying capability to defend against the 3 common corruptions following the denoised smoothing [Sal+20] paradigm studied in Chapter 12.

This work is accepted at ICML2023 ESFoMo workshop[Li+23b], and is also in submission to NeurIPS 2023, and is open-sourced.[2]

All the above contributions conclude this thesis.

[2]https://audiojourney.github.io/

Chapter 15

Future Work

The progression of this thesis has led to the emergence of several promising avenues for exploration.

Better performing Mixture of Expert Models for Audio The confusion matrix of the model shown in Chapter 12 indicates that high-performance models do not make rudimentary mistakes, this suggests the promising design choice is to design a mixture of expert models. Namely, we could develop music experts, and animal sound expert models to further exploit the ontology presented in AudioSet.

Scale invariant/global metric for Sharpness As identified in Chapter 12, our calculation of sharpness scores might have not been meaningful due to the scaling difference across architectures. A fair metric of sharpness defined across architecture would be of vast value.

More general definition of Robustness In this thesis, as we stated in Chapter 1, we mostly focused on the problem of input robustness. In the real world, as we also showed in Fig.fig:robustness, robustness is a much broader concept. Among those other categories of robustness, task robustness, i.e. output robustness is an extremely important aspect for us to study. Currently, works that study output robustness mostly report their empirical observations on the specific task, and the field has not seen a unified framework for input robustness. When studying input robustness, we usually only need to focus on the encoder side of the model, and the theory could support this relatively simple setup. On the contrary, when analyzing output robustness, the results can be affected by not only the representation of the input but would get more influence by the decoding side. In sequence prediction tasks like ASR or speech-to-speech translation, for example, factors including but not limited to decoder architecture, lattice design, and regularization would all impact output robustness. It would be extremely complicated if not impossible to ablate the most influential factor when the model is connected to another pre-trained language model. Given the complexity of analyzing output robustness, we see a huge potential and urgency to develop a standard framework to modularize decoders and quantify their individual impact on robustness.

Multitask Audio Foundational Model: In other domains, especially in the NLP domain, multitask training paradigm brought about foundational models like T5[Raf+20] that could do well on a variety of tasks. However, we do not have such models yet in the audio domain. This again is a chicken-and-egg problem, due to the limited size of data we have in the audio domain. To make such a model, we should look at related datasets for audio and potentially explore the usage of external LLMs that were trained on out-of-domain but similar data. For instance, we could explore using music, and speech datasets to mix with audio and experiment with multitask learning.

Diffusion model as foundation models for Audio: In Part I of the thesis, we revisited the evolution of models from CNNs to Transformers, and from the supervised training paradigm to the self-supervised paradigm, but still, we are bounded by the very limited size of data in audio. In Part II, we foresee the limited adoption of traditional robust training methods due to their prohibitive cost associated with growing model size and data size. In Chapter 13, diffusion models exhibited promising capabilities in terms of both data augmentation and denoising. Its capacity for data augmentation can enhance the diversity and volume of training data for audio classification. By generating synthetic samples that retain essential properties of the original data, a diffusion model can significantly improve the learning process and generalization performance of audio classification models.

Additionally, the denoising ability of diffusion models is particularly valuable for audio classification. In real-world scenarios, audio data is often corrupted with various types of noise, including background noise, equipment noise, or signal interference. These types of noise can significantly degrade the performance of classification models. By applying a diffusion model to the data, we can effectively reduce or eliminate such noise, thereby enhancing the quality of the input data and consequently the reliability of the classification results.

These benefits of diffusion models seem to be perfect candidates to address the challenges we identified in Part I and Part II of the thesis. A promising direction is to combine/unify data augmentation and noise reduction capabilities of diffusion models, this could potentially be achieved through building specific sampling algorithms geared towards denoising purposes (but maintaining fidelity to the class) instead of simple reconstruction. Along this line, another promising direction is to introduce more specific control/conditioning signals to allow for more accurate intervention to achieve the desired augmentation or denoising results. Specifically, we could potentially look into training diffusion models using the expectation of transformation instead of using Gaussian noise alone, which potentially could let us attain a model that can denoise common distortion and augment data by adding common distortions at the same time.

However, canonical Diffusion models trained with Gaussian noise might not be a silver bullet yet, it still has issues including but not limited to, e.g. 1) the slow-sampling process: DDPM [HJA20] variant sampling methods still need multiple steps to approximate the noising process. Diffusion is still comparatively slower than GAN if it were to generate the same quality. 2) Diffusion lacks interpretability, which is a problem that plagues all generative models. Recently, we see a huge potential in unifying the concepts of Cold Diffusion[Ban+22] with MAE[Poy+22] which could be a candidate for a one-step diffusion model. On the other hand, exploiting more domain-motivated conditioning signals for diffusion models and analyzing the cross-attention could potentially aid interpretability.

Appendix A

Appendix for Chapter 6

A.1 Quantization:

We investigated the effectiveness of quantization for the VGG and MobileNet families by quantizing VGG19_bn and DSCNN respectively. There are two main strategies for quantizing neural architectures: Quantization Aware Training (QAT) and Post Training Quantization (PTQ) [Kri+20]. QAT requires retraining, which is sometimes impractical without the model's original training data and can be computationally expensive. Since PTQ is post-hoc quantization, it would be the more feasible option for developers lacking resources or necessary NAS skill sets. For these reasons, we chose to study PTQ as a complement to the previous QAT study in wake-word detection [PRP22]. We implemented quantization using PyTorch's open-source quantization library. However, it only has full support for CPU [Con22]. The results of using PyTorch Eager Quantization for static PTQ to quantize VGG19_bn and DSCNN from float32 to int8 are shown in Table A.1.

Quantized int8 models on CPU (Raspberry Pi 4B)			
CPU	Digital F1 (%)	Over-Air F1 (%)	Latency (ms) (× Speedup)
VGG19_bn	92.63	97.57 ± 4.07	162.0 (× 2.30)
DSCNN	91.86	95.13 ± 4.28	115.0 (× 1.31)
Unquantized float32 models on CPU (Raspberry Pi 4B) from Table 6.5			
CPU	Digital F1 (%)	Over-Air F1 (%)	Latency (ms)
VGG19_bn	93.37	90.80 ± 10.19	372.0
DSCNN	92.15	90.28 ± 7.25	151.0

Table A.1: Quantized Model Performance, VGG19_bn, DSCNN are identical to Rybakov et al. [Ryb+20]

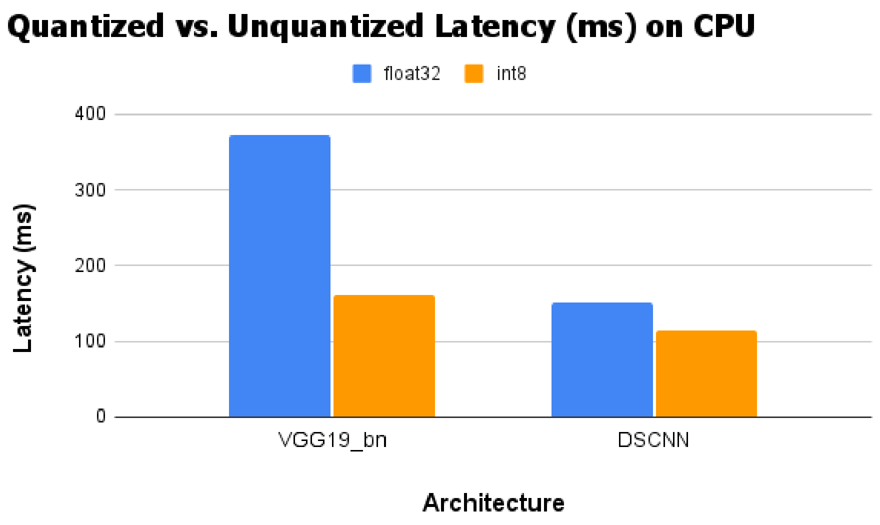


Figure A.1: Quantized vs. Unquantized Latency in ms on CPUQuantized Model Performance, VGG19_bn, DSCNN are identical to Rybakov et al. [Ryb+20]

For both models, we found that quantization improved latency without harming F1 scores relative to unquantized models (Table A.1). This suggests that PTQ is indeed a valuable strategy for developers seeking to increase model efficiency or to deploy architectures on edge devices.
However, we also noticed that the latency speedup for both models (2.3 times and 1.31 times, from Fig. A.1) were lower than the maximum speedup that could be achieved theoretically from float32 to int8 (4 times). Convolution's interaction with caches provides one possible explanation. As shown in Section 6.9.1, convolution is a component with heavy computation and may lead to frequent reloading of model weights into caches. After quantization, model weights would not need to be loaded into the cache as frequently since some weights which were different with 32-bit precision would be the same when quantized to 8-bit precision. In contrast, components with lighter computation complexity like DSCNN and MBConv blocks sequentially apply convolution one channel at a time, which decreases how often data would need to be reloaded into the cache. Since there is less reloading to begin with, they would benefit less from quantization than VGG19_bn. However, DSCNN is better for the overall latency despite receiving less speedup from quantization when compared to VGG19_bn.

Tables:

*The THOP library could not measure FLOPS for the Multi-Head Attention layer since it is implemented using the einops library. We are current finding another way to measure FLOPS for this layer.

A.2 Training Details:

All models were trained using our open-source code, which can be found at `https://github.com/Wakeword2023/EsFoMo2023.` The models were trained on the Google Speech Commands train dataset [War18] for the keyword 'Sheila.' The Speech Commands dataset contains 1372 utterances of the keyword. Before training, we augmented the dataset by applying transformations to the keyword utterances. Five copies of each transformed keyword utterance were put into the training dataset. In total, this gave us 6860 keyword utterances. We added 16558 non-keyword audio files to the training dataset, 712 of which were silence and 15846 of which contained randomly chosen non-keyword utterances from the Speech Commands dataset, which underwent the same transformations as the keyword utterances. The valid dataset was computed by applying transformations to audio files in the Speech Commands valid dataset for the keyword 'Sheila'. Only one copy of each transformed keyword utterance was added to the valid dataset, for a total of 188 valid keyword utterances. We added 620 non-keyword audio files to the valid dataset, 18 of which were silence and 602 of which contained randomly chosen non-keyword utterances from the Speech Commands dataset, which underwent the same transformations as the keyword utterances. We used the same training hyperparameters for all models except the Transformer, for which we referenced the training parameters described in [BOC21]. Table A.4 provides a breakdown of the training hyperparameters.

A.2.1 Digital Test Details:

Digital testing was conducted by inputting audio samples into the model and comparing the model's predictions with the samples' labels. The test dataset was computed by applying transformations to audio files in the Speech Commands test dataset for the keyword 'Sheila'. Only one copy of each transformed keyword utterance was added to the test dataset, for a total of 188 test keyword utterances. We added 465 non-keyword audio files to the test dataset, 18 of which were silence and 447 of which contained randomly chosen non-keyword utterances from the Speech Commands dataset, which underwent the same transformations as the keyword utterances. The F1 score resulting from digital testing on the test dataset is the 'Digital F1' score that appears in Table A.2.

A.2.2 Over-the-air Test Details:

As described in Table 6.5, for each model five over-the-air trials were conducted both on GPU and on CPU. The audio was recorded and processed in 1-second chunks during each over-the-air trial. Each trial lasted 20 seconds, yielding 20 audio chunks. Immediately after being recorded, the 1-second chunk was downsampled to 16kHz and converted into a logMel spectrogram before being sent as input to the classifier. While there are several available approaches to wake-word detection [Mit+20; Wan+21; Ryb+20; Cou+19], it has been shown that audio preprocessing using logMel spectrograms yields the best performance [PRP22], and so this is the approach that we followed. After classification, the model's prediction for whether the wake-word was present in that chunk was printed to the console. Subjects uttered a phrase including the wake-word ten times per trial and manually determined each model's prediction accuracy. This data was used to compute an average over-the-air F1 score for each model by Average $= \frac{1}{|S|}\sum_s \frac{1}{|T|}\sum_t \frac{\mathrm{TP}_t}{\mathrm{TP}_t+\frac{1}{2}(\mathrm{FP}_t+\mathrm{FN}_t)}$ where $s \in S$ is a unique subject and $t \in T$ is a unique trial. For each trial, latency was calculated as the time between a model receiving input and returning its corresponding prediction.

A.3 FLOPS Count & Hooks:

We measured the total floating point operations per second (FLOPS) and parameter count for each model using the THOP library [Zhu22]'s profiler. Since THOP measures MACS, we used the widely accepted estimation FLOPS = MACS × 2. In addition, timing functionality was added to each model in order to find the runtime for individual layers. PyTorch forward hooks collected and logged the timing results for all layers. We timed the following layers: Conv2d, Batchnorm2d, ReLU, Maxpool2d, Avgpool2d/AdaptiveAvgpool2d, Dropout, Linear, LayerNorm, and GELU. Avgpool2d and AdaptiveAvgpool2d were treated as the same layer for conciseness of presentation in Table A.2. We did not time Sigmoid, Softmax, or Swish/MemoryEfficientSwish layers. Finally, we modified the THOP library's profiler to log MACS for individual layers by having its dfs_count function keep track of MACS for individual leaf-node layers in the model graph. THOP automatically zeroes the MACS for Dropout, ReLU, GELU, LayerNorm, Sigmoid, Softmax, and Swish/MemoryEfficientSwish layers.

A.3.1 Batch Normalization Additional Analysis:

The effects of Batch Normalization's limited parallelizability are visible in Table A.2, where BN layers accounted for the most computation on GPU but not on CPU for three out of the six models. Since BN layers receive less relative GPU acceleration than convolution, it follows that its GPU and CPU latencies will be less disparate. The only model to see PAR for convolution layers decrease moving from GPU to CPU was EfficientNet_b1, the smallest model among those tested (Table A.2). This occurred in conjunction with an increase of time spent in BN layers, suggesting that EfficientNet_b1 has few enough parameters for convolution to be relatively low cost. This indicates that while most of the computation cost in these models comes from convolution, the cost of batch normalization is non-negligible and should be considered in model selection.

A.3.2 Images versus Audio Non-Transferability:

The results of Table 6.5 show that EfficientNet_b1 [TL19] and EfficientNet_b7 [TL19], optimized on both accuracy and FLOPS for computer vision tasks, do not transfer the same accuracy versus latency tradeoff to the audio wake-word task. Compared to DSCNN (MobileNet based architecture) [Ryb+20], EfficientNetV1 architectures have better accuracy but inferior latency. DSCNN precedes EfficientNetV1 and EfficientNetV2, which were trained using NAS (Neural Architecture Search) on CIFAR or other vision datasets. That DSCNN outperforms its successors in the wake-word detection task should motivate future works to train CNN-based audio models jointly optimized for FLOPS and accuracy.

A.3.3 Calculations:

For all models except Transformer, we broke model layers into seven categories: Conv2d, BatchNorm2d, ReLU, MaxPool2d, Linear, Dropout, and AveragePool2d. The AveragePool2d category represented both AveragePool2d and AdaptiveAveragePool2d layers. For Transformer, we broke model layers into x categories: Multi-Head Attention, Linear, LayerNorm, and GELU. For each trial, a model's layers were first sorted into these categories regardless of parameters. The aggregate runtime and run count were calculated for each category by summing the individual runtimes of layers in the same category, and by counting every time a layer of a specific category was run. Finally, aggregate FLOPS per category were calculated using the results from the THOP library, summing the FLOPS per layer for layers in the same category. After finding aggregate runtime, run count, and FLOPS per category for each trial, these values were averaged across all trials, as summarized by the following equations, where t is trials, n_t is the nth (and last) layer to run in trial t, c is a layer category, r_c is average runtime for category c, k_c is average run count for category c, and m_c is average FLOPS for category c:

$$r_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t} \text{runtime}(l)\{l \in C\}$$

$$k_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t} \{l \in C\}$$

$$m_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t} \text{FLOPS}(l)\{l \in C\}$$

A.4 Quantization Details:

We quantized both VGG19_bn and DSCNN using PyTorch Eager Mode quantization [Con22]. For DSCNN, we adapted our existing code for quantization following the

'Model Architecture' and 'Post-training Static Quantization' steps in the available tutorial at `https://pytorch.org/tutorials/advanced/static_quantization_tutorial.html`. For VGG19_bn, we used similar code following the 'Post-training Static Quantization' step in this tutorial. Both models were quantized from float32 to int8. Our quantization implementation is available in our repository. Both of these models were neither difficult nor time-consuming to quantize with the publicly available PyTorch tutorial, and the effects of quantization on latency (Table A.1) were readily apparent during testing.

The Post-Training Quantization Landscape: After quantizing VGG19_bn and DSCNN as representatives of the CNN family (VGG19_bn, ResNet50) and the DSCNN family (DSCNN, EfficientNet, EfficientNetV2), we sought to quantize the other models we examined in order to provide more detailed, granular analyses of quantization for these models. However, it soon became apparent that this is not as simple as using the widely available tools for PTQ.

Pytorch Eager Mode Quantization [Con22]: Pytorch Eager Mode quantization is still in beta. We successfully used it to quantize VGG19_bn and DSCNN, following PyTorch tutorials. It is not always very simple to use: bugs can be particularly confusing, and documentation is somewhat sparse. Nonetheless, it was far and away the best PTQ library that we experimented with, and it is the library that we plan on using in the future as we work on quantizing the rest of the models examined in this thesis.

PyTorch FX Graph Mode Quantization [Con22]: PyTorch introduced this mode of quantization, which is currently a prototype, as a simpler and more automatic alternative to PyTorch Eager Mode quantization. We found that it was more difficult to use than PyTorch Eager Mode quantization for VGG19_bn and DSCNN, both of which were already symbolically traceable. We could not get either of them to work within a reasonable amount of time following the official PyTorch FX quantization tutorials. Likewise, some of our other models like Transformer rely heavily on libraries and operations which PyTorch FX Graph Mode cannot quantize. To quantize our Transformer implementation using PyTorch FX Graph Mode, we would have to rewrite it entirely, which goes against the goal of the PyTorch FX library.

TorchQuant Library and the PTQ landscape: We also investigated third-party quantization libraries that advertised PTQ capabilities. In particular, we focused on the TorchQuant library [TAL21] which is geared towards quantization researchers, and has both QAT and PTQ modes. It also has fused implementations of EfficientNet, MobileNet, and ResNet designed to work with its own model of quantization. Without too much trouble, we were able to successfully integrate our own model implementations and input with the TorchQuant fused models and quantization. However we discovered that TorchQuant cannot actually do PTQ since it is not integrated with the PyTorch native ATen library. Given the instruction to quantize weights to eight bits, TorchQuant's quantization takes the float32 weights and zeroes out all information beyond eight bits. However, the weights remain in float32 format. This kind of 'simulated quantization' works well for QAT, and allows researchers to investigate how quantization impacts model accuracy without worrying about how it interacts with the PyTorch backend. But the goal of PTQ is to actually quantize the weights sent to the device to a smaller data representation in order to study the efficiency as well as the accuracy of quantized models, and the TorchQuant library cannot achieve this.

Takeaways: Because PTQ requires interaction with lower-level PyTorch, it is extremely challenging to do PTQ using PyTorch without using PyTorch's own quantization library. For this reason, current open-source options for PTQ are very limited, which creates challenges for developers. Part of the reason we chose to investigate PTQ rather than QAT was that we perceived it to be more practical: both because it doesn't require the computing power and technical background necessary to train a model as QAT does, and because it has a direct impact on a model's efficiency in addition to its theoretical accuracy. For researchers and developers looking to run neural networks on edge, PTQ could be a way to improve the efficiency of 'off-the-shelf' pre-trained models. Yet as we discovered, PTQ is more difficult than QAT, and although PyTorch's quantization libraries are not fully developed, there are few alternatives. The open-source tools available for implementing PTQ do not live up to their potential.

Future Work: We are still working on quantization for the other models we examined. In particular, there are several promising open-source fused transformer implementations that are designed to work with PyTorch eager quantization. We will update our repository as we quantize more models.

A.5 Hardware and Software:

In all of our experiments, our GPU was a 2020 13-inch MacBook Pro with an Apple M1 chip running macOS Ventura 13.1. We chose to use an M1 Mac since it has support through Pytorch MPS, and because it is a good example of a common, fairly portable GPU. Our CPU in all experiments was a Raspberry Pi 4B running 64-bit Raspberry Pi OS with Desktop, Debian version 11 (bullseye). We chose Raspberry Pi 4B as our CPU in this work because it is a common, open-source, portable CPU, and it is cheaper than many other CPU examples and thus more financially feasible for researchers and developers. The Raspberry Pi does not come with a built-in microphone, so we equipped it with the Seeed ReSpeaker 4-Mic Array for Raspberry Pi for all of our experiments. While the PyTorch device can be set to CPU on M1 Mac, we chose not to use it because it is not a good representative of a typical CPU. A 13-inch M1 MacBook Pro has eight cores and eight threads [Cm23], some of which are optimized for performance and others which are optimized for efficiency [Kos21]. This means that the M1 Mac CPU has greater potential for parallelization and can sometimes behave more like a GPU, as shown in Table A.5. We ran tests to obtain over-the-air latencies for all of our models, finding that parallelizable models (CNN family, Transformer) had relatively low latencies but models optimized for highly sequential devices (DSCNN family) had poorer latencies.

Based on our understanding of M1 architecture, the Mac M1 CPU does not seem like a good representation of a portable, edge CPU. Thus, we chose not to use it in our experiments.

Audio Acquisition Time: For both GPU and CPU, audio acquisition time is minimal. We measured that it was 1.5ms on M1 Mac and 1.0ms on Raspberry Pi. We calculated audio acquisition time as the difference between the time of an utterance of audio and the time when it is received by the device memory. This was done by simultaneously starting a stopwatch and a recording, observing the time of an utterance on the stopwatch, and then comparing this time to the time when the utterance was recorded.

A.6 Other Models

We also trained and tested MnasNet and DenseNet models. Since DenseNet is not an efficient architecture and MnasNet was optimized for object detection, they are not the focus of our main investigation, but we include their experimental results here.

Model (#Param)	**VGG19_bn** (38.9M) [Ryb+20]					**DSCNN** (2.23M) [Ryb+20]					**EfficientNet_b1** (0.06M) [TL19]				
	Avg Latency		AvgFLOPS	PAR (%)		Avg Latency		AvgFLOPS	PAR (%)		Avg Latency		AvgFLOPS	PAR (%)	
	GPU	CPU		GPU	CPU	GPU	CPU		GPU	CPU	GPU	CPU		GPU	CPU
Conv	0.369	12.938	3.90E+07	**37.56**	**56.57**	0.235	1.411	2.40E+05	39.51	**53.09**	0.340	1.348	5.63E+08	**67.38**	**60.09**
BN	0.334	3.656	1.39E+05	34.07	15.94	0.252	1.149	1.19E+04	**42.25**	43.22	0.220	1.438	1.15E+04	26.25	38.49
ReLU	0.136	0.542	0	15.53	2.67	0.151	0.134	0	17.04	3.41	-	-	-	-	-
SiLU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MaxPool	0.235	3.346	0	7.49	4.55	-	-	-	-	-	-	-	-	-	-
Linear	0.274	24.691	1.26E+07	5.14	20.22	0.347	0.249	5.12E+03	1.12	0.18	0.300	0.139	5.12E+03	0.52	0.05
Dropout	0.017	0.084	0	0.22	0.05	0.029	0.137	0	0.10	0.10	0.020	0.072	0	0.03	0.03
AvgPool	-	-	-	-	-	-	-	-	-	-	0.141	0.155	2.13E+02	5.82	1.44

Model (#Param)	**EfficientNet_b7** (0.32M) [TL19]					**EfficientNetV2_m** (53.5M) [TL21]					**EfficientNetV2_xl** (207M) [TL21]				
	Avg Latency		AvgFLOPS	PAR (%)		Avg Latency		AvgFLOPS	PAR (%)		Avg Latency		AvgFLOPS	PAR (%)	
	GPU	CPU		GPU	CPU	GPU	CPU		GPU	CPU	GPU	CPU		GPU	CPU
Conv	0.240	2.501	7.13E+10	**65.21**	**69.84**	0.189	2.865	1.37E+06	32.91	**59.10**	0.149	3.921	2.58E+06	31.02	**68.11**
BN	0.177	1.748	4.26E+03	28.69	29.14	0.201	1.570	1.38E+04	**35.09**	32.52	0.175	1.277	1.79E+04	**36.26**	21.18
ReLU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SiLU	-	-	-	-	-	0.090	0.123	0	14.45	2.34	0.075	0.146	0	14.52	2.46
MaxPool	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Linear	0.347	0.147	4.94E+05	0.345	0.02	0.121	0.434	8.76E+03	11.77	4.88	0.104	0.687	4.93E+05	12.37	7.14
Dropout	0.022	0.073	0	0.02	0.01	-	-	-	-	-	-	-	-	-	-
AvgPool	0.103	0.174	9.14E+01	5.73	1.00	0.118	0.201	1.22E+04	5.79	1.15	0.098	0.216	1.53E+04	5.84	1.12

Model (#Param)	**ResNet** (23.5M) [He+16a]				
	Avg Latency		AvgFLOPS	PAR (%)	
	GPU	CPU		GPU	CPU
Conv	0.383	3.286	3.09E+6	**44.79**	**57.08**
BN	0.299	2.248	1.57E+04	34.98	39.03
ReLU	0.156	0.087	0	16.90	1.40
SiLU	-	-	-	-	-
MaxPool	0.476	5.350	0	1.05	1.75
Linear	0.684	0.153	8.19E+03	1.51	0.05
Dropout	-	-	-	-	-
AvgPool	0.344	2.125	4.10E+03	0.76	0.70
LayerNorm	-	-	-	-	-
GELU	-	-	-	-	-

Table A.2: Analysis of Specific Layer Classes

Average Latency: Latencies of individual layers were averaged within their layer category for each trial, in milliseconds (ms)
PAR (Percentage of Aggregate Runtime): Measures the percentage of a model's total runtime spent computing layers in a specific category across trials. Percentages for categories taking up the most relative computation time are bolded
Average FLOPS: Measures the average floating point operations per second across all trials for a layer class
Conv: Convolution layers; **BN**: Batch normalization layers
Note: If a model does not have the layer in a row, it is filled with "-". 0 FLOPS indicates an existing layer has no FLOPS.

Model (#Param)	**Transformer** (0.60M) [BOC21]				
	Avg Latency (ms)		Avg FLOPS	PAR (%)	
	GPU	CPU		GPU	CPU
Multi-Head Attention	2.631	3.082	*	54.19	22.71
Linear	0.432	1.096	4.89E+2	19.26	17.43
LayerNorm	0.566	2.570	1.69E+4	23.29	37.96
GELU	0.134	2.804	0	2.75	20.70
Dropout	0.012	0.081	0	0.51	1.20

Table A.3: Transformer Layers and Operations

	Non-Transformer	Transformer [BOC21]
Batch Size	64	512
Epochs	75	75
Warmup Epochs	0	10
Weight Decay	0.01	0.1
Optimization	SGD	AdamW
Learning Rate	0.0001	0.001
Learning Rate Scheduler	Plateau	Cosine Annealing

Table A.4: Training Hyperparameters

CPU (Mac-M1)	Latency
VGG19_bn [Ryb+20]	69.5
DSCNN [Ryb+20]	350.6
EfficientNet_b1 [TL19]	566.1
EfficientNet_b7 [TL19]	2562.6
EfficientNetV2_m [TL21]	1711.3
EfficientNetV2_xl [TL21]	4401.1
ResNet50 [He+16a]	54.5
Transformer [BOC21]	23.8

Table A.5: Model Latency on Speech Commands Dataset

GPU (Mac-M1)	Digital F1	Over-the-Air F1	Latency
MnasNet [Tan+19]	91.58	82.59 ± 12.54	35.3
DenseNet [Hua+17]	93.37	94.85 ± 4.49	344.2
CPU (Raspberry Pi 4B)	**Digital F1**	**Over-Air F1**	**Latency**
MnasNet [Tan+19]	91.58	82.37 ± 14.16	131.1
DenseNet [Hua+17]	93.37	42.22 ± 6.81	3674.0

Table A.6: Additional Model performance on Speech Commands Dataset. F1 scores are scaled to 100.

Model (#Param)	**MnasNet** (3.10M) [Tan+19]				
	Avg Latency (ms)		Avg FLOPS	PAR (%)	
	GPU	CPU		GPU	CPU
Conv	0.197	1.160	2.41E+5	39.41	53.18
BN	0.228	0.943	1.20E+4	45.68	43.14
ReLU	0.096	0.094	6.54E+2	12.90	2.95
MaxPool	-	-	-	-	-
Linear	0.294	0.322	5.12E+3	1.19	0.30
Dropout	0.202	0.473	0	0.81	0.44
AvgPool	-	-	-	-	-
Model (#Param)	**DenseNet** (25.6M) [Hua+17]				
	Avg Latency (ms)		Avg FLOPS	PAR (%)	
	GPU	CPU		GPU	CPU
Conv	0.130	10.583	7.66E+7	33.68	80.58
BN	0.190	1.731	1.58E+6	47.89	12.82
ReLU	0.074	0.816	-	18.48	6.07
MaxPool	-	-	-	-	-
Linear	0.267	0.254	8.76E+3	0.37	0.04
Dropout	-	-	-	-	-
AvgPool	0.143	3.283	1.51E+3	0.58	0.49

Table A.7: Other Models: Analysis of Specific Layer Classes

Appendix B

Appendix for Chapter 11

B.1 Proof of Theorem 1

Let's consider a binary classification task as an example for simplicity. Let $(x_{A,i}, x_{V,i})$ be a point with different prediction results for audio modality A and video modality V. Assume $\exists a, b$, such that $a^T g(x_{A,i}) = -s < 0$ and $b^T h(x_{V,i}) = t > 0$ where $s, t > 0$, and the correct label is -1. For the point-wise robustness threshold $\epsilon_{A,i}$ of this point, where an attack $\{\delta_{\epsilon_{A,i}} : ||\delta_{\epsilon_{A,i}}||_P \leq \epsilon_{A,i}\}$ changes the prediction label. By definition, we know $a^T g(x_{A,i} + \delta_{\epsilon_{A,i}}) \geq 0$ and $a^T g(x_{A,i} + \delta) \leq 0$ for all $0 \leq \delta \leq \epsilon_{A,i}$. If $s < t$, then the fused network predicted the wrong label even without any noise.

$$\begin{aligned} f(x_{A,i}, x_{V,i}) &= (a, b)^T (g(x_{A,i}) \oplus h(x_{V,i})) && \text{(B.1)} \\ &= a^T g(x_{A,i}) + b^T h(x_{V,i}) && \text{(B.2)} \\ &= -s + t > 0 && \text{(B.3)} \end{aligned}$$

Otherwise, in the case of $s \geq t$, by applying Intermediate Value Theorem to $g(x)$, there exists a point $0 \leq \delta \leq \epsilon_A$ such that $a^T g(x_{A,i} + \delta) = -t/2$:

$$\begin{aligned} f(x_{A,i} + \delta, x_{V,i}) &= (a, b)^T (g(x_{A,i} + \delta) \oplus h(x_{V,i})) \\ &= a^T g(x_{A,i} + \delta) + b^T h(x_{V,i}) \\ &= -\frac{t}{2} + t > 0 \end{aligned} \qquad \text{(B.4)}$$

In both cases, we could find a noise $0 \leq \delta < \epsilon_{A,i}$ within the original unimodal robustness threshold to attack the multimodal network successfully. Vise versa for video. Thus, a unimodal attack can break a mulimodal model, which we also empirically verified the existence of such cases in our experiments. (see Table 2 of the main paper).

We postulate that such phenomenon is like the Mcgurk Effect, where multimodal fusion would further distort the already non-convex decision boundary (Figure 1 of the main paper), making the fused decision boundary very different than the original ones and unpredictable.

Appendix C

Appendix for Chapter 13

Appendix Outline:

1. Section C.1 will cover more details about prompt engineering, and the usage of entailment scoring.
2. Section C.2 will cover more details about our model training and hyperparameters
3. Section C.3 offers a further in-depth understanding of our **cross-attention mechanism** as described in Equation (4) in the main paper, and why it produced the results in Table 4 in the main paper.
4. Section C.4 includes more generation results from our diffusion model.

C.1 More on Prompt Engineering:

C.1.1 Prompting given audio-only weak labels:

Table C.1 shows the few-shot examples we feed Alpaca model with to generate audio captions based on the labels given.

C.1.2 Filter Hallucination and Obtain Visual Captions:

We noticed that many of the single-label audio captions often had hallucinations where there would be extra details added (usually from one of the examples given). One example of this would be *"A person is sprinting while a dog is barking and howling."* when only the *"run"* label was given.

To address this, we created visual captions to help enrich our audio data. Table C.2 shows the specific examples we used to join together the three visual captions that were generated using BLIP2 [Li+23c].

Prompt	**Response**
alarm, burp, inside, small room.	burping while an alarm plays inside a small room.
dog, bark, howl, speech.	a dog barking and howling with a person speaking as well.
Music, jazz, piano, singing, speaking.	a person plays jazz piano with a singer while people talk.
engine, vehicle, wind, music, speech.	people talking inside a car while driving and listening to music.
water, gargle, inside, small room.	air is passing through the water in their mouth in a small room with water.
scratch, hammer, metal.	hammer striking a metal surface and scratching sounds can be heard.
thunder, wind, bark, small room.	a dog is barking in a small room during a thunderstorm with audible wind.
gunshot, vehicle engine, siren, crash.	a car chase with gunfire and sirens where a vehicle crashes.
waterfall, wind, sizzle, crackle.	a fire is cracking with something sizzling near a waterfall with wind.
stream, cough, cat, Purr.	a cat purrs near a coughing person while a stream can be heard.

Table C.1: The list of audio labels is preceded by the prompt: *"For each of these, summarize the sounds into a single sentence: \n describe a situation with all of these sounds together:"*

C.1.3 Merging Audio and Visual Captions:

Table C.3 shows the specific examples that were used in the prompt to join our single-label audio class and summarized video captions. These examples help us filter the hallucination generated when creating captions from audio-only weak labels by utilizing visual information.

C.1.4 Audio-visual Entailment Score:

These scores can be used to address the audio-visual false positive issue in which the audio of the video does not match the visual frames presented. These scores also increase human understanding of image-audio-text correlations. We can further utilize these scores to help

Prompt	Response
a video game with a dragon in the water final fantasy xv - the end of an era final fantasy x-2 - the end of an era.	The end of an era awaits in Final Fantasy XV and X-2.
a person is using a printer to print out a document a white cash register with a keyboard and a keypad a person is holding a small piece of plastic on a plane.	a person using a piece of plastic to make a purchase at a cash register.
a man with a beard sitting on a bed a man in a cowboy hat is playing an acoustic guitar a man standing in the desert holding an acoustic guitar.	A man in a cowboy hat is playing a guitar and strumming away in the desert.

Table C.2: Few-shot merging visual caption examples. Each set of captions is preceded by this prompt: *"Create one sentence that summarizes these three simply:"*

perform caption filtering, gating based on the entailment score between the audio and visual concepts demonstrated within each video.

Entailment scoring with Alpaca, T5, and BERT:

We conducted entailment scoring using Alpaca[Tao+23], T5[Raf+20], and BERT[Dev+19] on the balanced set of AudioSet[Gem+17b]. This allowed us to compare the results of the three methods and how they would consider entailment differently. Since Alpaca[Tao+23] is a decoder-only model, we noticed that the entailment scoring was not always reliable, scoring differently on additional runs. This is due to its auto-regressive nature, which is highly dependent on its context. To stabilize the scoring, we employ few-shot examples (in Table C.4, Table C.5) to better utilize the in-context learning ability of Alpaca. When conducting entailment scoring for audios that only had one label, we utilized the AudioSet ontology description [Gem+17b] and label due to the hallucination in the default audio captions.

Despite using these examples, we still noticed that Alpaca would sometimes output "incorrect" scores, e.g. given the following pairing: (*"A background of traditional Indian music with lyrics from a bhangra song playing in the foreground."*, *"A bright star is twinkling in the night sky, shining amidst the dark velvety backdrop."*) a score of 0.9 or (*"A person*

Prompt	**Response**
man in a cowboy hat is throwing a rope while standing on a green field. Whip	An audio scene emphasizing the sharp sound of a whip, set against a visual backdrop of a man in a cowboy hat throwing a whip on a green field.
a man eating a cupcake at a table Tick	An audio scene centered around the subtle ticking sound, with a backdrop of a man enjoying a cupcake at a table.
a person using a piece of plastic to make a purchase. Cash register	An audio scene capturing the distinctive sound of a cash register, accompanying the moment when a person uses a piece of plastic to make a purchase.
a man brushing his teeth Toothbrush	a man cleans his teeth with a toothbrush's audible scrubbing.
man standing in the desert holding an acoustic guitar Country	An audio scene focused on the Country genre, featuring a man standing in the desert holding an acoustic guitar.

Table C.3: Caption-label pair is preceded by the following prompt: *"Summarize these two captions conditioned on the second caption, the second caption describes an audio class and is the main concept:"*

dribbling a basketball, slamming it on the ground, and speaking.", "*A man in a purple shirt is playing basketball, a man in a red shirt is playing soccer.*") a score of 0.1. These cases are very counter-intuitive for humans to explain.

On the other hand, we noticed that encoder-decoder models (we only use the encoder) such as T5 [Raf+20] and BERT[Dev+19] scoring tended to score some single-label (ontology-based) captions lower than an alpaca-generated caption that had less which audio-visual

Prompt	**Response**
a man in a cowboy hat is throwing a rope while standing on a green field. Whip : The sound of whipping, i.e., the greatly accelerated motion of the tip of a flexible structure, as the result of concentrated angular momentum.	0.85
a man eating a cupcake at a table Tick : A metallic tapping sound.: A metallic tapping sound.	0.00
a person using a piece of plastic to make a purchase. Cash register : Sounds of a mechanical or electronic device for registering and calculating transactions, usually attached to a drawer for storing cash.	0.45
a man brushing his teeth Toothbrush : Sound of an instrument used to clean the teeth and gums consisting of a head of tightly clustered bristles mounted on a handle.	1.00
man standing in the desert holding an acoustic guitar Country : A genre of United States popular music with origins in folk, Blues and Western music, often consisting of ballads and dance tunes with generally simple forms and harmonies accompanied by mostly string instruments such as banjos, electric and acoustic guitars, dobros, and fiddles as well as harmonicas.	0.90

Table C.4: The caption-label pair is preceded by the following prompt: *"on a scale from 0 to 1, output the probability that the first caption describes a scenario with the second caption's sound description:"*

correspondence. For example, when the caption pairing was (*"An ice cream truck outside a small room, playing music."*, *"An old digital audio box sits proudly on a table, displaying its unique blue and white design."*), T5 gave it a score of 0.81 whereas when the pairing was (*"Fire : Sounds resulting from the rapid oxidation of a material in the exothermic chemical process of combustion, releasing heat, light, and various reaction products."*, *"Firefighters are putting out a blazing fire in a building."*), T5 only gave it a score of 0.80 despite the two captions being much closer in meaning.

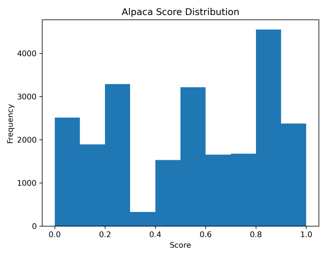
Although there were some discrepancies when analyzing these scores, we still found that overall, T5 scoring seemed to be more consistent with the actual content of the captions. The different score distributions for the three metrics utilized on the balanced set can be found here (Figure C.1a, Figure C.1b, Figure C.1c).

These distributions reinforce how T5 had a generally higher score range whereas BERT had scores in a lower range (even below 0) as compared to Alpaca. It also shows how

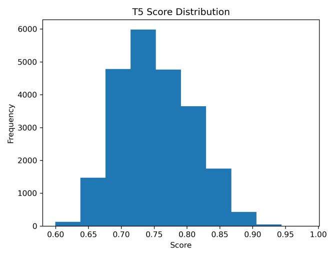
Prompt	**Response**
two young men wearing red and white shirts A person is speaking while there is a loud gush of air.	0.10
goat grazing on grass in the mountains A goat making music and someone speaking as well.	0.75
a young boy is riding a skateboard down a sidewalk A male is singing and a child is singing along to the same music.	0.05
a person is holding a gun with a glove on it Gunfire and cap gun.	0.80
a screenshot of a game with three people in red and blue outfits Gunfire and cap gun.	0.00
a person playing a ukulele with their hands There is background music with a mandolin being played.	0.70
a man in a green shirt is talking to the camera A cat meowing and a person speaking.	0.50
a person holding a snake in front of a door A snake hissing.	0.60

Table C.5: The caption pair is preceded by the following prompt: *"on a scale from 0 to 1, output the probability that the first caption describes a scenario with the second caption's sound description:"*

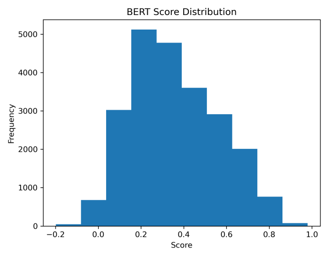
T5 and BERT had similarly shaped distributions whereas Alpaca tended to spread a bit more evenly across the spectrum (with peaks at different points in the score distribution). The result of the 3 different types of scoring on the balanced set can be found on our open-source page[1]. We have also utilized T5 scoring on the unbalanced set for AudioSet which will also be made available.



(a) Alpaca-generated entailment score distribution for balanced set



(b) T5-generated entailment score distribution for balanced set



(c) BERT-generated entailment score distribution for balanced set

Figure C.1: Comparison of entailment score distributions for different models on AudioSet [Gem+17b] balanced set.

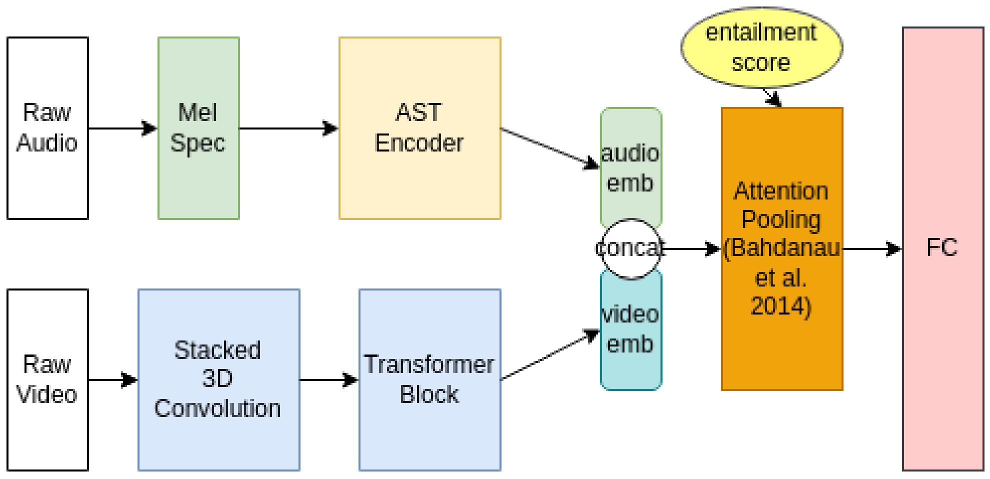


Figure C.2: Entailment score penalizing attention score at the attention pooling layer

Effect of Audio-visual Entailment Scores

To test the effectiveness of our entailment score, we employ a similar audio-visual classification pipeline as described in [Li+22b], which is illustrated in Figure C.2. Our audio encoder is the same AST/DeiT model[GCG21a] used in the main paper. The video is encoded by a pre-trained R2+1D[Tra+18] model. The naive fusion is pure concatenation. We plug in our entailment scores in the attention mechanism by discounting the attention score for the video portion. Specifically, $\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{t,i}\mathbf{h}_i$ the corresponding embedding indexes' attention score gets discounted by the entailment score. The result is shown in Table C.6. Although our best score still lags behind the best SoTA model on AudioSet, the

[1]https://audiojourney.github.io/

point is our improvement over naive fusion shows all three types of entailment scores outperform the naive version of fusion.

Model	Backbone	PT	AS-20k (mAP)			AS-2M (mAP)		
			A	V	A+V	A	V	A+V
Naive Fusion	DeiT-B/R2+1D	IN+KI-SL	34.6±.20	18.1±.09	37.4±.18	45.4±.70	23.9±.12	46.5±.29
Alpaca score Fusion	DeiT-B/R2+1D	IN+KI-SL	34.6±.20	18.1±.09	38.4±.14	45.4±.70	23.9±.12	47.6±.33
T5 score Fusion	DeiT-B/R2+1D	IN+KI-SL	34.6±.20	18.1±.09	**39.1**±.10	45.4±.70	23.9±.12	**49.5**±.42
BERT score Fusion	DeiT-B/R2+1D	IN+KI-SL	34.6±.20	18.1±.09	38.7±.15	45.4±.70	23.9±.12	49.1±.37
AST [GCG21a]	DeiT-B	IN	34.6	-	-	45.4	-	-
MBT [Nag+21b]	ViT-B	IN-SL	31.3	27.7	43.9	41.5	31.3	49.6
CAV-MAE [Gon+22b]	ViT-B	SSL	37.7	19.8	42.0	46.6	26.2	51.2

Table C.6: Metrics are mAP for AS. For pre-training (PT) dataset, AS:AudioSet, KI:Kinetics (for R2+1D[Tra+18]), and IN:ImageNet. SSL: self-supervised learning, SL: supervised learning; We gray-out baselines. Best single models in AS-2M are compared (no ensembles).

C.1.5 More Details about Caption Generation:

Similarity Score (Table 2 of main paper) We recruited 5 human subjects to evaluate 500 samples and report their average ratings for zero-shot, one-shot, and few-shot and A-V Merge captions and WavCaps [Mei+23] generated results in Table 2 of the main paper. When evaluating our captions, we ask the human subject to provide a similarity score between the generated captions and the AudioCaps[Kim+19] ground truth, ranging from 0-1. In the cases of ambiguous samples or when AudioCaps[Kim+19] labels are not reliable, we also provide the original youtube links to the evaluators and ask them to use the *audio content* as the ground truth.

Key Statistics Comparison: To further compare the different caption sets, we analyzed the vocabularies of AudioCaps [Kim+19] versus the WavCaps [Mei+23] and captions our system generated from the same clips. By observing the top 20 words (Figure C.5, Figure C.3, Figure C.4), we can see that AudioCaps and WavCaps have many similarities with 14 of the top 20 words being identical. These similarities help support the closeness of vocabulary between the two sets (resulting in high automatic metrics), meanwhile reducing the diversity and generalizability of the captions. We also noticed that the top 20 words of our audio-video merged captions contain the term ‘audio scene’ which reflects the examples we used when prompting Alpaca. This shows a clear path that we could inject our own inductive bias into any future LLM-based dataset augmentations. In addition to

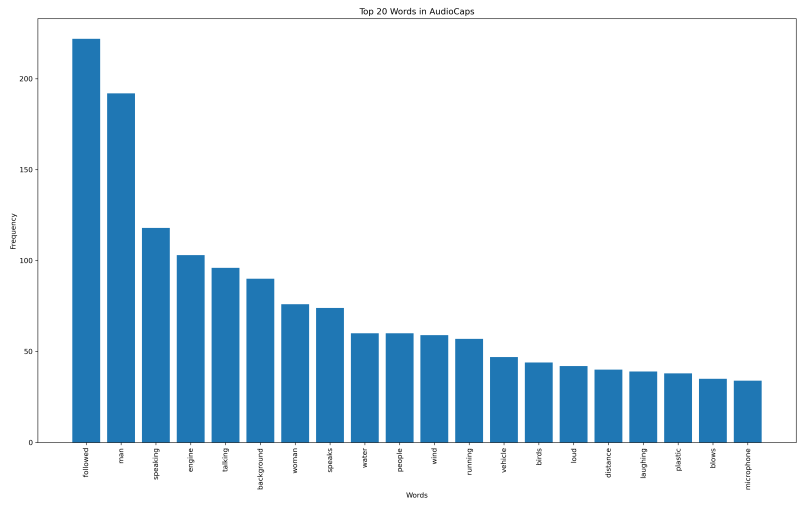


Figure C.3: Top 20 words from the vocabulary of the AudioCaps samples (not including stop words)

the vocabulary size difference, we also noticed a key difference in the length of captions generated through our system versus those of AudioCaps or WavCaps. The minimum caption length for all three main types of captions was all 3. The maximum caption differed a lot, with the AudioCaps [Kim+19] samples having a maximum of 31, WavCaps [Mei+23] samples having a maximum of 25, and our audio-video merged captions having a maximum 45. Additionally, the average length of AudioCaps is 10.49, for WavCaps is 6.89, and for our audio-video merge was 17.07. This demonstrates a drastic increase in length and thus richness of the captions generated by our LLM-based audio-video merging methods. We see that the WavCaps [Mei+23] captions are generally the shortest which suggests a lack of variation in structure when generating captions.

Some key examples of where utilizing audio-video merged captions benefited our model over WavCaps[Mei+23] are included in Table C.7. We see that in these examples, our captions utilized details such as the flag or the caption on the video to determine what exactly the sound was, adding key details that weren't apparent in the WavCaps captions. These also demonstrate how simple WavCaps captions are overall.

Overall, we believe this is a very promising paradigm to augment the existing audio dataset. In the future, if we could leverage high-performance classifiers to auto-label audio from the wild or to fine-tune the LLM to plug into the Automatic Audio Captioning system, it would be a clear pathway to scale up audio training datasets.

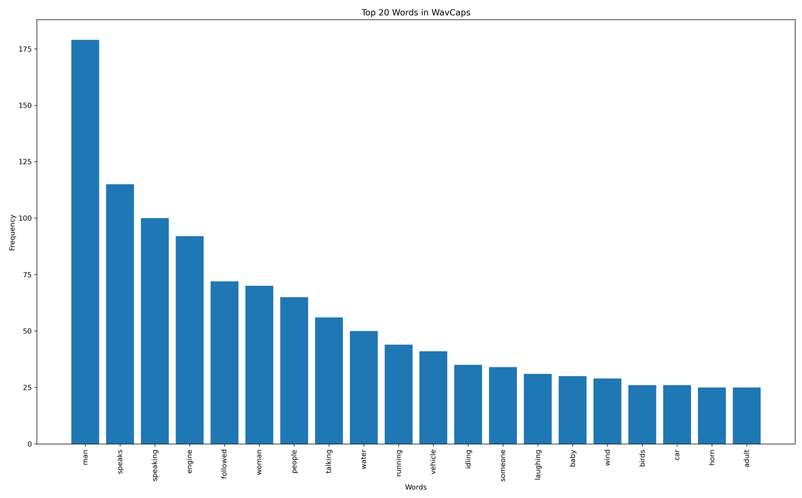


Figure C.4: Top 20 words from the vocabulary of the WavCaps samples (not including stop words)

C.2 Training Details

C.2.1 Implementation

We fine-tuned our models from models available on HuggingFace Diffuser Library, specifically their v1.4 model [Rom+22]. While we initialized our model from the Stable Diffusion checkpoint, the model, training code, and larger pipeline have been heavily modified to fit our purposes. Most notably, we had to make changes to work with non-HuggingFace models, such as the Encodec model[Déf+22], along with changes to the U-Net to adapt to wide-channel inputs. All data for models other than ours in Table C.8 was copied from their respective papers training details as they do not provide *training* code as of the time of writing, only AudioLDM[Liu+23] has public code for loading checkpoints.

For our training, we exclusively trained on individual machines with eight A100 GPUs. We chose most hyperparameters following [Rom+22], with variations to fit our hardware and model. We could use significantly larger batch sizes due to the extremely high compression from the Encodec [Déf+22] model's discrete codebook; we lowered our memory overhead by 56.5% per sample. Pre-computing the codebook codes made these

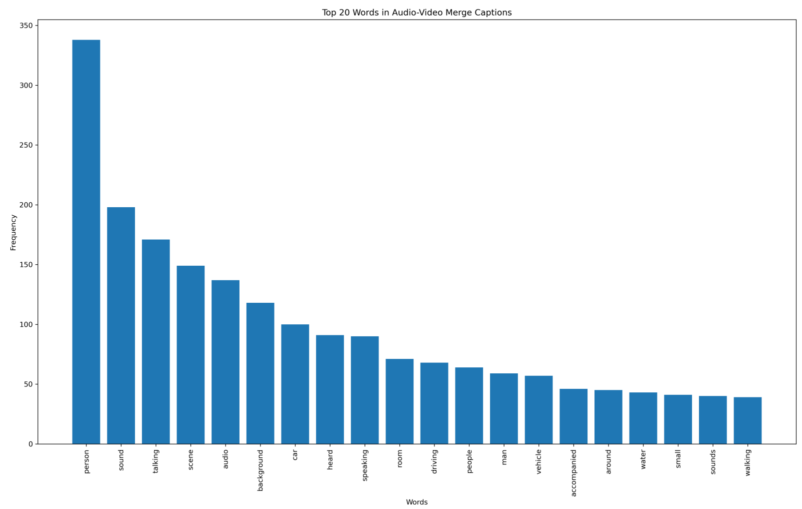


Figure C.5: Top 20 words from the vocabulary of our audio-video merged samples (not including stop words)

gains possible by allowing larger batches and faster training. As Table 4 of the main paper described, we trained multiple models, varying the text encoder with all other parameters and stages remaining the same. However, as will be described in Section C.3, the T5 models required an additional linear projection layer from the length 1028 T5 encoding to the expected 786 input dimension for cross-attention. This extra layer adds parameters to the model and would affect training, but the added parameters are negligible compared to the size of the U-Net. Due to our A100 GPUs are the 40GB version, without model parallel, we could not afford to unfreeze and fine-tune the text encoder together with training the UNet.

C.2.2 Encodec Latent Embedding Space

Figure C.6 displays the Encodec [Déf+22] pipeline through the quantization stage. Perceptually the reconstruction is excellent with a subjective evaluation (MUSHRA score) of 88.0[2]. This figure does not show the entire end-to-end process raw audio to generate raw-audio, as we do not use them in training our model. We initially trained our model on the intermediate discrete stage (RVQ codebook). However, this proved difficult due to the discrete nature of this latent space. While we could easily have trained the model on the pre-quantization latent, this would not have allowed us to leverage the compressed quantized representations as this would have required reading wav files from disk. We

[2]https://en.wikipedia.org/wiki/MUSHRA

Youtube ID	Audio Label	Video Caption	Merged Caption	WavCaps[Mei+23]
BjWf0keANT8	Sine Wave	10,000-hertz sine wave audio frequency.	10,000-hertz audio frequency, represented by a sine wave.	a continuous sine wave sound
83IJft_3Z4E	Throbbing	An ultrasound image of a baby in the womb, proof of new life and potential new beginnings.	An audio scene depicting the throbbing sound of a heartbeat, accompanying the visual of an ultrasound image of a baby in the womb, proof of new life and potential new beginnings.	a heartbeat is being recorded
6hj2F5xvGYE	Male singing	A man is singing into a microphone in front of an American flag.	A male singing into a microphone in front of an American flag, capturing the emotion of the patriotic song.	someone is singing a song

Table C.7: Examples of our Audio-Video merged captions versus WavCaps[Mei+23]

trained our final model on the post-quantization latent representation displayed at the bottom.

C.2.3 Training Instability from Frozen Blocks

One warning from the [Rom+22] paper for fine-tuning their models is catastrophic forgetting. While we faced this issue when training our CLAP models, it did not noticeably affect the training of our T5 models. Our U-Net consists of a few major components: a conv_in block, down blocks, a mid-block, up blocks, and a conv_out block. During experimentation, we attempted to accelerate training and conserve the pre-trained weights of *Stable Diffusion-1-4* by freezing the weights of all blocks other than conv_in and conv_out for 5,000 training steps. This method proved inferior to simply allowing the entire pipeline to learn together in final loss value and training stability. Figure C.8 shows an example training loss graph for one of the aforementioned experiments, clearly displaying the high degree of instability that emerged after the blocks were unfrozen. This instability alone would not be a reason to abandon this technique; however, the more troubling trend was the loss values plateauing around 0.4 compared to 0.19 in our best models.

	Diffusion (Denoising Network)				Classification	
Configuration	DiffSound[Yan+23]	AudioGen[Kre+22]	AudioLDM[Liu+23]	Ours	AS-20K	AS-2M
Optimizer	AdamW	Adam	-	AdamW	AdamW	AdamW
Optimizer β_1 - β_2	0.9 - 0.94	-	-	0.9 - 0.999	0.9 - 0.999	
Base learning rate	3.0e-6	5.0e-4	1.0e-4	2.56e-4	0.001	2e-4
LR schedule	Constant	Inv Sqrt	Constant	CosDecay	CosDecay	CosDecay
Noise schedule	Sc-Linear	-	Sc-Linear	Cosine	-	-
ChannelMultiplier	-	1,2,4,8	1,2,3,5	1,2,4,4	-	-
Diffusion Steps	-	-	1K	1K	-	-
Warm-up epochs	-	-	-	-	1	4
Training Epochs	600	-	-	-	60	10
Warm-up steps	-	3K	-	1K	1K	1K
Training Steps	-	200K	1.5M	40K	-	-
Batch size	16	256	8	192	256	32
GPUs	32	128	1	8	4	4
GPU Type	V100	A100	A100	A100	V100	V100
SpecAug [Par+19]	-	-	-	-	192/48	192/48
Mixup [Zha+18]	-	-	-	-	0.5	0.5
Loss Function	-	ℓ_1,ℓ_2,CE	MSE	MSE	BCE	BCE
Sampler	DDIM[SME20]	-	DDIM	PNDM[Liu+22]	-	-
Sample Steps	100	-	200	45	-	-
Guidance Scale	-	1 - 5	4.5	3.5 - 7.5	-	-
Normalization	-	-	-	Channel	(-4.27, 4.57)	(-4.27, 4.57)

Table C.8: All "-" values are either unknown or not applicable to the given model. All values are from respective papers and appendices sections on training. Inv Sqrt = Inversed Square root; Sc-Linear = Scaled Linear; CosDecay = Cosine with Decay. For normalization we include a "-" if the values are unknown, channel for our per-channel normalization, or (μ, σ) for the dataset.

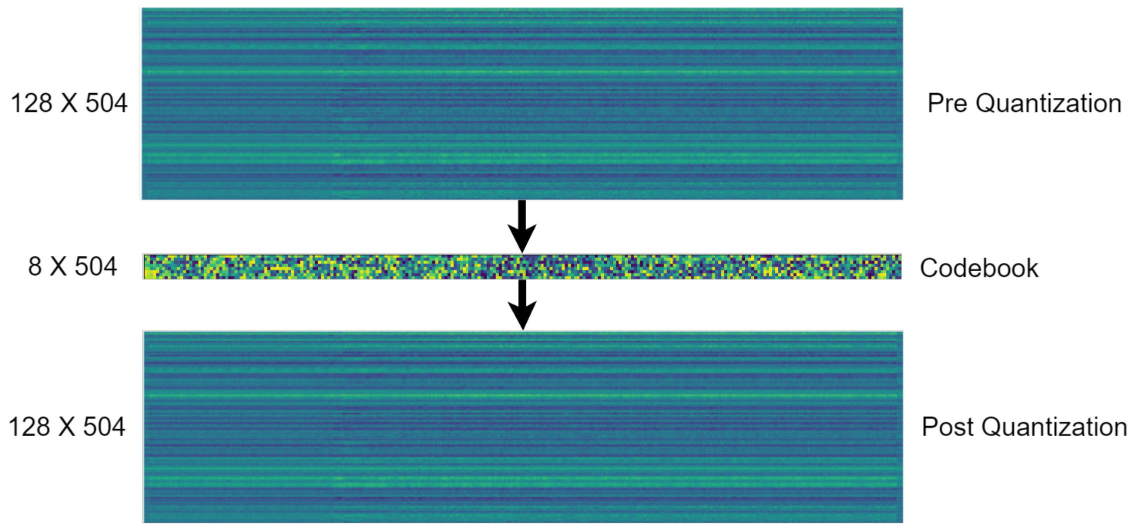


Figure C.6: Encoder-decoder pair not included in the figure. The codebook stage is cropped in the figure to improve visibility.

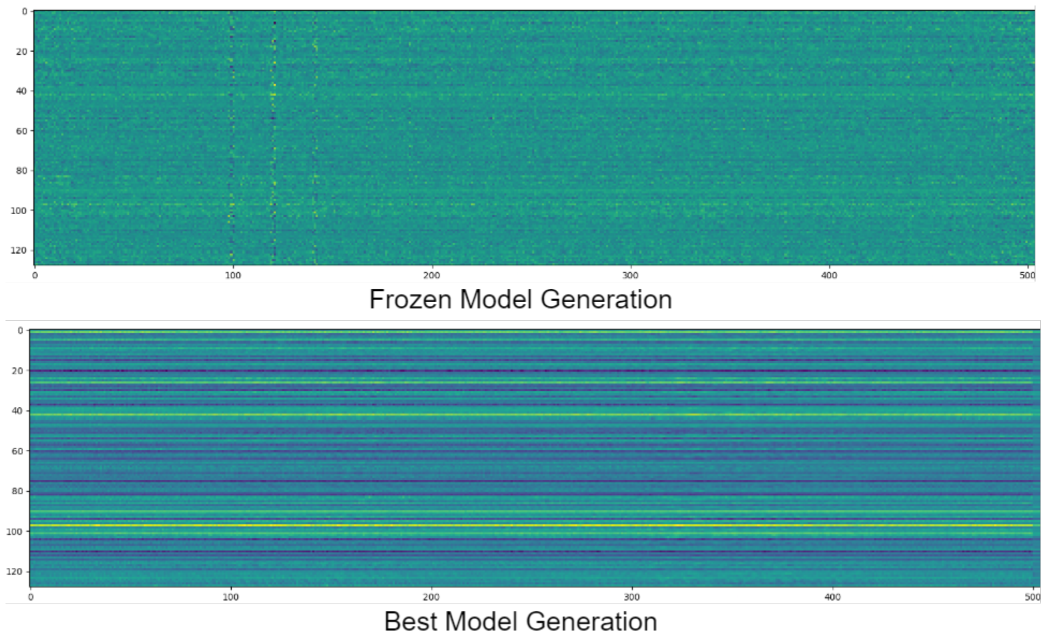


Figure C.7: Comparing this to other examples, such as Figure C.6 clearly shows the lack of quality from frozen models.

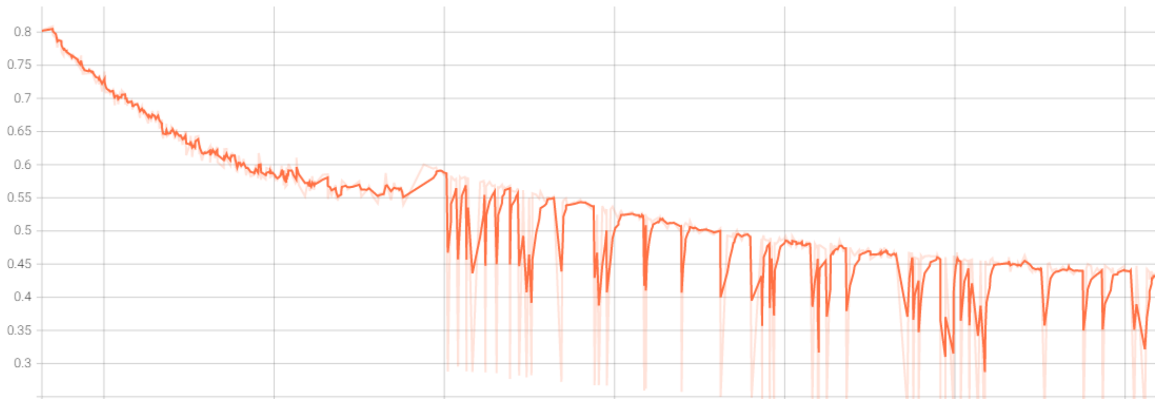


Figure C.8: This model started with all layers other than conv_in and conv_out frozen, then unfroze these blocks after 5,000 training steps.

C.3 Cross Attention Mechanism

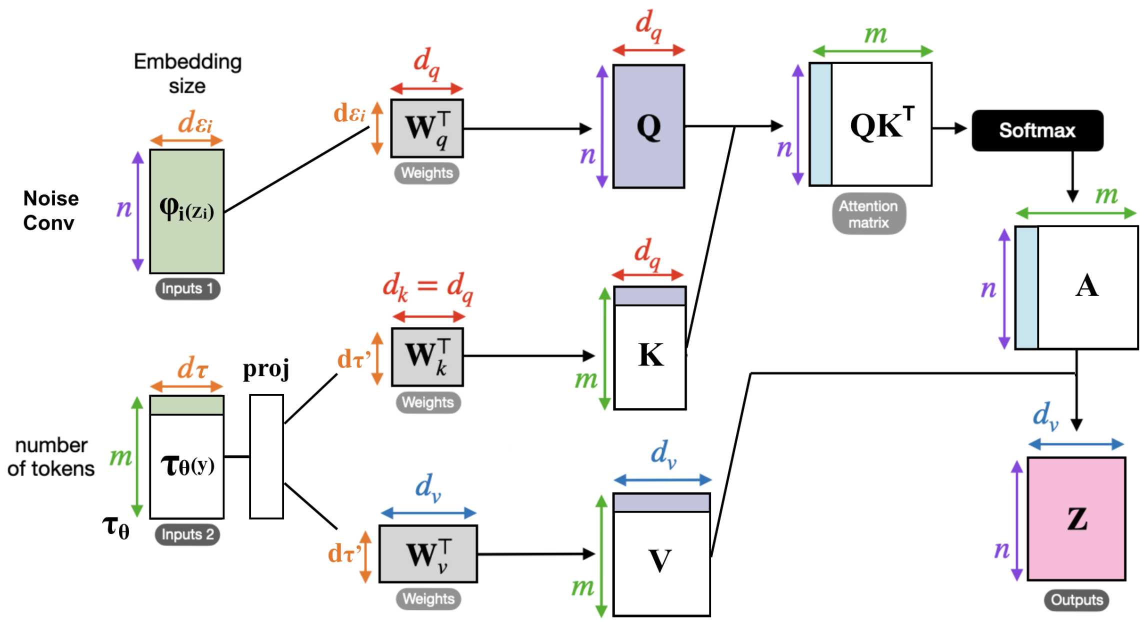


Figure C.9: Illustration of the cross-attention mechanism under masking scenario, the white-colored portions indicate masking.

In Table 4 of our main paper, we observed our *AudioJourney-CLAP* models generally underperform *AudioJourney-T5* models, we believe there could be 2 main reasons:
1) CLAP[Wu*+23] was trained on 660k samples, which is way smaller dataset than what T5[Raf+20] was trained on. Although the CLAP text encoder demonstrated good performance on audio embedding [Wu*+23], T5 may be superior in a general setting.
2) Since CLAP [Wu*+23] output matches with d, we did not adapt its last layer. Using a frozen encoder would solely depend on the W_q, W_k, W_v to learn the mapping, which might be suboptimal.

$$\text{Cross-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d}}\right) \cdot \mathbf{V}$$

where $\mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$

and $\mathbf{W}_Q^{(i)} \in \mathbb{R}^{d_q \times d_\epsilon^i},\ \mathbf{W}_K^{(i)} \in \mathbb{R}^{d_k \times d_{\tau'}}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d_v \times d_{\tau'}},\ \varphi_i(\mathbf{z}_i) \in \mathbb{R}^{n \times d_\epsilon^i},\ \tau_\theta(y) \in \mathbb{R}^{m \times d_\tau}$

in our case: $d_\tau = 1024, d_{\tau'} = 768, d_k = d_v = d_q = d = 768($ initialized from Stable Diffusion [Rom+22] weights),

d_ϵ^i is the i^{th}layer of Unet φ_i's output size

(main paper equation (4))

A surprising result in Table 4 of our main paper is that the masked model performed worse than the unmasked model. This is counter-intuitive, given that attention masking is a common mechanism used to handle variable-length inputs. Figure C.9 explains how masking negatively affects cross-attention in the U-Net. As is illustrated, the white part of the text embedding $\tau_\theta(y)$ indicates the masked-out content because the max sequence length is larger than the number of audio caption tokens. Toward the end, you can see if we pass this mask to the U-Net, this would result in a low-rank dot-product of the attention score A and the V tensor, which result in a low-ranked Z. In an extreme case of when the token length is 1, this is reduced to a rank-1 vector dot product of column by row. We believe this low-ranked representation of Z is suboptimal to the full-ranked version when unmasked.

Therefore, to compensate for the above issue, we reduce our max token length to 50, and use the unmasked versions in our best-performing recipe.

C.4 Additional Samples and Demos

For the overall listening experience, we put our listening samples and spectrogram visualizations to our anonymous website: `https://audiojourney.github.io/` and the code and implementations are at: `https://github.com/audiojourney/audiojourney.github.io`

Our Audio-journey models could serve as the base model similar to Stabel Diffusion [Rom+22] in vision, and it would allow a separate smaller network to learn new concepts from the new dataset (ControlNet) [ZA23], and would also allow finetuning through low-rank approximation like Dreambooth [Rui+22].

Bibliography

[AB79] J.B. Allen and D.A. Berkley. "Image method for efficiently simulating small-room acoustics". In: *Journal of the Acoustical Society of America* 65.4 (1979), pp. 943–950.

[Aba+16] Martın Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[Abu+16] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. "YouTube-8M: A Large-Scale Video Classification Benchmark". In: *arXiv preprint* abs/1609.08675 (2016). arXiv: 1609.08675.

[Alw+19] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. "Self-Supervised Learning by Cross-Modal Audio-Video Clustering". In: *arXiv preprint arXiv:1911.12667* (2019).

[Amo13] Jaume Amores. "Multiple instance classification: Review, taxonomy and comparative study". In: *Artificial Intelligence* 201 (2013), pp. 81–105.

[And+13] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. "Deep canonical correlation analysis". In: *International Conference on Machine Learning*. 2013, pp. 1247–1255.

[And+23] Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. "A modern look at the relationship between sharpness and generalization". In: *arXiv preprint arXiv:2302.07011* (2023).

[Ass+23] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. "Self-supervised learning from images with a joint-embedding predictive architecture". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 15619–15629.

[Ath+17] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples". In: *CoRR* abs/1707.07397 (2017). arXiv: 1707.07397. URL: http://arxiv.org/abs/1707.07397.

[Ath+18] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples". In: *International Conference on Machine Learning*. 2018, pp. 284–293.

[AVT16] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "Soundnet: Learning sound representations from unlabeled video". In: *Advances in Neural Information Processing Systems*. 2016, pp. 892–900.

[AVT17] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "See, Hear, and Read: Deep Aligned Representations". In: *arXiv preprint arXiv:1706.00932* (2017).

[AW18] Aharon Azulay and Yair Weiss. "Why do deep convolutional networks generalize so poorly to small image transformations?" In: *arXiv preprint arXiv:1805.12177* (2018).

[Awa+17] George Awad, Asad Butt, Jonathan Fiscus, David Joy, Andrew Delgado, Martial Michel, Alan F Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, et al. "Trecvid 2017: Evaluating ad-hoc and instance video search, events detection, video captioning and hyperlinking". In: *Proceedings of TRECVID*. 2017.

[Awa+22] George Awad, Keith Curtis, Asad A. Butt, Jonathan Fiscus, Afzal Godil, Yooyoung Lee, Andrew Delgado, Jesse Zhang, Eliot Godard, Baptiste Chocot, Lukas Diduch, Jeffrey Liu, Yvette Graham, and Georges Quénot. "An overview on the evaluated video retrieval tasks at TRECVID 2022". In: *Proceedings of TRECVID 2022*. NIST, USA. 2022.

[AZ17a] Relja Arandjelovic and Andrew Zisserman. "Look, listen and learn". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 609–617.

[AZ17b] Relja Arandjelovic and Andrew Zisserman. "Look, Listen and Learn". In: *CoRR* abs/1705.08168 (2017). arXiv: 1705.08168. URL: http://arxiv.org/abs/1705.08168.

[AZ18] Relja Arandjelovic and Andrew Zisserman. "Objects that Sound". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Vol. 11205. Lecture Notes in Computer Science. Springer, 2018, pp. 451–466. DOI: 10.1007/978-3-030-01246-5_27. URL: https://doi.org/10.1007/978-3-030-01246-5_27.

[Bab08] Boris Babenko. "Multiple instance learning: algorithms and applications". In: *PubMed/NCBI Article* (2008).

[Bae+20] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* 2020.

[Bae+22] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. "data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language". In: *CoRR* abs/2202.03555 (2022). arXiv: 2202.03555. URL: https://arxiv.org/abs/2202.03555.

[BAM18] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal machine learning: A survey and taxonomy". In: *IEEE transactions on pattern analysis and machine intelligence* 41.2 (2018), pp. 423–443.

[BAM19] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. "Effectiveness of self-supervised pre-training for speech recognition". In: *CoRR* abs/1911.03912 (2019). arXiv: 1911.03912. URL: http://arxiv.org/abs/1911.03912.

[Ban+22] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. "Cold diffusion: Inverting arbitrary image transforms without noise". In: *arXiv preprint arXiv:2208.09392* (2022).

[BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[BDG18] Mateusz Budnik, Mikail Demirdelen, and Guillaume Gravier. "A study on multimodal video hyperlinking with visual aggregation". In: *2018 IEEE International Conference on Multimedia and Expo.* IEEE. 2018, pp. 1–6.

[BDW21] Hangbo Bao, Li Dong, and Furu Wei. "BEiT: BERT Pre-Training of Image Transformers". In: *CoRR* abs/2106.08254 (2021). arXiv: 2106.08254. URL: https://arxiv.org/abs/2106.08254.

[Ben+21] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency.* 2021, pp. 610–623.

[Ber+23] Cody Berger, Juncheng B Li, Yiyuan Li, Aaron Berger, Dmitri Berger, Karthik Ganesan, Emma Strubell, and Florian Metze. "Dissecting Efficient Architectures for Wake-Word Detection". In: *Workshop on Efficient Systems for Foundation Models @ ICML2023.* 2023. URL: https://openreview.net/forum?id=lmaAcSViye.

[Bho+21] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. “Understanding robustness of transformers for image classification”. In: *arXiv preprint arXiv:2103.14586* (2021).

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. Springer, 2007. ISBN: 0387310738.

[BOC21] Axel Berg, Mark O’Connor, and Miguel Tairum Cruz. “Keyword Transformer: A Self-Attention Model for Keyword Spotting”. In: *Interspeech 2021*. ISCA. 2021, pp. 4249–4253.

[Boi+17] Rémi Bois, Vedran Vukotić, Anca-Roxana Simon, Ronan Sicre, Christian Raymond, Pascale Sébillot, and Guillaume Gravier. “Exploiting multimodality in video hyperlinking to improve target diversity”. In: *International Conference on Multimedia Modeling*. Springer. 2017, pp. 185–197.

[BPH22] Alan Baade, Puyuan Peng, and David Harwath. “MAE-AST: Masked Autoencoding Audio Spectrogram Transformer”. In: *arXiv preprint arXiv:2203.16691* (2022).

[Bro+17] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. “Adversarial patch”. In: *arXiv preprint arXiv:1712.09665* (2017).

[Bro+20a] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[Bro+20b] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[BSE17] Rohan Badlani, Ankit Shah, and Benjamin Elizalde. *DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System*. Tech. rep. DCASE2017 Challenge, 2017.

[Cak15] Emre Cakir. *et al. "Polyphonic sound event detection using multi label deep neural networks." 2015 international joint conference on neural networks (IJCNN).* 2015.

[Car04] Rich Caruana. *et al. "Ensemble selection from libraries of models."Proceedings of the twenty-first international con-ference on Machine learning*. ACM, 2004.

[Car+16] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. "Hidden voice commands". In: (2016), pp. 513–530.

[Car+19] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. "On evaluating adversarial robustness". In: *arXiv preprint arXiv:1902.06705* (2019).

[Ç+17] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection". In: *IEEE/ACM Trans. on Audio, Speech, and Language Process.* 25.6 (2017). ISSN: 2329-9290. DOI: 10.1109/TASLP.2017.2690575.

[CD11] David L Chen and William B Dolan. "Collecting highly parallel data for paraphrase evaluation". In: *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* ACL. 2011, pp. 190–200.

[CGK15] Miriam Cha, Youngjune Gwon, and HT Kung. "Multimodal sparse representation learning and applications". In: *arXiv preprint arXiv:1511.06238* (2015).

[CH20] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *International conference on machine learning*. PMLR. 2020, pp. 2206–2216.

[Che+20a] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. "Vggsound: A large-scale audio-visual dataset". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2020, pp. 721–725.

[Che+20b] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. "Generative Pretraining From Pixels". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event.* Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1691–1703. URL: http://proceedings.mlr.press/v119/chen20s.html.

[Che+22] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection". In: *arXiv preprint arXiv:2202.00874* (2022).

[Che23] Ting Chen. "On the importance of noise scheduling for diffusion models". In: *arXiv preprint arXiv:2301.10972* (2023).

[CHG22] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. "When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=LtKcMgGOeLt.

[Cho+14] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results". In: *CoRR* abs/1412.1602 (2014). URL: http://arxiv.org/abs/1412.1602.

[Cho15] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[Cho+17] K. Choi, G. Fazekas, M. Sandler, and K. Cho. "Convolutional recurrent neural networks for music classification". In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*. 2017. DOI: 10.1109/ICASSP.2017.7952585.

[Cho+22] Dading Chong, Helin Wang, Peilin Zhou, and Qingcheng Zeng. *Masked Spectrogram Prediction For Self-Supervised Audio Pre-Training*. 2022. arXiv: 2204.12768 [cs.SD].

[Chu+14a] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[Chu+14b] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). URL: http://arxiv.org/abs/1412.3555.

[Cis+17] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. "Parseval networks: Improving robustness to adversarial examples". In: *arXiv preprint arXiv:1704.08847* (2017).

[Cm23] CPU-monkey. *Apple M1 vs Raspberry Pi 4 B (Broadcom BCM2711)*. https://www.cpu-monkey.com/en/compare_cpu-apple_m1-vs-raspberry_pi_4_b_broadcom_bcm2711. 2023.

[CMW04] Christopher Cieri, David Miller, and Kevin Walker. “The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text.” In: *LREC*. Vol. 4. 2004, pp. 69–71.

[Con22] PyTorch Contributors. *QUANTIZATION*. https://pytorch.org/docs/stable/quantization.html. 2022.

[Cou+19] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. “Efficient keyword spotting using dilated convolutions and gating”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6351–6355.

[Cow15] Nelson Cowan. “George Miller’s magical number of immediate memory in retrospect: Observations on the faltering progression of science.” In: *Psychological review* 122.3 (2015), p. 536.

[CP18] Jingze Chi and Yuxin Peng. “Dual Adversarial Networks for Zero-shot Cross-media Retrieval.” In: *International Joint Conferences on Artificial Intelligence*. 2018, pp. 663–669.

[CRK19] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *arXiv preprint arXiv:1902.02918* (2019).

[CTK+22] Nicholas Carlini, Florian Tramer, J Zico Kolter, et al. “(Certified!!) Adversarial Robustness for Free!” In: *arXiv preprint arXiv:2206.10550* (2022).

[CW18] Nicholas Carlini and David Wagner. “Audio adversarial examples: Targeted attacks on speech-to-text”. In: *arXiv preprint arXiv:1801.01944* (2018).

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. “An Empirical Study of Training Self-Supervised Vision Transformers”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9620–9629. DOI: 10.1109/ICCV48922.2021.00950. URL: https://doi.org/10.1109/ICCV48922.2021.00950.

[CZ17a] João Carreira and Andrew Zisserman. “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502. URL: https://doi.org/10.1109/CVPR.2017.502.

[CZ17b] Joao Carreira and Andrew Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2017, pp. 4724–4733.

[Dai+16] Wei Dai, Juncheng Li, Phuong Pham, Samarjit Das, and Shuhui Qu. *Acoustic Scene Recognition with Deep Neural Networks (DCASE Challenge 2016).* Tech. rep. DCASE2016 Challenge, Sept. 2016.

[Dai+17] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. "Very deep convolutional neural networks for raw waveforms". In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE. 2017, pp. 421–425.

[Déf+22] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. "High fidelity neural audio compression". In: *arXiv preprint arXiv:2210.13438* (2022).

[Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: (2009).

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.2011 (2011), pp. 2121–2159.

[DLS16] Jianfeng Dong, Xirong Li, and Cees GM Snoek. "Word2VisualVec: Image and Video to Sentence Matching by Visual Feature Prediction". In: *arXiv preprint arXiv:1604.06838* (2016).

[DN21] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[Dos+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[DT05] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177. URL: https://doi.org/10.1109/CVPR.2005.177.

[Du+20] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. "Sirenattack: Generating adversarial audio for end-to-end acoustic systems". In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 2020, pp. 357–369.

[DV15] Yann N. Dauphin, Harm de Vries, and Yoshua. "Equilibrated Adaptive Learning Rates for Non-convex Optimization". In: NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1504–1512. URL: http://dl.acm.org/citation.cfm?id=2969239.2969407.

[Ebr+18] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. "Hotflip: White-box adversarial examples for text classification". In: 2 (2018), pp. 31–36.

[Eng+17] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. "A rotation and a translation suffice: Fooling cnns with simple transformations". In: *arXiv preprint arXiv:1712.02779* (2017).

[ESS16] Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. "Dissimilarity-based sparse subset selection". In: *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2016), pp. 2182–2197.

[Evt+18] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. "Robust physical-world attacks on deep learning models". In: *CVPR* 1 (2018).

[EWS10] Florian Eyben, Martin Wöllmer, and Björn Schuller. "Opensmile: The Munich Versatile and Fast Open-source Audio Feature Extractor". In: *Proceedings of the 18th ACM International Conference on Multimedia*. MM '10. Firenze, Italy: ACM, 2010, pp. 1459–1462. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874246. URL: http://doi.acm.org/10.1145/1873951.1874246.

[EZ+16] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. *CP-JKU Submissions for DCASE-2016: a Hybrid Approach Using Binaural I-Vectors and Deep Convolutional Neural Networks*. Tech. rep. DCASE2016 Challenge, Sept. 2016.

[Fag+18] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. "VSE++: Improved Visual-Semantic Embeddings". In: *British Machine Vision Conference (BMVC)* (2018).

[Far+10] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. "Every picture tells a story: Generating sentences from images". In: *European Conference on Computer Vision*. Springer. 2010, pp. 15–29.

[For+19] Logan Ford, Hao Tang, François Grondin, and James Glass. "A Deep Residual Network for Large-Scale Acoustic Scene Analysis". In: *Proc. Interspeech 2019*. 2019, pp. 2568–2572. DOI: `10.21437/Interspeech.2019-2731`. URL: `http://dx.doi.org/10.21437/Interspeech.2019-2731`.

[For+20] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. "Sharpness-aware minimization for efficiently improving generalization". In: *arXiv preprint arXiv:2010.01412* (2020).

[FPW16] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. "Spatiotemporal Residual Networks for Video Action Recognition". In: *NIPS*. 2016.

[Fra+12] Muhammad Moazam Fraz, Paolo Remagnino, Andreas Hoppe, Bunyarit Uyyanonvara, Alicja R Rudnicka, Christopher G Owen, and Sarah A Barman. "An ensemble classification-based approach applied to retinal blood vessel segmentation". In: *IEEE Transactions on Biomedical Engineering* 59.9 (2012), pp. 2538–2548.

[Fro+13] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. "Devise: A deep visual-semantic embedding model". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2121–2129.

[FWL14] Fangxiang Feng, Xiaojie Wang, and Ruifan Li. "Cross-modal retrieval with correspondence autoencoder". In: *ACM Multimedia Conference*. ACM. 2014, pp. 7–16.

[Gal+22] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: *arXiv preprint arXiv:2208.01618* (2022).

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Aistats*. Vol. 9. 2010, pp. 249–256.

[GCG21a] Yuan Gong, Yu-An Chung, and James Glass. "AST: Audio Spectrogram Transformer". In: *arXiv preprint arXiv:2104.01778* (2021). Ed. by Hynek Hermansky, Honza Cernocký, Lukás Burget, Lori Lamel, Odette Scharenborg, and Petr Motlícek, pp. 571–575. DOI: `10.21437/Interspeech.2021-698`. URL: `https://doi.org/10.21437/Interspeech.2021-698`.

[GCG21b] Yuan Gong, Yu-An Chung, and James Glass. “Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3292–3306. DOI: 10.1109/TASLP.2021.3120633.

[Gem+17a] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. “Audio Set: An ontology and human-labeled dataset for audio events”. In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*. 2017.

[Gem+17b] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. “Audio set: An ontology and human-labeled dataset for audio events”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 776–780.

[GHM92] John J Godfrey, Edward C Holliman, and Jane McDaniel. “SWITCHBOARD: Telephone speech corpus for research and development”. In: *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 1992, pp. 517–520.

[GL84] D. Griffin and Jae Lim. “Signal estimation from modified short-time Fourier transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243. DOI: 10.1109/TASSP.1984.1164317.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. “Speech Recognition with Deep Recurrent Neural Networks”. In: *CoRR* abs/1303.5778 (2013). URL: http://arxiv.org/abs/1303.5778.

[Gol+15a] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. “Convolutional neural networks for acoustic modeling of raw time signal in lvcsr”. In: *Interspeech*. 2015.

[Gol+15b] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. “Convolutional neural networks for acoustic modeling of raw time signal in lvcsr”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[Gon+14] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. “A multi-view embedding space for modeling internet images, tags, and their semantics”. In: *International Journal of Computer Vision* 106.2 (2014), pp. 210–233.

[Gon+21] Yuan Gong, Cheng-I Lai, Yu-An Chung, and James R. Glass. “SSAST: Self-Supervised Audio Spectrogram Transformer”. In: *ArXiv* abs/2110.09784 (2021).

[Gon+22a] Yuan Gong, Sameer Khurana, Andrew Rouditchenko, and James Glass. *CMKD: CNN/Transformer-Based Cross-Model Knowledge Distillation for Audio Classification.* 2022. DOI: 10.48550/ARXIV.2203.06760. URL: https://arxiv.org/abs/2203.06760.

[Gon+22b] Yuan Gong, Andrew Rouditchenko, Alexander H Liu, David Harwath, Leonid Karlinsky, Hilde Kuehne, and James Glass. "Contrastive Audio-Visual Masked Autoencoder". In: *arXiv preprint arXiv:2210.07839* (2022).

[GTM19] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. "Large-Scale Weakly-Supervised Pre-Training for Video Action Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE, 2019, pp. 12046–12055. DOI: 10.1109/CVPR.2019.01232.

[Guo+18] Jinxi Guo, Kenichi Kumatani, Ming Sun, Minhua Wu, Anirudh Raju, Nikko Ström, and Arindam Mandal. "Time-delayed bottleneck highway networks using a dft feature for keyword spotting". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2018, pp. 5489–5493.

[Han+14a] Awni Hannun et al. "Deep speech: Scaling up end-to-end speech recognition". In: *arXiv preprint arXiv:1412.5567* (2014).

[Han+14b] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition". In: *CoRR* abs/1412.5567 (2014). URL: http://arxiv.org/abs/1412.5567.

[He+15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[He+15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778.

[He+16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[He+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2016, pp. 770–778.

[HE17] Christian Andreas Henning and Ralph Ewerth. “Estimating the information gap between textual and visual representations”. In: *International Conference on Multimedia Retrieval.* ACM. 2017, pp. 14–22.

[He+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975. URL: https://doi.org/10.1109/CVPR42600.2020.00975.

[He+21a] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked autoencoders are scalable vision learners”. In: *arXiv preprint arXiv:2111.06377* (2021).

[He+21b] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked autoencoders are scalable vision learners”. In: *arXiv preprint arXiv:2111.06377* (2021).

[Her+17a] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. “CNN Architectures for Large-Scale Audio Classification”. In: *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP).* 2017.

[Her+17b] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. “CNN Architectures for Large-Scale Audio Classification”. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2017. URL: https://arxiv.org/abs/1609.09430.

[Her+21] Shawn Hershey, Daniel PW Ellis, Eduardo Fonseca, Aren Jansen, Caroline Liu, R Channing Moore, and Manoj Plakal. “The Benefit of Temporally-Strong Labels in Audio Event Classification”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2021, pp. 366–370.

[HGD19] Kaiming He, Ross Girshick, and Piotr Dollár. “Rethinking imagenet pre-training”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 4918–4927.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[HMV16] Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. *DCASE2016 Baseline System*. Tech. rep. DCASE2016 Challenge, Sept. 2016.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[HSST04] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. "Canonical correlation analysis: An overview with application to learning methods". In: *Neural Computation* 16.12 (2004), pp. 2639–2664.

[Hsu+21] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3451–3460.

[Hua+17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[Hua+18] Po-Yao Huang, Junwei Liang, Jean-Baptiste Lamare, and Alexander G Hauptmann. "Multimodal Filtering of Social Media for Temporal Monitoring and Event Analysis". In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM. 2018, pp. 450–457.

[HWW15] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. "Speech acoustic modeling from raw multichannel waveforms". In: *ICASSP*. IEEE. 2015, pp. 4624–4628.

[HWW17] Yan Huang, Wei Wang, and Liang Wang. "Instance-aware image and sentence matching with selective multimodal lstm". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2017, pp. 2310–2318.

[HYH13] Micah Hodosh, Peter Young, and Julia Hockenmaier. "Framing image description as a ranking task: Data, models and evaluation metrics". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 853–899.

[IC14] Ozan Irsoy and Claire Cardie. "Opinion Mining with Deep Recurrent Neural Networks". In: EMNLP. Citeseer, 2014.

[IEM18] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. "Prior convictions: Black-box adversarial attacks with bandits and priors". In: *arXiv preprint arXiv:1807.07978* (2018).

[Ily+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. “Adversarial examples are not bugs, they are features”. In: *arXiv preprint arXiv:1905.02175* (2019).

[IS15a] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[IS15b] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[Ito17] Keith Ito. *The LJ Speech Dataset*. https://keithito.com/LJ-Speech-Dataset/. 2017.

[Iyy+18] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. “Adversarial example generation with syntactically controlled paraphrase networks”. In: *arXiv preprint arXiv:1804.06059* (2018).

[Izm+18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. “Averaging weights leads to wider optima and better generalization”. In: *arXiv preprint arXiv:1803.05407* (2018).

[JLD19] Matt Jordan, Justin Lewis, and Alexandros G Dimakis. “Provable certificates for adversarial examples: Fitting a ball in the union of polytopes”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 14082–14092.

[Jor] Michael Jorgensen. “VOLUMES OF n-DIMENSIONAL SPHERES AND ELLIPSOIDS”. In: “https://www.whitman.edu/documents/Academics/Mathematics/2014/jorgenmd.pdf".

[JQG22] Jinyuan Jia, Wenjie Qu, and Neil Gong. “MultiGuard: Provably Robust Multi-label Classification against Adversarial Examples”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 10150–10163.

[JS83] David A Jaffe and Julius O Smith. “Extensions of the Karplus-Strong plucked-string algorithm”. In: *Computer Music Journal* 7.2 (1983), pp. 56–69.

[Kay+17] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[KB14a] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[KB14b] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* abs/1412.6980 (2014). arXiv: 1412.6980.

[KB14c] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[KBD05] Patrick Kenny, Gilles Boulianne, and Pierre Dumouchel. “Eigenvoice modeling with sparse training data”. In: *IEEE transactions on speech and audio processing* 13.3 (2005), pp. 345–354.

[KCL23] Simran Kaur, Jeremy Cohen, and Zachary Chase Lipton. “On the maximum hessian eigenvalue and generalization”. In: *Proceedings on*. PMLR. 2023, pp. 51–65.

[Kes+16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. “On large-batch training for deep learning: Generalization gap and sharp minima”. In: *arXiv preprint arXiv:1609.04836* (2016).

[KFF15] Andrej Karpathy and Li Fei-Fei. “Deep visual-semantic alignments for generating image descriptions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2015, pp. 3128–3137.

[Kim+19] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. “AudioCaps: Generating Captions for Audios in The Wild”. In: *NAACL-HLT*. 2019.

[Kir+15] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. “Skip-thought vectors”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 3294–3302.

[KJL14] Andrej Karpathy, Armand Joulin, and Fei Fei F Li. “Deep fragment embeddings for bidirectional image sentence mapping”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 1889–1897.

[KKB20] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. “Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17022–17033.

[Kle+15] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. “Associating neural word embeddings with deep image representations using fisher vectors”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2015, pp. 4437–4446.

[Kon+18] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D Plumbley. “Audio set classification with attention model: A probabilistic perspective”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 316–320.

[Kon+19a] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition”. In: *arXiv preprint arXiv:1912.10211* 28 (2019), pp. 2880–2894. DOI: 10.1109/TASLP.2020.3030497. URL: https://doi.org/10.1109/TASLP.2020.3030497.

[Kon+19b] Qiuqiang Kong, Changsong Yu, Yinlong Xu, Turab Iqbal, Wenwu Wang, and Mark D. Plumbley. “Weakly Labelled AudioSet Tagging With Attention Neural Networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27 (2019), pp. 1791–1802.

[Kon+20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. “Diffwave: A versatile diffusion model for audio synthesis”. In: *arXiv preprint arXiv:2009.09761* (2020).

[Kos21] Aleksandar Kostovic. “Apple M1’s ‘Small’ Icestorm Cores Benchmarked Against ’Big’ Firestorm Cores”. In: (2021).

[Kou+21a] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. “Efficient Training of Audio Transformers with Patchout”. In: *CoRR* abs/2110.05069 (2021). arXiv: 2110.05069. URL: https://arxiv.org/abs/2110.05069.

[Kou+21b] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. *Efficient Training of Audio Transformers with Patchout.* 2021. arXiv: 2110.05069 [cs.SD].

[KR16] Anurag Kumar and Bhiksha Raj. “Audio Event Detection Using Weakly Labeled Data”. In: *Proc. of the ACM Conf. on Multimedia (MM)*. Amsterdam, The Netherlands, 2016. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2964310.

[Kre+22] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. “Audiogen: Textually guided audio generation”. In: *arXiv preprint arXiv:2209.15352* (2022).

[Kri+20] Raghuraman Krishnamoorthi, James Reed, Min Ni, Chris Gottbrath, and Seth Weidman. *Introduction to Quantization on PyTorch*. https://pytorch.org/blog/introduction-to-quantization-on-pytorch/. 2020.

[KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger. 2012, pp. 1106–1114. URL: http://books.nips.cc/papers/files/nips25/NIPS2012_0534.pdf.

[KSH12b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *NIPS*. 2012, pp. 1097–1105.

[KSZ14] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. “Unifying visual-semantic embeddings with multimodal neural language models”. In: *arXiv preprint arXiv:1411.2539* (2014).

[Kum+17] Kenichi Kumatani, Sankaran Panchapagesan, Minhua Wu, Minjae Kim, Nikko Strom, Gautam Tiwari, and Arindam Mandai. “Direct modeling of raw audio with dnns for wake word detection”. In: (2017), pp. 252–257.

[Lee97] Joon Ho Lee. “Analyses of multiple evidence combination”. In: *ACM SIGIR Forum*. Vol. 31. SI. ACM. 1997, pp. 267–276.

[LH17] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[Li+17] Juncheng Li, Wei Dai, Florian Metze, Shuhui Qu, and Samarjit Das. “A comparison of deep learning methods for environmental sound detection”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 126–130.

[Li18] Jerry Zheng Li. “Principled approaches to robust machine learning and beyond”. PhD thesis. Massachusetts Institute of Technology, 2018.

[Li+18] Juncheng Li, Yun Wang, Joseph Szurley, Florian Metze, and Samarjit Das. “A Light-Weight Multimodal Framework for Improved Environmental Audio Tagging”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 6832–6836.

[Li+19] Juncheng Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J Zico Kolter, and Florian Metze. “Adversarial music: Real world audio adversary against wake-word detection system”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[Li+21] Juncheng B Li, Kaixin Ma, Shuhui Qu, Po-Yao Huang, and Florian Metze. “Audio-visual event recognition through the lens of adversary”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 616–620.

[Li+22a] Juncheng B Li, Shuhui Qu, Po-Yao Huang, and Florian Metze. *AudioTagging Done Right: 2nd comparison of deep learning methods for environmental sound classification*. 2022. DOI: 10.48550/ARXIV.2203.13448. URL: https://arxiv.org/abs/2203.13448.

[Li+22b] Juncheng B Li, Shuhui Qu, Xinjian Li, Po-Yao Bernie Huang, and Florian Metze. "On adversarial robustness of large-scale audio visual learning". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 231–235.

[Li+22c] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. "MViTv2: Improved multiscale vision transformers for classification and detection". In: *CVPR*. 2022.

[Li+23a] Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. *Your Diffusion Model is Secretly a Zero-Shot Classifier*. 2023. arXiv: 2303.16203 [cs.LG].

[Li+23b] Juncheng B Li, Jackson Sam Michaels, Laura Yao, Lijun Yu, Zach Wood-Doughty, and Florian Metze. "Audio-Journey: Efficient Visual+LLM-aided Audio Encodec Diffusion". In: *Workshop on Efficient Systems for Foundation Models @ ICML2023*. 2023. URL: https://openreview.net/forum?id=vzMXsTCdFB.

[Li+23c] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models". In: *arXiv preprint arXiv:2301.12597* (2023).

[Liu+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[Liu+20] Andy T. Liu, Shu-Wen Yang, Po-Han Chi, Po-chun Hsu, and Hung-yi Lee. "Mockingjay: Unsupervised Speech Representation Learning with Deep Bidirectional Transformer Encoders". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 6419–6423. DOI: 10.1109/ICASSP40776.2020.9054458. URL: https://doi.org/10.1109/ICASSP40776.2020.9054458.

[Liu+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9992–10002. DOI: 10.1109/ICCV48922.2021.00986. URL: https://doi.org/10.1109/ICCV48922.2021.00986.

[Liu+22] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. “Pseudo numerical methods for diffusion models on manifolds”. In: *arXiv preprint arXiv:2202.09778* (2022).

[Liu+23] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. “AudioLDM: Text-to-Audio Generation with Latent Diffusion Models”. In: 2023. arXiv: 2301.12503 [cs.SD].

[LLY20] Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. “Adversarial Vertex Mixup: Toward Better Adversarially Robust Generalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[LQM22] Juncheng B Li, Shuhui Qu, and Florian Metze. “Robustness of Neural Architectures for Audio Event Detection”. In: *arXiv preprint arXiv:2205.03268* (2022).

[LQM+22] Juncheng B Li, Shuhui Qu, Florian Metze, et al. “AudioTagging Done Right: 2nd comparison of deep learning methods for environmental sound classification”. In: *arXiv preprint arXiv:2203.13448* (2022).

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[LSK19] Juncheng Li, Frank R Schmidt, and J Zico Kolter. “Adversarial camera stickers: A physical camera-based attack on deep learning systems”. In: *arXiv preprint arXiv:1904.00759* 32 (2019).

[Lu+22] Yen-Ju Lu, Zhong-Qiu Wang, Shinji Watanabe, Alexander Richard, Cheng Yu, and Yu Tsao. “Conditional diffusion probabilistic model for speech enhancement”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7402–7406.

[Mad+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[Man+17] R Manmatha, Chao-Yuan Wu, Alexander J Smola, and Philipp Krahenbuhl. "Sampling Matters in Deep Embedding Learning". In: *IEEE International Conference on Computer Vision*. IEEE. 2017, pp. 2859–2867.

[Man22] Valery Manokhin. *Awesome Conformal Prediction*. Version v1.0.0. "If you use Awesome Conformal Prediction, please cite it as below.". Apr. 2022. DOI: 10.5281/zenodo.6467205. URL: https://doi.org/10.5281/zenodo.6467205.

[McF+15] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi YAMAMOTO, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty. *librosa: 0.4.1*. Oct. 2015. DOI: 10.5281/zenodo.32193. URL: https://doi.org/10.5281/zenodo.32193.

[MD+17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Universal adversarial perturbations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Ieee. 2017, pp. 1765–1773.

[Meh+21] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. "An empirical investigation of the role of pre-training in lifelong learning". In: *arXiv preprint arXiv:2112.09153* (2021).

[Mei+23] Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D Plumbley, Yuexian Zou, and Wenwu Wang. "WavCaps: A ChatGPT-Assisted Weakly-Labelled Audio Captioning Dataset for Audio-Language Multimodal Research". In: *arXiv preprint arXiv:2303.17395* (2023).

[MHV16a] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. "Metrics for Polyphonic Sound Event Detection". In: *Applied Sciences* 6.6 (2016), p. 162. ISSN: 2076-3417. DOI: 10.3390/app6060162. URL: http://www.mdpi.com/2076-3417/6/6/162.

[MHV16b] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. "TUT database for acoustic scene classification and sound event detection". In: *Proc. of the European Signal Process. Conf. (EUSIPCO)*. 2016.

[Mic+21] Daniel Michelsanti, Zheng-Hua Tan, Shi-Xiong Zhang, Yong Xu, Meng Yu, Dong Yu, and Jesper Jensen. "An overview of deep-learning-based audio-visual speech enhancement and separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2021).

[Mik+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[Mit04] Joan L Mitchell. “Introduction to digital audio coding and standards”. In: *Journal of Electronic Imaging* 13.2 (2004), p. 399.

[Mit+17] Niluthpol C Mithun, Juncheng B Li, Florian Metze, Amit K Roy-Chowdhury, and Das Samarjit. “CMU-UCR-BOSCH TRECVID 2017: Video to Text Retrieval”. In: *TRECVID 2017 Workshop*. in Proceedings from NIST, 2017.

[Mit+18a] Niluthpol Chowdhury Mithun, Juncheng Li, Florian Metze, and Amit K Roy-Chowdhury. “Learning Joint Embedding with Multimodal Cues for Cross-Modal Video-Text Retrieval”. In: *ACM International Conference on Multimedia Retrieval*. 2018.

[Mit+18b] Niluthpol Chowdhury Mithun, Sirajum Munir, Karen Guo, and Charles Shelton. “ODDS: real-time object detection using depth sensors on embedded GPUs”. In: *ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE Press. 2018, pp. 230–241.

[Mit+18c] Niluthpol Chowdhury Mithun, Panda Rameswar, Evangelos Papalexakis, and Amit Roy-Chowdhury. “Webly Supervised Joint Embedding for Cross-Modal Image-Text Retrieval”. In: *ACM International Conference on Multimedia*. 2018.

[Mit+19] Niluthpol C Mithun, Juncheng Li, Florian Metze, and Amit K Roy-Chowdhury. “Joint embeddings with multimodal cues for video-text retrieval”. In: *International Journal of Multimedia Information Retrieval* 8.1 (2019), pp. 3–18.

[Mit+20] Simon Mittermaier, Ludwig Kürzinger, Bernd Waschneck, and Gerhard Rigoll. “Small-footprint keyword spotting on raw audio data with sinc-convolutions”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7454–7458.

[MLF15] Zhuang Ma, Yichao Lu, and Dean Foster. “Finding linear structure in large datasets with scalable canonical correlation analysis”. In: *International Conference on Machine Learning*. 2015, pp. 169–178.

[Moo+23] R Channing Moore, Daniel PW Ellis, Eduardo Fonseca, Shawn Hershey, Aren Jansen, and Manoj Plakal. “Dataset Balancing Can Hurt Model Performance”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[MPRC16] Niluthpol Chowdhury Mithun, Rameswar Panda, and Amit K Roy-Chowdhury. "Generating diverse image datasets with limited labeling". In: *ACM Multimedia Conference.* ACM. 2016, pp. 566–570.

[Nag+20] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. "Voxceleb: Large-scale speaker verification in the wild". In: *Comput. Speech Lang.* 60 (2020).

[Nag+21a] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. "Attention Bottlenecks for Multimodal Fusion". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual.* Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan. 2021, pp. 14200–14213. URL: https://proceedings.neurips.cc/paper/2021/hash/76ba9f564ebbc35b1014ac498fafadd0-Abstract.html.

[Nag+21b] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. "Attention bottlenecks for multimodal fusion". In: *arXiv preprint arXiv:2107.00135* (2021).

[Nai+18] Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. "Stress Test Evaluation for Natural Language Inference". In: *arXiv preprint arXiv:1806.00692* (2018).

[Nas+21] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Khan, and Ming-Hsuan Yang. "Intriguing Properties of Vision Transformers". In: *Advances in Neural Information Processing Systems.* Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021. URL: https://openreview.net/forum?id=o2mbl-Hmfgd.

[NHK17] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. "Dual Attention Networks for Multimodal Reasoning and Matching". In: *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2017, pp. 299–307.

[Nii+22] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. "Masked Spectrogram Modeling using Masked Autoencoders for Learning General-purpose Audio Representation". In: *arXiv:2204.12260* (2022). arXiv: 2204.12260 [eess.AS]. URL: https://arxiv.org/abs/2204.12260.

[NP33] Jerzy Neyman and Egon Sharpe Pearson. "IX. On the problem of the most efficient tests of statistical hypotheses". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231.694-706 (1933), pp. 289–337.

[Nvi22] Nvidia. *NVIDA CUDA Programming Guide*. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. 2022.

[OLV18a] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[OLV18b] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[Ota+16] Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. “Learning joint representations of videos and sentences with web image search”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 651–667.

[Owe+16] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H Adelson, and William T Freeman. “Visually indicated sounds”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2016, pp. 2405–2413.

[Pan+16a] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. “Jointly modeling embedding and translation to bridge video and language”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2016, pp. 4594–4602.

[Pan+16b] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. “Multi-Task Learning and Weighted Cross-Entropy for DNN-Based Keyword Spotting.” In: (2016), pp. 760–764.

[Pap+16] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. “Crafting adversarial input sequences for recurrent neural networks”. In: (2016), pp. 49–54.

[Pap+18] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. “SoK: Security and privacy in machine learning”. In: (2018), pp. 399–414.

[Par+19] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”. In: *ArXiv* abs/1904.08779 (2019).

[Pas+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[Pat+16] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. "Context Encoders: Feature Learning by Inpainting". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2536–2544. DOI: 10.1109/CVPR.2016.278. URL: https://doi.org/10.1109/CVPR.2016.278.

[Pat+21] Mandela Patrick, Po-Yao Huang, Ishan Misra, Florian Metze, Andrea Vedaldi, Yuki M. Asano, and João F. Henriques. "Space-Time Crop & Attend: Improving Cross-modal Video Representation Learning". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 10540–10552. DOI: 10.1109/ICCV48922.2021.01039. URL: https://doi.org/10.1109/ICCV48922.2021.01039.

[PC21] Sayak Paul and Pin-Yu Chen. "Vision transformers are robust learners". In: *arXiv preprint arXiv:2105.07581* (2021).

[Pic15a] Karol J Piczak. "Environmental sound classification with convolutional neural networks". In: *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2015, pp. 1–6.

[Pic15b] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd Annual ACM Conference on Multimedia*. Brisbane, Australia: ACM Press, 2015, pp. 1015–1018. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806390. URL: http://dl.acm.org/citation.cfm?doid=2733373.2806390.

[Pol06] Robi Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[Pol07] Robi Polikar. "Bootstrap inspired techniques in computational intelligence: ensemble of classifiers, incremental learning, data fusion and missing features". In: *IEEE Signal Processing Magazine* 24.4 (2007), pp. 59–72.

[Pov+11a] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. "The Kaldi speech recognition toolkit". In: *IEEE 2011 workshop on automatic speech recognition and understanding*. CONF. IEEE Signal Processing Society. 2011.

[Pov+11b] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Nagendra Goel, Mirko Hannemann, Yanmin Qian, Petr Schwarz, and Georg Stemmer. "The kaldi speech recognition toolkit". In: *IEEE 2011 workshop*. 2011.

[Poy+22] Huang Poyao, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, Christoph Feichtenhofer, et al. “Masked autoencoders that listen”. In: *arXiv preprint arXiv:2207.06405* (2022).

[PRP22] David Peter, Wolfgang Roth, and Franz Pernkopf. “End-to-end keyword spotting using neural architecture search and quantization”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 3423–3427.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. ACL. 2014, pp. 1532–1543.

[PXS17] Romain Paulus, Caiming Xiong, and Richard Socher. “A Deep Reinforced Model for Abstractive Summarization”. In: *arXiv* abs/1705.04304 (2017).

[Qin+19] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel. “Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition”. In: *arXiv e-prints* (Mar. 2019). arXiv: 1903.10346.

[QNTLBC16] Thi Quynh Nhi Tran, Hervé Le Borgne, and Michel Crucianu. “Aggregating image and text quantized correlated components”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2046–2054.

[Rad+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[Rad+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[Raf+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[Ram+16] Vasili Ramanishka, Abir Das, Dong Huk Park, Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, and Kate Saenko. “Multimodal video description”. In: *ACM Multimedia Conference*. ACM. 2016, pp. 1092–1096.

[Ram+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8821–8831. URL: http://proceedings.mlr.press/v139/ramesh21a.html.

[RDE12] Anthony Rousseau, Paul Deléglise, and Yannick Esteve. "TED-LIUM: an Automatic Speech Recognition dedicated corpus." In: *LREC*. 2012, pp. 125–129.

[Red+16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2016, pp. 779–788.

[Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[RK16] Sravana Reddy and Kevin Knight. "Obfuscating gender in social media writing". In: (2016), pp. 17–26.

[Rom+13] Gerard Roma, Waldo Nogueira, Perfecto Herrera, and Roc de Boronat. "Recurrence quantification analysis features for auditory scene classification". In: *DCASE Challenge, Tech. Rep* (2013).

[Rom+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[RT21] Rahul Rahaman and Alexandre H Thiery. "Uncertainty quantification and deep ensembles". In: *Advances in Neural Information Processing Systems* (2021).

[Rui+22] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: *arXiv preprint arXiv:2208.12242* (2022).

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "ImageNet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[Ryb+20] Oleg Rybakov, Natasha Kononenko, Niranjan Subrahmanya, Mirkó Visontai, and Stella Laurenzo. "Streaming Keyword Spotting on Mobile Devices". In: *Proc. Interspeech 2020* (2020), pp. 2277–2281.

[Sai+15] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. "Learning the speech front-end with raw waveform cldnns". In: *Proc. Interspeech*. 2015.

[Sal+20] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. "Denoised smoothing: A provable defense for pretrained classifiers". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21945–21957.

[San+18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[SBD18] Robin Scheibler, Eric Bezzam, and Ivan Dokmanić. "Pyroomacoustics: A python package for audio room simulation and array processing algorithms". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 351–355.

[SBS19] Vinod Subramanian, Emmanouil Benetos, and Mark B. Sandler. "Robustness of Adversarial Attacks in Sound Event Classification". In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, NY, USA, Oct. 2019, pp. 239–243.

[Sch+18] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. "Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding". In: *arXiv preprint arXiv:1808.05665* (2018).

[Sch+19] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, 2019, pp. 3465–3469. DOI: 10.21437/Interspeech.2019-1873. URL: https://doi.org/10.21437/Interspeech.2019-1873.

[Sch+22] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. "Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *arXiv preprint arXiv:2210.08402* (2022).

[SE20] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution.* 2020. arXiv: 1907.05600 [cs.LG].

[Ser14] B Series. “Method for the subjective assessment of intermediate quality level of audio systems”. In: *International Telecommunication Union Radio-communication Assembly* (2014).

[Ser+16] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. “Very deep multilingual convolutional neural networks for LVCSR”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2016, pp. 4955–4959.

[SFF10] Richard Socher and Li Fei-Fei. “Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora”. In: *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2010, pp. 966–973.

[SG64] Abraham Savitzky and M. J. E. Golay. “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.” In: *Anal. Chem.* 36.8 (July 1964), pp. 1627–1639. DOI: 10.1021/ac60214a047. URL: http://dx.doi.org/10.1021/ac60214a047.

[Sha+18] Ankit Shah, Anurag Kumar, Alexander G Hauptmann, and Bhiksha Raj. “A closer look at weak label learning for audio events”. In: *arXiv preprint arXiv:1804.09288* (2018).

[Sha+21] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. “On the adversarial robustness of visual transformers”. In: *arXiv preprint arXiv:2103.15670* (2021).

[SHM22] Bowen Shi, Wei-Ning Hsu, and Abdelrahman Mohamed. “Robust Self-Supervised Audio-Visual Speech Recognition”. In: *CoRR* abs/2201.01763 (2022). arXiv: 2201.01763. URL: https://arxiv.org/abs/2201.01763.

[SJB14a] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. “A dataset and taxonomy for urban sound research”. In: *Proceedings of the 22nd ACM international conference on Multimedia.* ACM. 2014, pp. 1041–1044.

[SJB14b] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. “A dataset and taxonomy for urban sound research”. In: *Proc. of the ACM Int. Conf. on Multimedia and Expo (ICME).* 2014.

[SKP15a] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *CVPR.* 2015, pp. 815–823.

[SKP15b] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2015, pp. 815–823.

[SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[Smi+18] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. "Don't Decay the Learning Rate, Increase the Batch Size". In: *International Conference on Learning Representations*. 2018.

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958. URL: http://dl.acm.org/citation.cfm?id=2670313.

[Sri+21] Sangeeta Srivastava, Yun Wang, Andros Tjandra, Anurag Kumar, Chunxi Liu, Kritika Singh, and Yatharth Saraf. "Conformer-Based Self-Supervised Learning for Non-Speech Audio Tasks". In: *arXiv preprint arXiv:2110.07313* (2021).

[SRR20] Sanchari Sen, Balaraman Ravindran, and Anand Raghunathan. "Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks". In: *ICLR* (2020).

[ST04] Yoiti Suzuki and Hisashi Takeshima. "Equal-loudness-level contours for pure tones". In: *The Journal of the Acoustical Society of America* 116.2 (2004), pp. 918–933.

[ST19] Leslie N Smith and Nicholay Topin. "Super-convergence: Very fast training of neural networks using large learning rates". In: *Artificial intelligence and machine learning for multi-domain operations applications*. Vol. 11006. International Society for Optics and Photonics. 2019, p. 1100612.

[SZ14a] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv preprint* abs/1409.1556 (2014).

[SZ14b] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[SZ14c] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[SZ14d] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556.

[Sze+15] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[Sze+17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." In: *AAAI*. Vol. 4. 2017, p. 12.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[Tai+14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: Closing the gap to human-level performance in face verification". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1701–1708.

[TAL21] Shyam A Tailor, Milad Alizadeh, and Nicholas D Lane. *TorchQuant: A Hackable Quantization Library For Researchers, By Reseachers*. 2021.

[Tan+19] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. 2019. arXiv: 1807.11626 [cs.CV].

[Tao+23] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[TL19] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[TL21] Mingxing Tan and Quoc Le. "Efficientnetv2: Smaller models and faster training". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10096–10106.

[Tou+21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. "Training data-efficient image transformers & distillation through attention". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10347–10357.

[Tra+18] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. “A closer look at spatiotemporal convolutions for action recognition”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6450–6459.

[Tra+19] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. “Video classification with channel-separated convolutional networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5552–5561.

[Tra+23] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. “Effective data augmentation with diffusion models”. In: *arXiv preprint arXiv:2302.07944* (2023).

[Tse+18] Shao-Yen Tseng, Juncheng Li, Yun Wang, Florian Metze, Joseph Szurley, and Samarjit Das. “Multiple Instance Deep Learning for Weakly Supervised Small-Footprint Audio Event Detection”. In: *Proc. Interspeech 2018* (2018), pp. 3279–3283.

[Tsi+18] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. “Robustness may be at odds with accuracy”. In: *arXiv preprint arXiv:1805.12152* (2018).

[TTS16] Atousa Torabi, Niket Tandon, and Leonid Sigal. “Learning language-visual embedding for movie understanding with natural-language”. In: *arXiv preprint arXiv:1609.08124* (2016).

[Tüs+14] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. “Acoustic modeling with deep neural networks using raw time signal for LVCSR.” In: *INTERSPEECH*. 2014, pp. 890–894.

[UBG09] Nicolas Usunier, David Buffoni, and Patrick Gallinari. “Ranking with ordered weighted pairwise classification”. In: *International Conference on Machine Learning*. ACM. 2009, pp. 1057–1064.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 978-1-5108-6096-4. URL: http://dl.acm.org/citation.cfm?id=3295222.3295349.

[VBV21] Sergey Verbitskiy, Vladimir Berikov, and Viacheslav Vyshegorodtsev. “Eranns: Efficient residual audio neural networks for audio pattern recognition”. In: *arXiv preprint arXiv:2106.01621* (2021).

[Ven+15] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. "Sequence to sequence-video to text". In: *IEEE International Conference on Computer Vision*. IEEE. 2015, pp. 4534–4542.

[Ven+16] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. "Order-embeddings of images and language". In: *International Conference on Learning Representations*. 2016.

[Vin+08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by William W. Cohen, Andrew McCallum, and Sam T. Roweis. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[Vin+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11 (2010), pp. 3371–3408. URL: http://portal.acm.org/citation.cfm?id=1953039.

[VRG16] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. "Bidirectional joint representation learning with symmetrical deep neural networks for multimodal and crossmodal applications". In: *ACM International Conference on Multimedia Retrieval*. ACM. 2016, pp. 343–346.

[VRG17] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. "Generative adversarial networks for multimodal representation learning in video hyperlinking". In: *ACM International Conference on Multimedia Retrieval*. ACM. 2017, pp. 416–419.

[VRG18] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. "A Cross-modal Approach to Multimodal Fusion in Video Hyperlinking". In: *IEEE MultiMedia* 25.2 (2018), pp. 11–23.

[Wan+17] Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. "Adversarial cross-modal retrieval". In: *ACM Multimedia Conference*. ACM. 2017, pp. 154–162.

[Wan+18] Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. "Learning two-branch neural networks for image-text matching tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[Wan18] Yun Wang. “Polyphonic sound event detection with weak labeling”. In: *Thesis* (2018).

[Wan+21] Luyu Wang, Pauline Luc, Yan Wu, Adria Recasens, Lucas Smaira, Andrew Brock, Andrew Jaegle, Jean-Baptiste Alayrac, Sander Dieleman, Joao Carreira, and Aaron van den Oord. *Towards Learning Universal Audio Representations*. 2021. DOI: 10.48550/ARXIV.2111.12124. URL: https://arxiv.org/abs/2111.12124.

[War18] P. Warden. “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition”. In: *ArXiv e-prints* (Apr. 2018). arXiv: 1804.03209 [cs.CL]. URL: https://arxiv.org/abs/1804.03209.

[War18] Pete Warden. “Speech commands: A dataset for limited-vocabulary speech recognition”. In: *arXiv preprint arXiv:1804.03209* (2018).

[WC18] DeLiang Wang and Jitong Chen. “Supervised speech separation based on deep learning: An overview”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1702–1726.

[Wei+21] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan L. Yuille, and Christoph Feichtenhofer. “Masked Feature Prediction for Self-Supervised Visual Pre-Training”. In: *CoRR* abs/2112.09133 (2021). arXiv: 2112.09133. URL: https://arxiv.org/abs/2112.09133.

[WK18] Eric Wong and Zico Kolter. “Provable defenses against adversarial examples via the convex outer adversarial polytope”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5286–5295.

[WK20] Min Wu and Marta Kwiatkowska. “Robustness Guarantees for Deep Neural Networks on Videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[WLL16] Liwei Wang, Yin Li, and Svetlana Lazebnik. “Learning deep structure-preserving image-text embeddings”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2016, pp. 5005–5013.

[WLM18] Yun Wang, Juncheng Li, and Florian Metze. “Comparing the Max and Noisy-Or Pooling Functions in Multiple Instance Learning for Weakly Supervised Sequence Learning Tasks”. In: *Proc. Interspeech 2018* (2018), pp. 1339–1343.

[WLM19] Yun Wang, Juncheng Li, and Florian Metze. “A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 31–35.

[WML23] Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. "Sharpness Minimization Algorithms Do Not Only Minimize Sharpness To Achieve Better Generalization". In: *arXiv preprint arXiv:2307.11007* (2023).

[Woo04] Jim Woodhouse. "Plucked guitar transients: Comparison of measurements and synthesis". In: *Acta Acustica united with Acustica* 90.5 (2004), pp. 945–965.

[WRK20] Eric Wong, Leslie Rice, and J Zico Kolter. "Fast is better than free: Revisiting adversarial training". In: *arXiv preprint arXiv:2001.03994* (2020).

[WTF19] Weiyao Wang, Du Tran, and Matt Feiszli. "What Makes Training Multi-Modal Networks Hard?" In: *arXiv preprint arXiv:1905.12681* (2019).

[WTF20] Weiyao Wang, Du Tran, and Matt Feiszli. "What Makes Training Multi-Modal Classification Networks Hard?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 12695–12705.

[WTJ21] Ross Wightman, Hugo Touvron, and Hervé Jégou. "ResNet strikes back: An improved training procedure in timm". In: *arXiv preprint arXiv:2110.00476* (2021).

[Wu+18] Minhua Wu, Sankaran Panchapagesan, Ming Sun, Jiacheng Gu, Ryan Thomas, Shiv Naga Prasad Vitaladevuni, Bjorn Hoffmeister, and Arindam Mandal. "Monophone-based background modeling for two-stage on-device wake word detection". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2018, pp. 5494–5498.

[Wu*+23] Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. "Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP.* 2023.

[Xu+15] Ran Xu, Caiming Xiong, Wei Chen, and Jason J Corso. "Jointly Modeling Deep Video and Compositional Text to Bridge Vision and Language in a Unified Framework." In: *AAAI.* Vol. 5. 2015, p. 6.

[Xu+16] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. "Msr-vtt: A large video description dataset for bridging video and language". In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 5288–5296.

[Yan+21] Shu wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung yi Lee. "SUPERB: Speech Processing Universal PERformance Benchmark". In: *Proc. Interspeech 2021*. 2021, pp. 1194–1198. DOI: 10.21437/Interspeech.2021-1775.

[Yan+23] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. "Diffsound: Discrete diffusion model for text-to-sound generation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2023).

[Yao+20] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. "Pyhessian: Neural networks through the lens of the hessian". In: *2020 IEEE international conference on big data (Big data)*. IEEE. 2020, pp. 581–590.

[YLM22] Laura Yao, Juncheng Li, and Florian Metze. "CMU-VIDION: Modified BLIP with Audio for Video to Text Description". In: (2022).

[YM15] Fei Yan and Krystian Mikolajczyk. "Deep correlation for matching images and text". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2015, pp. 3441–3450.

[Yu+18] Changsong Yu, Karim Said Barsim, Qiuqiang Kong, and Bin Yang. "Multi-level Attention Model for Weakly Supervised Audio Classification". In: *ArXiv* abs/1803.02353 (2018).

[Yun+19] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 6022–6031. DOI: 10.1109/ICCV.2019.00612. URL: https://doi.org/10.1109/ICCV.2019.00612.

[YYH04] Rong Yan, Jun Yang, and Alexander G Hauptmann. "Learning query-class dependent weights in automatic video retrieval". In: *ACM Multimedia Conference*. ACM. 2004, pp. 548–555.

[ZA23] Lvmin Zhang and Maneesh Agrawala. "Adding conditional control to text-to-image diffusion models". In: *arXiv preprint arXiv:2302.05543* (2023).

[Zei+13] Matthew D Zeiler et al. "On rectified linear units for speech processing". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3517–3521.

[Zha+17a] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. "DolphinAttack: Inaudible voice commands". In: (2017), pp. 103–117.

[Zha+17b] Liang Zhang, Bingpeng Ma, Guorong Li, Qingming Huang, and Qi Tian. "Multi-Networks Joint Learning for Large-Scale Cross-Modal Retrieval". In: *ACM Multimedia Conference*. ACM. 2017, pp. 907–915.

[Zha+17c] Xishan Zhang, Ke Gao, Yongdong Zhang, Dongming Zhang, Jintao Li, and Qi Tian. "Task-Driven Dynamic Fusion: Reducing Ambiguity in Video Description". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2017, pp. 3713–3721.

[Zha+17d] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. "Loss functions for image restoration with neural networks". In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57.

[Zha+18] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[Zho+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Places: A 10 million image database for scene recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[Zho+22] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. "Understanding the robustness in vision transformers". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27378–27394.

[Zhu22] Ligeng Zhu. *THOP*. https://github.com/Lyken17/pytorch-OpCounter. 2022.