Knowledge-enhanced Representation Learning for Multiview Context Understanding

Jonathan Francis

CMU-LTI-22-005

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Eric Nyberg (co-Chair)
Jean Oh (co-Chair)
Louis-Philippe Morency
Alessandro Oltramari (Bosch Research)

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

Keywords: Multimodal Representation Learning, Embodied Artificial Intelligence, Robot Navigation, Autonomous Driving, Machine Learning, Domain Knowledge, Primitives, Priors, Constraints

To my family, friends, committee, and Bosch Research Pittsburgh colleagues.

Abstract

Computational context understanding refers to an agent's ability to fuse disparate sources of information for decision-making and is, therefore, generally regarded as a prerequisite for sophisticated machine reasoning capabilities, as in artificial intelligence (AI). *Data-driven* and *knowledge-driven* methods are two classical techniques in the pursuit of such machine sense-making capability. However, while data-driven methods seek to model the statistical regularities of events by making observations in the real-world, they remain difficult to interpret and they lack mechanisms for naturally incorporating external knowledge. Conversely, knowledge-driven methods combine structured knowledge bases, enable symbolic reasoning based on axiomatic principles, and yield more interpretable predictions; however, they often lack the ability to estimate the statistical salience of an inference or robustly accommodate perturbations in the input. To combat these issues, we use *hybrid* AI methodology as a general framework for combining the strengths of both approaches. Specifically, we inherit the concept of *neuro-symbolism* as a way of using domain knowledge to guide the learning progress of deep neural networks. Domain knowledge appears in many forms, including: (i) graphical models, which characterise such relationships between entities as dependence, independence, causality, correlation, and partial correlation; (ii) commonsense knowledge, which covers spatial knowledge, affordances from objects' physical properties, semantic relations, and functional knowledge; (iii) privileged information, in the form of demonstrations or soft labels from an expert agent; (iv) learned behaviour primitives and priors, which agents may compose for generalisable and transferable task-execution; and (v) auxiliary tasks, objectives, and constraints — carefully-chosen, for constrained optimisation.

Regardless of the type of domain knowledge available, the same practical objective remains: to learn meaningful neural representations, for downstream tasks of interest. The underlying goal of neural representation learning is to statistically identify the best explanatory factors of variation in the agent's input data or observations, often requiring intuition about the complementarity between multiple modalities or *views* in the input. While there has been much focus on learning effective neural representations for specific tasks, then transferring or adapting the learned representations to other tasks, comparatively less focus has been placed on representation learning in the presence of various types of domain knowledge. This knowledge could be used to recover information about the underlying generating process, to design effective modelling strategies in learning problems, to ensure model transferability or generalisability, or to understand the complementarity between views.

This thesis studies the avenues by which the aforementioned types of domain knowledge can be combined with neural representations, in order to achieve improved model performance and generalisability for the following problem domains: neural commonsense reasoning, multimodal robot navigation, and autonomous driving. This thesis contributes a collection of tools, methodologies, tasks, international AI challenges and leaderboards, datasets, and knowledge graphs; additionally, this work led to the successful organisation of two international workshops in safe learning for autonomous driving.

Contents

List of Figures

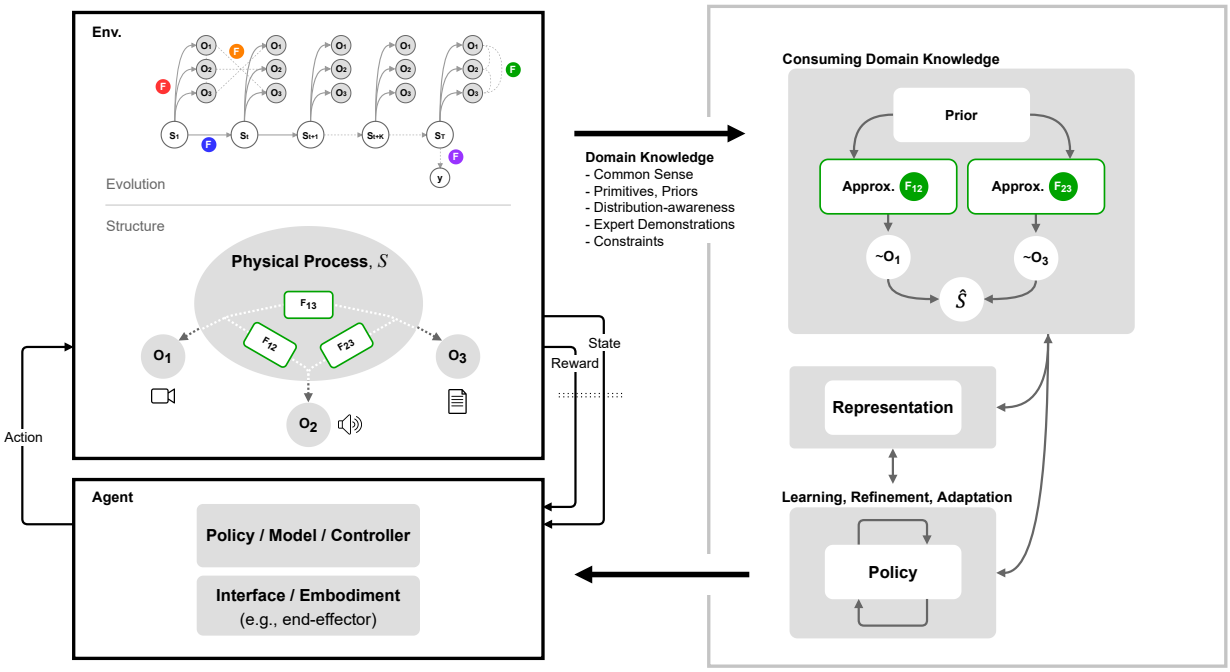
List of Tables

Chapter 1

Introduction

Cognitive agents rely on notions of *representationalism*, for characterising the structural properties of a given scene context and for engaging in decision-making, on the basis of that context understanding. Here, *representationalism* refers to the agent's ability to build a mental model, based on its sensory observations in the real-world and based on its background knowledge from past experience. This mental model is more than just the fusion of the agent's sensory inputs: it encompasses the agent's notions of beliefs (representing the knowledge of the world), desires (representing the states in the world that the agent wishes to assume; its goal(s)), and intentions (representing the present desire that the agent is committed to achieve; its 'active' sub-goal(s)) [99, 307]. More importantly, this mental model serves as a window through which sensory input is perceived, analysed, and transformed—in the present and future—predicated on background knowledge from the underlying domain [210]. Castelfranchi [42] and Wooldridge and Jennings [307] refined the concept of 'goal-direction', wherein agents are capable of reasoning, based on internal anticipatory and regulatory representations, about the consequences of their actions in an environment context. Lakoff et al. [156] further characterised how this notion of agency is predicated on embodiment, which, in turn, is essential to meaning and to agents' abilities to draw rational inferences.

Analogously, computational context understanding refers to a machine's ability to engage in decision-making, through the combination of background knowledge with the fusion of disparate sources of information, and is generally regarded as a prerequisite for sophisticated reasoning, as in artificial intelligence (AI). *Connectionism* (goal orientation, based on data and observations) and *symbolism* (goal orientation, based on background knowledge) are two classical perspectives in the pursuit of such machine sense-making capability. However, while data-driven methods seek to model the statistical regularities of events, by making observations in the real-world, they remain difficult to interpret and they lack mechanisms for naturally incorporating external information which could influence how those observations are interpreted or represented. Conversely, knowledge-driven methods may utilise physical models of the world, combine structured knowledge bases, perform symbolic reasoning based on axiomatic principles, and are often more interpretable in their downstream predictions; however, they often lack the ability to estimate the statistical salience of an inference or robustly accommodate perturbations in the input. To combat these issues, we study *hybrid* AI methodology as a general framework for combining the strengths of both approaches. Specifically, we inherit the concept of neuro-symbolism, as a way of using *domain knowledge* to guide the learning process of deep neural networks. Indeed, systems that unify observations with domain knowledge lend themselves well to the encoding of human intention [193]. Domain knowledge appears in many forms, including: (i) graphical models, which characterise such relationships between



(a) Interaction Paradigm

(b) Reasoning Paradigm

Figure 1.1: Overview of a hybrid model, where an agent is subject to domain knowledge-inspired learning. We discuss a collection of problems, wherein domain knowledge is utilised for learning better representations and improving model generalisation. (a, top) To characterise the evolution of this physical process, we can think about a set of *governing functions*, where the function in **Red** characterises various observation models that, in effect, generate the observations $\{O_1, O_2, O_3\}$. **Blue** characterises the transition dynamics in the environment, **Orange** controls the temporal interdependencies between views, **Purple** characterises the data-labelling function (which we can think of as the *teacher*, for an arbitrary learner that attempts to perform a mapping from observations to these teacher labels), and **Green** characterises the complementarity between views. Not all connections in the state evolution are drawn. View in colour.

entities as dependence, independence, causality, correlation, and partial correlation; (ii) commonsense knowledge, which covers spatial knowledge, affordances from physical object properties, semantic relations, and functional knowledge; (iii) privileged information, in the form of expert demonstrations or soft labels from an expert agent; (iv) auxiliary tasks or objectives, carefully-chosen, for constrained optimisation; and (v) structured behaviour primitives, which may be composed for generalisable and transferable task-execution.

Regardless of the type of domain knowledge available, the same practical objective remains: to learn meaningful neural representations, for downstream tasks of interest. The underlying goal of neural representation learning is to statistically identify the best explanatory factors of variation in the agent's input data or observations, often requiring intuition about the complementarity between multiple modalities or *views* in the input [16, 23, 176, 297]. By understanding these factors of variation, we can glean insight about (i) the distribution of the data-generating process that underlies both the agent's observations and the dynamics of the environment; and about (ii) aspects that may confound an agent's learning paradigm, such as noise and distribution shifts. Pursuit of this objective usually involves learning statistical mappings from the input space to a latent vector space, in order to facilitate the emergence of desired intrinsic properties (e.g., concentration of probability mass in lower-dimensional manifolds, inter-mode density, intra-mode sparsity) and extrinsic properties (e.g., improved downstream task performance) from the use of the neural representation. Indeed, much focus has been placed on learning effective neural representations for a specific task, then transferring or adapting the learned representations to different, related tasks. However, comparatively less focus has been placed on representation learning in the presence of various types of domain knowledge, where this knowledge can be used to recover information about the underlying generating process, to design effective modelling strategies in learning problems, to ensure model transferability or generalisability, or to understand the complementarity between views (Fig. 1.1)

1.1 Thesis Overview

Thesis statement: Learning representations in the absence of domain knowledge is insufficient for true model generalisability to unseen environments; various forms of domain knowledge can be systematically leveraged, across various AI tasks, in order to improve cross-domain model performance. Thus, it is not appropriate to study domain knowledge in a single, siloed domain.

This thesis studies the avenues by which domain knowledge can be combined with neural representations, across diverse problem domains, to achieve, e.g., improved generalisability and sample-efficiency.

Specifically, we address the following research challenges:

1. Injecting domain knowledge in neural systems, studied across a large cross-section of research areas, applications, and problem domains.

2. Developing extensible neural architectures that flexibly incorporate domain knowledge into their perceptual pipelines and/or training objectives. As the primary theme of this thesis, we provide specification and analysis of hybrid modelling strategies, throughout, such as: (i) commonsense knowledge grounding (for extraction) and attention based combination with neural context (injection) for question answering (Section 4.2); (ii) leveraging domain knowledge for modularity/hierarchicality, in multimodal perception pipelines, for robot navigation (Section 5.2) and autonomous driving (Section 6.3); and (iii) combining neural models with classical control (Sections 6.3, 7.2.3, and 7.2).

3. Leveraging knowledge as a statistical prior, with careful selection of modelling strategy (e.g., variational inference, normalising flow) and objectives, in the context of learning embedded skills for robot navigation (Section 5.3), multi-agent trajectory prediction (Section 6.2) and goal prediction (Section 6.3) in autonomous driving.

4. Grounding neural predictions on physics-based models to ensure conformance to, e.g., vehicle dynamics (Sections 6.2, 6.3, 7.2.3, 7.2).

5. Understanding the alignment between the downstream task and the types of domain knowledge that would facilitate improved optimisation and performance. Without this analysis, even the introduction of additional information can have a negative impact of model performance, as models may learn to ignore the higher-frequency variation in the knowledge-based context, in favour of easier (but task-specific) training signals in the samples. We study these concepts, in the context of noise-reduction in commonsense question answering (Sections 4.2, 4.3) and learning embedded skills in robot navigation (Section 5.3).

6. Performing knowledge transfer to unseen domains, e.g., in the context of zero-shot evaluation for commonsense question answering on unseen datasets (Section 4.3) and transferable robot skills (Section 5.3).

This thesis is organised as follows:

Chapter 2 presents preliminaries, to introduce topics covered in the thesis. This chapter starts with a discussion of various domain knowledge types, then provides an overview of multiview representation learning—with specific treatment of visual, textual, and knowledge representations, and learning to align representations from multiple views.

Chapter 3 describes a collection of application areas, that were chosen to illustrate both the challenges and opportunities of domain knowledge-enhancement for neural functional approximation. For neuro-symbolic commonsense reasoning, we consider commonsense question answering and constrained text generation. For embodied multimodal robot navigation, we consider waypoint prediction and learning transferable navigation primitives. Finally, in the context of autonomous driving, we consider multi-agent trajectory forecasting and learning to drive settings.

Chapter 4 highlights commonsense knowledge as a dominant form of domain knowledge for guiding the training of neural language models (LMs), in question answering and constrained text generation tasks. We consider challenges in understanding knowledge/task alignment for improved downstream task performance, developing neuro-symbolic architectures for commonsense reasoning, knowledge elicitation from symbolic resources through extraction and injection mechanisms, and understanding knowledge-based pre-training tasks for improved downstream performance through synthetic QA set generation.

Chapter 5 covers the use of priors, primitives, and skills, in the context of robot learning for navigation tasks. In the first section, we introduce the use of knowledge-driven scene priors for semantic audio-visual embodied navigation: we combine semantic information from our novel knowledge graph that encodes object-region relations, spatial knowledge from dual Graph Convolutional Networks, and background knowledge from a series of pre-training tasks—all within a reinforcement learning framework for audio-visual navigation. We define a new audio-visual navigation sub-task, where agents are evaluated on novel sounding objects, as opposed to unheard clips of known objects; and we show improvements over strong baselines in generalisation to unseen regions and novel sounding objects, within the Habitat-Matterport3D simulation environment, under the SoundSpaces task. In the second section, we propose

a framework for distilling a navigation agent's experience into a representation of reusable maneuvers, where the high-frequency and label-independent variation in the instructions and visual context are removed. Our approach encourages the agent to generalise reusable navigation maneuvers on the basis of similarities across high-level textual instructions, conditioned on past actions, in order to leverage experience in executing familiar sub-commands from new instructions. We achieve this association through a method called auxiliary variable variational approximation, which allows us to introduce a latent variable for estimating the conditional posterior distribution over all observations and executed trajectories (otherwise intractable for most distributions of interest). The agent then learns how to compose samples from this skill space and couple them with conventional multimodal co-grounding mechanisms, through policy refinement, for improved generalisability and downstream performance.

Chapter 6 covers the use of statistical priors, in the context of autonomous driving tasks, with a focus on how knowledge of the form of a distribution (which underlies trajectories or waypoint goals) leads to multiple benefits, including: (i) model generalisability, beyond simple interpolation between point-wise samples in a dataset; (ii) the ability to reveal agent intentionality for more interpretable predictions; and (iii) the ability to infer implicit rules about admissible behaviour in the environment. In the first section, we propose a model that addresses generalisation challenges in multi-agent trajectory forecasting tasks, through the introduction of an informative (and annotation-free) prior and rich multimodal encodings of agent-to-agent and scene-to-agent information. We offer new metrics to evaluate the diversity of trajectory predictions, while ensuring admissibility of each trajectory. Based on our new metrics as well as those used in prior work, we compare our model with strong baselines and ablations across two datasets and show a 35% performance-improvement over the state-of-the-art. In the second section, we learn the prior jointly with the task of simulated urban driving, where we introduce a distribution-aware trajectory generation mechanism that remains conformant to both road geometry and vehicle kinematics. We show how our agent uses this learned prior to generalise to completely out-of-distribution driving scenarios, such as busy towns, abnormal turns, unseen traffic patterns such as roundabouts, etc.

Chapter 7 studies the use of constraint functions as domain knowledge in autonomous driving settings. We observe that technologies which are to be applied to safety-critical applications must adhere to safety constraints, throughout their interactions with their environments, as any safety infraction in urban/highway driving or high-speed racing, could lead to catastrophic failures. Moreover, their training environments must capture sufficient realism (e.g., in visual rendering, vehicular dynamics, and task objectives), in order for the methods to be eligible for simulation-to-real transfer. The chapter starts by addressing the lack of realistic simulation for producing safety-aware methods: we introduce the `Learn-to-Race` (`L2R`) framework as a particularly challenging proving ground for safe learning algorithms. `L2R` is OpenAI-gym compliant training environment for simulated *Formula*-style racing, which enables agents to learn to race on high-precision models of real-world tracks (e.g., the famed Thruxton Circuit and the Las Vegas Motor Speedway) and to use a suite of rich multimodal sensory information, predicated on accurate vehicle dynamics. We additionally define the L2R AI task, by introducing two (2) objectives and seven (7) metrics that characterise and measure performance, safety, and desirable agent behaviour. We additionally provide an official `L2R` task dataset of expert demonstrations and a series of baseline experiments and reference implementations. In this task, algorithms must learn to control vehicles at their physical limits, with minimal margins for safety, while making sub-second decisions in a fast-changing environment and while remaining robust to distribution shifts and novel road features. Next, to address the challenges in safe policy optimisation, we propose the use of safety constraints for autonomous racing, inspired by the theoretical foundations of Hamilton-Jacobi (HJ) reachability analysis in optimal control. Here, we define a safety controller that intervenes whenever an agent approaches bad

states, and we show that even an agent that generates actions at random is guaranteed to stay on the drivable area; we further show that an arbitrary learning policy that is coupled with this safe controller is able to learn performant driving behaviour, both safely and sample-efficiently. Finally, we demonstrate that the HJ safety value can be learned and updated directly from vision context, thereby expanding HJ reachability to applications where high-fidelity dynamics models may not be available. While not necessary for convergence, we warm-start the safety value function using values pre-computed with a nominal model. Next, the value function is updated directly on transitions of ego-agent's frontal camera view and vehicle speed. We report state-of-the-art results on the `L2R` benchmark task, and we show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

Chapter 2

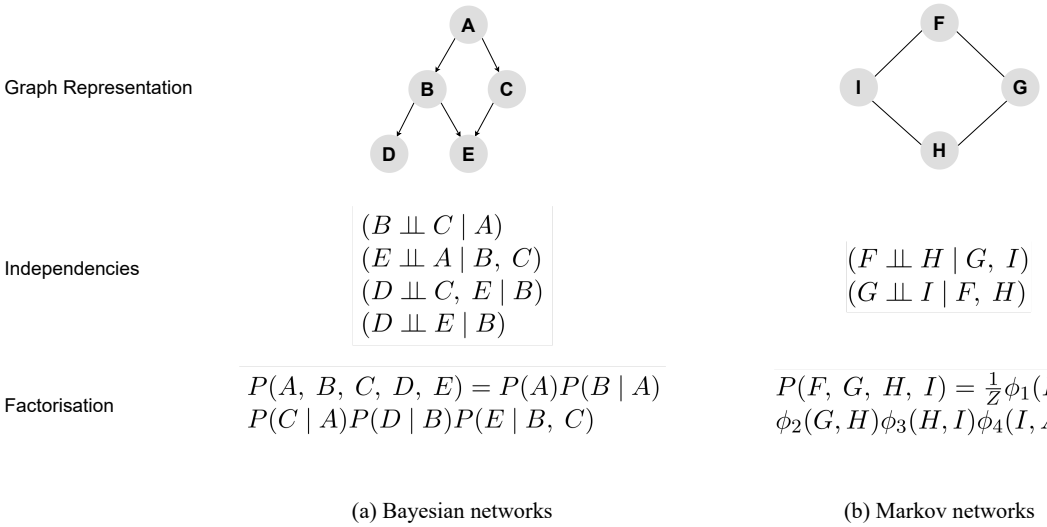
Preliminaries

In this chapter, we introduce the requisite concepts for characterising various types of domain knowledge and multiview representation learning. We begin by discussing various forms of domain knowledge that are common to problem areas of interest; in this thesis, we highlight: symbolic commonsense knowledge, (probabilistic) graphical models, learning with constraints, and knowledge distillation. Next, we discuss the two stages of multiview representation learning: (i) characterising, representing, and encoding multiple modalities, such as vision, natural language, and commonsense knowledge; and (ii) learning to align representations from multiple views.

2.1 Domain Knowledge

We refer to domain knowledge as, simply, the elevated understanding about the environment in which an agent operates that would aid in its learning and/or goal-direction [119]. This opportunity for utilising domain knowledge in learning has been studied in many areas, including: Library and Information Sciences [119], Information Retrieval [66, 67, 80, 81], Human Cognitive Developmental Psychology and Perception [82, 86, 304], and Pedagogy [5, 295]. In the context of Robotics and Artificial Intelligence: Romea et al. [243] propose a method for encoding domain knowledge as constraints, for lifelong concept learning and object discovery; Candido et al. [39] exploit domain knowledge in the specification of primitives, for more efficient planning and control; and Tokmakov et al. [285] train object-tracking models, by replicating human perceptual notions of object permanence and grounding. However, these works have neglected to establish a taxonomy of domain knowledge or to characterise how domain knowledge can be concretely operationalised in a diverse set of fields.

Following these investigations, we can identify several forms in which domain knowledge manifests for constructive use in learning problems, e.g.: commonsense knowledge, probabilistic graphical models, generalized distillation (including pre-training and multi-objective learning), and reasoning primitives (e.g., skills and modular architectural components), and constraint functions. In this thesis, we study how these instances of domain knowledge can be used to guide the learning process of artificial agents, in a variety of downstream tasks and applications. In the remainder of this section, we describe some relevant concepts that will aid in our characterisation of knowledge-enhanced learning.



(a) Bayesian networks

(b) Markov networks

Figure 2.1: Depiction of two types of graphical models.

2.1.1 Common Sense

Commonsense knowledge encompasses practical knowledge about the world, which cognitive agents (e.g., humans) are expected to have and leverage for reasoning. An agent is said to have common sense, as McCarthy et al. [186] argues in his seminal work, "... if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows." In the context of machine commonsense reasoning, various resources exist for specifying commonsense knowledge in the form of symbolic representations (e.g., knowledge graphs) that aid in an agent's execution of a downstream task (e.g., question answering, robot navigation). Various categories of commonsense knowledge have been discussed at length by philosophers, computational linguists, and cognitive psychologists[1], where we can identify: *declarative commonsense*, whose scope encompasses factual knowledge, e.g., 'the sky is blue', 'Paris is in France'; *taxonomic knowledge*, e.g., 'football players are athletes', 'cats are mammals'; *relational knowledge*, e.g., 'the nose is part of the skull', 'handwriting requires a hand and a writing instrument'; *procedural commonsense*, which includes prescriptive knowledge, e.g., 'one needs an oven before baking cakes', 'the electricity should be off while the switch is being repaired' [120]; *sentiment knowledge*, e.g., 'rushing to the hospital makes people worried', 'being in vacation makes people relaxed'; and *metaphorical knowledge* (e.g., 'time flies', 'raining cats and dogs'). Because these different commonsense knowledge *types* (e.g., procedural commonsense knowledge, relational commonsense knowledge) may support corresponding unique modes of reasoning (e.g., cause-effect resolution and story understanding, spatial navigation and co-reference), we can refer to these types as specific dimensions or *domains* of commonsense knowledge. In Chapter 4, we confirm that, indeed, different reasoning modes necessitate the corresponding type of commonsense knowledge and combining inappropriate commonsense domain knowledge with reasoning pipelines is detrimental to downstream performance.

Examples of symbolic representation of commonsense knowledge, in declarative form, can be taken from `ConceptNet` [264]: a commonsense knowledge graph, which contains over 21 million edges and 8 million nodes (1.5 million of which are in the partition for the English vocabulary). Any given pair of

[1]Interested readers are referred to [70].

nodes in the knowledge graph are connected by an edge and can be concisely specified as a triple:

$$(C1, r, C2),$$

where the natural language head and tail concepts, $C1$ and $C2$, are associated by commonsense relation r, e.g., as in *(dinner, AtLocation, restaurant)*. These declarative constructs are convenient, because they have straightforward logical consequences and because they can be easily lexicalised into natural language statements. Thanks to its coverage, `ConceptNet` is one of the most popular semantic networks for common sense. `ATOMIC` [250] is another knowledge-base that focuses on procedural knowledge. Triples are of the form *(Event, r,* $\{$*Effect* | *Persona* | *Mental-state*$\}$), where head and tail elements are short sentences or verb phrases and r represents an *if-then* relation type, e.g.:

(X compliments Y, xIntent, X wants to be nice)

In Section 2.2.1, we describe how commonsense knowledge can be combined with neural representations. In Chapter 4, we show that, by identifying the most appropriate type of commonsense knowledge for a particular problem (e.g., from declarative, taxonomic, relational, procedural, sentiment, and metaphorical common sense), we can consistently improve downstream performance and cross-domain generalisation of neural language models, on multiple-choice commonsense question-answering tasks in natural language processing. We introduce a new model that performs commonsense knowledge grounding and lexicalisation (for extraction) and trilinear attention-based combination with neural context (injection) in Section 4.2, and we propose a novel neuro-symbolic framework for zero-shot question answering across commonsense task in Section 4.3. Additionally, we introduce the use of knowledge-driven scene priors for semantic audio-visual embodied navigation, in Section 5.2, where we combine semantic information from our novel knowledge graph that encodes object-region relations, spatial knowledge from dual Graph Convolutional Networks, and background knowledge from a series of pre-training tasks—all within a reinforcement learning framework for audio-visual navigation.

2.1.2 Probabilistic Graphical Models

Probabilistic graphical models [145] combine domain knowledge and statistical experience, through declarative representation of directed (Bayesian networks) and undirected (Markov networks) graphical structures. These graphical structures model joint probability distributions over sets of random variables and allow for flexible factorisation (Figure 2.1), enabling us to understand various relations between random variables (correlations, independencies, dependencies, causalities, and partial correlations) and thereby allow us to encode expert knowledge from the underlying problem domain. In Section 6.3, we discuss a possible factorisation of the distribution over control actions as motivation for defining modular architectures in visuomotor control tasks for autonomous driving.

Indeed, these probabilistic graphical models allow us to reason about the probability distributions that underlie the input and output quantities of, e.g., generative models, as in: Generative Adversarial Networks, variational Bayesian methodology, and Normalizing Flow. Central to the focus areas in this thesis, the following subsections provide examples of the latter two algorithmic classes and how domain knowledge can be introduced therein.

Variational Approximation

Variational Bayesian methods became popular for their ability to enable efficient approximate inference and learning over probabilistic models, whose latent random variables have intractable posterior distributions [137]. Rather than directly attempting to answer queries of some intractable posterior P, variational

inference allows us to represent the quantity of interest as the result of an optimisation problem, wherein the desired solution is approximated. First, P is projected to a tractable family of distributions Q, inference is performed on the projected Q, then the distributions are aligned through posterior regularisation.

More formally, we consider a joint probability distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, which characterises a probabilistic model $p_\theta(\mathbf{x}|\mathbf{z})$ and prior $p(\mathbf{z})$. As a proxy for the intractable posterior, we introduce the variational approximate distribution $q_\phi(\mathbf{z}|\mathbf{x})$, with parameters ϕ, and use it in pursuit of a bound on the marginal likelihood of the observations (as in [132]), referred to as the evidence lower bound (ELBO):

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$\geq \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \mathbb{E}_q(\log p_\theta(\mathbf{x}|\mathbf{z})),$$

In order to use this bound to optimise $\{\theta, \phi\}$ over large datasets, we compute the gradient of the expected log-likelihood $\nabla_\phi \mathbb{E}_{q_\phi(z)}(\log\, p_\theta(\mathbf{x}|\mathbf{z}))$ through a Monte Carlo gradient estimate, with reparameterisation, assuming $q_\phi(z)$ to be of Gaussian form, $\mathcal{N}(z|\mu, \sigma^2)$ and $\phi = \{\mu, \sigma^2\}$, such that:

$$z \sim \mathcal{N}(z|\mu, \sigma^2) \Leftrightarrow z = \mu + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0, 1)$$

This enables us to differentiate, with respect to ϕ, the parameters of the variational distribution, using a Monte Carlo approximation, with draws from the *base distribution*:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}(f_\theta(z)) \Leftrightarrow \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}(\nabla_\phi f_\theta(\mu + \sigma\epsilon))$$

Finally, we often follow common practice [101, 140, 235, 236, 268] in using a posterior projection model—an inference network—which allows us to compute a set of global variational parameters that may be used for both train- and test-time inference.

The selection of q_x provides an opportunity for injecting domain knowledge into the variational inference framework, where the prior proposal q_x can be produced through pre-training. In Chapter 5, we pre-train the embedding and inference networks on tasks that are close to their intended specialisation in the downstream process. We further utilise this variation formulation for generating a representation of learnable primitives that guide a navigation policy's downstream task-execution.

Normalizing Flow

For many problems of interest, as in trajectory forecasting and navigation, a limitation of the variational methodology is that the available families for the choice of approximating distributions still lack the expressiveness for capturing diverse modes in the data distribution [217, 235]. Moreover, variational autoencoders (VAEs) [137] are only *approximately* able infer the values of the latent variables that correspond to a data-point [139]. Avoiding the prospect of having to directly specify more complex distributions

that will fit the data well (intractable for most distributions of interest) and allowing for direct evaluation of the latent density, Normalizing Flow provides a general framework for transforming a simple probability density (*base distribution*) into a more expressive one, through a series of invertible mappings [139, 217, 219, 235, 274].

Formally, let f be an invertible and smooth function, with $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\mathbf{x} = f(\mathbf{z})$, $\mathbf{z} \sim p_{\mathbf{z}}$, $f^{-1} = g$, and thus $g \circ f(\mathbf{z}) = \mathbf{z}$, for d-dimensional random vectors $\mathbf{x}$ and $\mathbf{z}$. Further, we attribute to f the property of *diffeomorphism* [192], which ensures that $q_{\mathbf{x}}$ remains well-defined and obtainable through a change of variables, and ensures that $p_{\mathbf{z}}$ is uniformly distributed on the same domain as the data space [167]—insofar as both f and its inverse f^{-1} are differentiable and $\mathbf{z}$ retains the same dimension as $\mathbf{x}$:

$$q_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z}) \left|\det \frac{\partial f}{\partial z}\right|^{-1} = p_{\mathbf{z}}(f^{-1}(\mathbf{x})) \left|\det \frac{\partial f^{-1}}{\partial \mathbf{x}}\right|$$

We can construct arbitrarily complex densities, by *flowing* $\mathbf{z}$ along the path created by a chain of K successive *normalizing* distributions $p_{\mathbf{z}}(\mathbf{z})$, with each successive distribution governed by a diffeomorphic transformation:

$$\mathbf{x} = \mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z_0})$$

Following this sequence of transformations, our main interfaces with the flow-based model are through either sampling or evaluating its density, where, in the former, we sample from $p_{\mathbf{z}}(\mathbf{z})$ and must compute the forward transformation f; in the latter, we must compute the inverse transformation f^{-1}, its Jacobian determinant, and the $p_{\mathbf{z}}(\mathbf{z})$ density evaluation. In practice, f and f^{-1} are implemented as neural networks and the *base distribution* $p_{\mathbf{z}}$ is typically taken to be a simple one, e.g., a multivariate normal or a uniform distribution over the admissible regions of the state-space.

This selection of $p_{\mathbf{z}}$ actually provides the first opportunity for injecting domain knowledge in flow-based models—where $p_{\mathbf{z}}$ can be chosen/adjusted to be more informative for the downstream task.[2] As we will see in Section 6.2.2, we select an annotation-free prior distribution over the drivable area, in a multi-agent vehicle trajectory prediction problem, which proves crucial to achieving diverse and admissible trajectory predictions. The second opportunity for injecting domain knowledge follows from conditional flow variants that incorporate some side information. Various implementations in the literature feature side information in only simplistic forms (e.g., one-hot vector representations). In Section 6.3.4, we learn a posterior that conditions this prior on more complex information from the scene context, in learning-to-drive settings, making the resultant model more generalisable and sensitive to dynamic obstacles.

2.1.3 Constraints

Training an agent exclusively with data is neither sample-efficient, nor interpretable. Systems that unify concrete examples with logical rules and constraints, instead, lend themselves well to the encoding of human intention and domain knowledge [193].

Consider a statistical model, $p_\theta(\mathbf{x}|\mathcal{C})$, where $\mathbf{x} \sim p_\theta(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ is a generated random variable of dimension d (in the case of a generative model), or a label (in the case of a discriminative model), with $\mathcal{C}$

[2]For example, a distributional prior that is learned through pre-training on a task that may be felt to be similar or relevant to the downstream task of interest.

as some input scene context. Next, suppose we have some parameterised constraint function $f_\phi(\mathbf{x}) \in \mathbb{R}$ which encodes some value on the model's predictions, where higher values of f denote a better $\mathbf{x}$, with respect to some domain knowledge.

Natural language processing example. Taking an example [123] from sentiment classification in natural language processing, suppose a model must classify the sentiment (positive or negative) in the following statement S: *This was a terrific movie, but the director could have done better.* A challenge for language models is that of characterising the contrastive nature of such sequences (given by the conjunction 'but'), in order to infer the dominant sentiment. Whereas the correct sentiment label of 'negative' may well be available for this particular example in a dataset, a trained model would not be immediately informed from the label, alone, about 'why' this example contains negative sentiment. For this reason, we may wish to encode a constraint, such as `[Sentence with structure A-but-B` $\rightarrow$ `sentiment of B dominates]`, as the following *soft* first-order logic [14] rule:

$$f := \textit{has-'A-but-B'-structure(S)} \implies \{\mathbf{1}(y = +) \implies \sigma_\theta(B)_+ \;\&\; \sigma_\theta(B)_+ \implies \mathbf{1}(y = +)\},$$

where $\mathbf{1}(\cdot)$ is an indicator function that takes 1 when its argument is true, and 0 otherwise; $\sigma_\theta(\cdot) \in [0, 1]$ are soft model predictions; class '+' represents 'positive'; and $\sigma_\theta(B)_+$ is the element of $\sigma_\theta(B)$ for class '+'. When the label is positive (i.e., $y = +$), the rule takes the value $\sigma_\theta(B)_+$, else $1 - \sigma_\theta(B)_+$.

One way to impose constraint information on the model is to maximise the expectation over the model's input distribution, with respect to our constraint function, or:

$$\min_\theta \mathcal{L}(\theta) - \alpha\, \mathbb{E}_{p_\theta}(f_\phi(\mathbf{x})),$$

where $\mathcal{L}(\theta)$ is the model's optimisation objective (e.g., cross-entropy) and the expectation term performs posterior regularisation [97] thereupon, by imposing the domain knowledge constraint. As a consequence of this optimisation, we would then take the gradient against this expectation, which is generally more effective than other forms of regularisation, such as sparsity-promoting penalties [97, 124, 131, 244]. Unfortunately, however, the derivative of this term can exhibit high variance, through the commonly-used log-derivative trick [314], if the density of the generative distribution p_θ is not one that can be efficiently reparameterised (e.g., Gaussian family [137]). An alternative is to impose the constraint on some variational distribution q:

$$\min_\theta \mathcal{L}(\theta) - \alpha[\mathrm{KL}(q(\mathbf{x})||p_\theta(\mathbf{x})) - \lambda\, \mathbb{E}_{q_\theta}(f_\phi(\mathbf{x}))]$$

Finally, this aggregate objective is optimised using an EM-style methodology [97], where, in the E-step, we optimise the constraint-based component with respect to our variational distribution:

$$q^*(\mathbf{x}) = p_\theta(\mathbf{x}) \exp\{\alpha f(\mathbf{x})\}/Z,$$

for normalisation term Z. Fixing q from the E-step, we minimise with respect to the model's parameters θ for the M-step:

$$\min_\theta \mathcal{L}(\theta) - \mathbb{E}_{q^*}(\log p_\theta(\mathbf{x}))$$

Robotics example. We inherit another example [51] from the application of reinforcement learning to autonomous systems, where the goal is to make learning-based robotics and autonomous driving agents safer and more sample-efficient. We can define this notion of 'safety' by way of a learnable objective added to the standard reward-centric objective in reinforcement learning: the agent not only must maximise expected reward, but must also constrain the cost (on expectation) incurred by engaging in unsafe behaviour, such as crashing into obstacles, driving off the road, etc.

This problem of combing safety constraints with reinforcement learning (RL) agents is often formulated as a constrained Markov decision process (CMDP), i.e., on top of an MDP $(\mathcal{X}, \mathcal{U}, R, \mathcal{F})$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\mathcal{F} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ characterises the system dynamics, and $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function. The CMDP includes an additional set of cost functions, $\{C_1, \ldots, C_m\}$, where each $C_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ maps state-action transitions to costs characterising constraint violations.

The objective of RL is to find a policy $\pi : \mathcal{X} \to \mathcal{U}$ that maximises the expected cumulative rewards, $V_R^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k R(x_k, u_k) | x_0 = x\right]$, where $\gamma \in [0, 1)$ is a temporal discount factor. Similarly, the expected cumulative costs are defined as $V_{C_i}^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k C_i(x_k, u_k) | x_0 = x\right]$. Notice that the CMDP requires the policy to be feasible, by imposing limits d_i for the costs, i.e., $V_{C_i}(\pi) \leq d_i, \forall i$. Putting everything together, the RL problem in a CMDP is:

$$\pi^* = \arg\max_{\pi} \; V_R^\pi(x) \quad \text{s.t.} \quad V_{C_i}^\pi(x) \leq d_i \;\; \forall i. \tag{2.1}$$

Solving a CMDP problem is challenging, because the policy needs to be optimised over the set of feasible states; this requires off-policy evaluation of the constraint functions, to determine whether a policy is feasible [2]. As a result, safety grows with experience, but requires diverse state-action pairs, including unsafe ones [265]. Various approaches in literature attempt to guarantee the safety of general, continuous non-linear systems. These can be combined with MDP formulation, methods typically rely on knowledge of the environment dynamics. Control barrier functions (CBFs) provide a measure of safety with gradients that inform the acceptable safe actions [7]. For specific forms of dynamics, e.g., control-affine [58], and unlimited actuation bounds, this approach can be scalable to higher-dimensional systems and can be paired with an efficient online quadratic program for computing the instantaneous control [58]. Unfortunately, finding a valid control barrier function for a general system is a nontrivial task. Lyapunov-based methods [61, 62] suffer from the same limitation of requiring hand-crafted functions.

Hamilton-Jacobi reachability is a technique that uses continuous-time dynamic programming to directly compute a value function that captures the optimal safe control for a general nonlinear system [19, 90]. This method can provide hard safety guarantees for systems, subject to bounded uncertainties and disturbances. To generate the safety constraint, one can apply HJ reachability to a general nonlinear system model, denoted as $\dot{x} = f(x, u)$. Here x is the state, u is the control contained within a compact set $\mathcal{U}$. The dynamics are assumed bounded and Lipschitz continuous. For discrete-time approximations the time step $\Delta t > 0$ is used. Within this framework, RL agents can learn complex control policies safely. We hypothesise that it is more effective to use a low-fidelity model to delineate safe and unsafe states, compared to using the same model for RL or even model-predictive control (MPC).

We denote all allowable states as $\mathcal{K}$, for which there exists a terminal reward $l(x)$, such that $x \in \mathcal{K} \iff l(x) \geq 0$. An $l(x)$ that satisfy this condition is the signed distance to the boundary of $\mathcal{K}$. Taking autonomous driving as an example, $\mathcal{K}$ is the drivable area and $l(x)$ is the shortest distance to road boundary or obstacle. This set $\mathcal{K}$ is the complement of the failure set that must be avoided. The goal of this HJ reachability problem is to compute a safety value function that maps a state to its safety value with respect to $l(x)$ over time. This is done by capturing the minimum reward achieved over time by the system

applying an optimal control policy:

$$V_S(x,T) = \sup_{u(\cdot)} \min_{t\in[0,T]} l(\xi^{u,d}_{x,T}(t)), \tag{2.2}$$

where ξ is the state trajectory, $T < 0$ is the initial time, and 0 is the final time. To solve for this safety value function, a form of continuous dynamic programming is applied backwards in time from $t = 0$ to $t = T$ using the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI):

$$\min\left\{\frac{\partial V_S}{\partial t} + \max_{u\in\mathcal{U}}\langle f(x,u), \nabla V_S(x)\rangle, l(x) - V_S(x,t)\right\} = 0, \quad V_S(x,0) = l(x). \tag{2.3}$$

The super-zero level set of this function is called the reachable tube, and describes all states from which the system can remain outside of the failure set for the time horizon. For the infinite-time, if the limit exists, we define the converged value function as $V_S(x) = \lim_{T\to-\infty} V_S(x,T)$. Once the safety value function is computed, the optimal safe control can be found online by solving the Hamiltonian: $\pi^*_S(x) = \arg\max_{u\in\mathcal{U}}\langle f(x,u), \nabla V_S(x)\rangle$. This safe control is typically applied in a least-restrictive way wherein the safety controller becomes active only when the system approaches the boundary of the reachable tube, i.e., $u \sim \pi$ if $V_S(x,T) \geq 0$ and π^*_S otherwise.

There are two major drawbacks to HJ reachability. The first is that the technique suffers from the curse of dimensionality and scales exponentially with number of states in the system. Because of this, the technique can only be used directly on systems of up to 4-5 dimensions [19]. When using specific dynamics formulations and/or restricted controllers, this upper limit can be extended [57, 148]. Second, because of this computational cost, the value function is typically computed offline based on assumed system dynamics and bounds on uncertainties. This can lead the safety analysis to be invalid or overly conservative. The newly introduced discounted safety Bellman equation [91] modifies the HJI-VI in (7.3) in a time-discounted formulation for discrete time:

$$V_S(x) = (1-\gamma)l(x) + \gamma\min\left\{l(x), \max_{u\in\mathcal{U}} V_S(x + f(x,u)\Delta t)\right\}, \quad V_S(x,0) = l(x). \tag{2.4}$$

This formulation induces a contraction mapping, which enables convergence of the value function when applied to dynamic programming schemes commonly used in RL. We note that the convergence only holds for tabular Q-learning, not for its neural counterpart [122].

As we have seen in this section, constraints can be the result of a variational approximation, it can be a logical evaluation, a cost limit, a value function, a teacher (i.e., a "critic" or a "discriminator" in other algorithmic domains), or an auxiliary prediction task (as in multi-task learning): it just needs to encompass the set of rules or constraints of interest [63, 124, 159][3]. In the first example (Natural language processing), we saw in the original formulation for posterior regularisation that f had to be pre-specified and fixed during training. Extensions eliminate this requirement, where the parameters ϕ of f can also be updated in the above M-step, from either self-supervision from the current $q^*(\mathbf{x})$, from expert demonstrations, or policy gradient updates from a reward signal [124]. In the second example (Robotics), we observed the downsides of defining safety as a cost limit constraint: the optimisation of the CMDP problem requires agents to collect experience from a diverse set of states, including unsafe ones. On the other hand, previous attempts at combining the CMDP formulation with optimal control theory (such as HJ reachability) yields methodology for defining hard safety constraints via a safety value function, but they remain limited to low

[3]We note that imposing post-hoc constraints on the predictions of the agent, alone, does *not* qualify in this discussion as learning with constraints, since the agent would not be aware of these constraints, during its optimisation.

dimensional problems and do not enable updates of safety value over time. In Section 7.2, we introduce a method that incorporates HJ reachability theory into the CMDP framework, scales to high-dimensional problems via neural functional approximation, enables safety value updates directly from visual context, and is benchmarked against state-of-the-art methods on OpenAI Safety Gym and Learn-to-Race, the new autonomous racing simulation and training environment.

2.1.4 Generalized Distillation

The introduction of domain knowledge simplifies a difficult learning paradigm, for example, when some privileged information (e.g., expert demonstrations from a teacher, artificial or otherwise) is only available during training and not during task execution time. Generalized Distillation [177] is a general framework, providing machinery for both knowledge distillation (transferring knowledge from an ensemble of possibly-large teacher models to a possibly small student model) [3, 106, 178, 284, 330] as well as learning under privileged information (LUPI; transferring experience from context that the student will not be privy to during test time) [12, 177, 286, 287].

We start with some paired data ($\{x_i, x_i^*, y_i\}$), where x_i^* and x_i are multiple views of the same input space. In fact, x_i^* can be separate modality or view, an arbitrary partition of the observation space, or a projection of x_i. In practice, though, x_i^* is thought to contain additional information from the underlying data-generating process. Using this privileged information in x_i^*, we learn a teacher network f_t:

$$f_t = \underset{f \in \mathcal{F}_t}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, \sigma(f(x_i))) + \Omega(||f||),$$

where $\sigma : \mathbb{R}^c \rightarrow \Delta^c$ is the softmax operator that produces c-dimensional probability vectors Δ^c and $\Omega : \mathbb{R} \rightarrow \mathbb{R}$ performs regularisation on the primary learning objective $\mathcal{L}(\cdot)$ (e.g., cross-entropy). We use the teacher network to generate soft labels $s = \{\sigma(f_t(x_i^*)/T)\}_{i=1}^n$, with temperature parameter T. Next, we learn the student network $f_s \in \mathcal{F}$, through multi-objective training on the original ($\{(x_i, y_i)\}_{i=1}^n$) and teacher-generated ($\{(x_i, s_i)\}_{i=1}^n$) paired data:

$$f_s = \underset{f \in \mathcal{F}_s}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} [(1-\lambda)\mathcal{L}(y_i, \sigma(f(x_i))) + \lambda \mathcal{L}(s_i, \sigma(f(x_i)))],$$

Generalized Distillation can be seen as an philosophical predecessor to imitation learning, pre-training, and multi-task learning [177], where teacher networks can seen as labelling functions, providing the student with an imitation prior from the underlying data-generating process (Figure 1.1(a)). However, exclusively using expert demonstrations can limit the student's generalisation capability to novel scenarios that are outside the support of the training distribution. As we will see later, we leverage lexicalised symbolic commonsense knowledge, in order to generate datasets as a pre-training basis for zero-shot commonsense question answering tasks (Section 4.3). We also leverage a similar imitation objective to perform variational behaviour cloning, in the context of vision-and-language navigation, for first generating a latent representation of robot navigation skills before using this skill space for regularisation during policy refinement (Section 5.3). In Section 5.2, we introduce a knowledge-driven learning approach, that leverages an imitation objective for the creation of a visuo-acoustic prior over indoor environments, in the context of semantic audio-visual robot navigation. Finally, we combine an imitation prior with distribution-regularising objectives, allowing agents to learn from an annotation-free distribution over trajectory futures (trajectory

prediction for autonomous driving; Section 6.2) and an obstacle-aware distribution over navigation waypoints (visuomotor control for autonomous driving; Section 6.3).

2.2 Multiview Representation Learning

Multiview representation learning is concerned with the notion of learning representations (often, statistical feature vector spaces) that combine and characterise and factors of variation in multiple streams of input data. These input streams may take the form of natural signals (*modalities*: structure-borne sound, audible articulation, video, images, text), e.g., from an underlying data-generating physical process (Figure 1.1a), or may simply be multiple synthetic transformations or projections of those underlying signals (*views*: text and symbolic knowledge, egocentric images and top-down scene maps, etc.). In this section, we describe various unimodal representations, then discuss how their alignment can be learned.

2.2.1 Visual, Textual, and Knowledge Representations

Visual Representations

Visual representation learning is a key component of machine perception, where a visual encoding model (an 'encoder') performs the task of extracting semantic information about entities present in structured data inputs, such as RGB images or depth frames. Encoders typically act as functional maps, thereby projecting inputs to some, e.g., lower-dimensional (denser, more-concise) latent vector space. Recent approaches use convolutional neural networks (CNNs) for this feature extraction in various applications, such as: image classification, image captioning, visual question-answering, image retrieval, object detection, semantic segmentation, depth estimation from RGB, image reconstruction, etc.; here, many such approaches rely on specific deep CNN architectures, such as VGG [262], ResNet [116], Deeplab [56], and RCNN [113]. Most modern deep CNN architectures rely on, in addition to other types of neural layers, tiers of convolutions as preliminary feature extraction [115]. High-level features appear to extract general objectness information, such as edges or extrema, while lower-level filters provide object-specific feature attributes. Traditional architectures, such as ResNet, perform regression on the final layers of the convolutional feature extractor, in order to take advantage of the most relevant and specific features. On the other hand, "hypercolumns" extract data from successive convolutional layers and align or concatenate them together before the regression layers [111], with interpolation forming the connective tissue that bridges layers of different size. This denser representation has been shown to improve performance in domains such as image segmentation [125].

Textual Representations

For language representation learning, text encoding seeks to extract semantic information from sequences of textual input, through a combination of temporal modelling strategies and projections to vector embedding spaces. Encoding strategies such as word2vec [189] and GloVe [221] are regarded as 'context-free', as they generate separate embedding vectors for each token (word) in the input sequence, independent of its surrounding tokens; contextual embeddings are generated from encoders that consider the surrounding context of tokens in the input sequence. Recurrent neural networks (RNNs), including popular variants such as Long Short-Term Memory (LSTMs) [121] and Gated Recurrent Units (GRUs) [60], were commonly-used neural network architectures for processing sequential data such as natural language. RNNs take in a sequence of inputs and generate output latent representations that summarise the

sequence, from the start of the sequence to each consecutive element (or timestep). This sequence modelling can be employed in reverse time, where the representations summarise sequences from the end, in order to better capture later temporal dependencies amongst elements; or bidirectionally, with separate sequence modelling streams for each direction, capturing both earlier and later interdependencies. Here, LSTMs and GRUs were preferred over the standard RNN formulation, as they captured longer-term sequential dependencies between earlier elements and later elements, in each directional stream. The output latent representations (or 'hidden states') may be fused with other latent representations through some vector combination (e.g., concatenation), may be influenced by other representations (often from different modalities) through attention mechanisms, or may simply be directly passed to a decoder model for downstream prediction or signal translation. Vaswani et al. [288] introduced the Transformer model, which better characterises long-term dependencies in sequential inputs, by way of several layers of multi-head self attention mechanisms and position-wise feed-forward networks, employed separately and identically on each sequential element. In additional to sequential modelling of natural language [74, 288], various works have shown success in pre-processing other types of modalities into sequences (e.g., RGB image pixel sequences), for use with Transformer-based models in computer vision applications [135], robotics tasks [69, 85], and multimodal settings [163, 258].

Knowledge Representations

Given a set of symbolic knowledge triples from a knowledge graph (Section 2.1.1), we want to map this representation to a sub-symbolic vector space, to make for easier consumption by ML models.

Knowledge graph embeddings. Given structured representation of (e.g., commonsense) knowledge in the form of triples, given by (h, r, t), various approaches encode these head h, relation r, and tail t entities as vectors or matrices. We can then perform vector operations on these representations, in order to ultimately project the overall triple to a knowledge graph embedding space [211]. Chief among the relevant approaches is the class of translational distance-based algorithms [296], such as TransE [30], where r is specifically represented as a transition vector from h to t, such that $h + r \approx t$. The RESCAL algorithm [205] encodes relations as matrices, captures the interaction between entities relations using a bi-linear scoring function, and can therefore capture complex patterns over multiple hops. HolE [206] extends RESCAL, improving on its space and time complexity, without sacrificing its the expressivity of the resultant representation. Whereas HolE only represents r as a vector, the approach captures pairwise interactions of entities as composable vectors, through a circular correlation operation.

Grounded knowledge statements. For ConceptNet knowledge triples (see Section 2.1.1), we first convert concept-relation tokens into regular tokens, in order to generate a pseudo-sentence, e.g.: "*(book, AtLocation, library)*" would be converted to "book at location library." Next, we use the BERT [74] embedding layer to generate an embedding of this pseudo-sentence, with C denoting a ConceptNet relation:

$$H_C = \text{BiLSTM}(C) \tag{2.5}$$

If we let $H_C \in \mathbb{R}^{1\times 2l}$ be the concatenation of the final hidden states and l be the number of hidden units in the LSTM layer, then m ConceptNet relations would yield the commonsense knowledge matrix $H_M \in \mathbb{R}^{m\times 2l}$. We adopt the attention mechanism used in QAnet [320] to model the interaction between

H_M and the BERT encoding output T_{enc}:

$$\widetilde{H}_M = H_M \cdot W_{proj} \tag{2.6}$$

$$\mathcal{S} = \text{Att}(H_M, T_{enc}) \tag{2.7}$$

$$A_m = \text{softmax}(\mathcal{S}) \cdot \widetilde{H}_M \tag{2.8}$$

$$A_t = \text{softmax}(\mathcal{S}) \cdot \text{softmax}(\mathcal{S}^T) \cdot T_{enc} \tag{2.9}$$

$$T_C = [T_{enc}; A_m; T_{enc} \circ A_m; T_{enc} \circ A_t] \tag{2.10}$$

$$T_{out} = \text{ReLU}(T_C \cdot W_a) \tag{2.11}$$

Specifically, H_M is first projected into the same dimension as T_{enc}, using $W_{proj} \in \mathbb{R}^{2l \times d}$. Then, the similarty matrix $\mathcal{S} \in \mathbb{R}^{n \times m}$ is computed using tri-linear attention, as in Equation 4.2. We then use $\mathcal{S}$ to compute text-to-knowledge attention $A_m \in \mathbb{R}^{n \times d}$ and knowledge-to-text attention $A_t \in \mathbb{R}^{n \times d}$. Finally, the knowledge-aware textual representation $T_{out} \in \mathbb{R}^{n \times d}$ is computed, where $W_a \in \mathbb{R}^{4d \times d}$. T_{out} is fed to subsequent layers (in place of T_{enc}), in order to generate the prediction.

2.2.2 Learning to Align Representations from Multiple Views

Because a single physical process generates multiple modalities that act in observation of its events (or because multiple views may be related through a family of transformations or structured perturbations), these views are said to share the property of *complementarity*—exhibiting pairwise interdependencies in their signal content, conditioned on events from the data-generating process. Harnessing this complementarity between views is often concerned with performing the task of multiview *alignment*: i.e., finding these relationships between sub-components of instances of two or more views, in a way that confirms inclusion of domain knowledge and/or contributes to some measurable downstream performance improvement.

In many problem domains of interest—from natural language processing, to multimodal robot instruction-following, to autonomous driving—an artificial agent is often presented with collections of modalities that aid in its execution of one or more downstream tasks. Further, we may wish to encode instances of each modality using parameterised and learnable functions (such as neural networks) as more concise summaries of the input, essentially mapping the input modalities to their own respective vector-based representation spaces (see Section 2.2.1, above). We may then combine these representations, in order to provide the agent with a more holistic characterisation of its context. The hope is that this unified context leads to better scene understanding and, in turn, improved task performance and generalisation.

Typically, this combination happens through a series of projections and learning objective-based constraints, e.g., (i) by using a single projection function (can also be a neural network) to map all the feature vectors to a joint multiview representation; (ii) projecting the features to their own individual (and often lower-dimensional) spaces, then coordinating them by imposing a relatedness constraint, such as maximising correlation or mutual information, minimising cosine distance, reducing distributional divergence, etc.; and finally, (iii) learning to sub-select relevant elements in one feature space, given some conditioning information from another, as in cross-view attention ("soft" coordination) and grounding.

Cross-view grounding through attention mechanisms has gained significant popularity recently, particularly in the context of embodied vision-language planning tasks (EVLP) [94], wherein the vision-based and language-based modalities have an especially additive contribution to instruction-following [9, 181], embodied question answering [68], and goal-directed manipulation [146, 259].

Chapter 3

Focus Areas

The scope of application for knowledge-enhanced modelling techniques is very broad and cannot be reasonably studied within just one domain; in past work, we consider several. In [93, 171, 172, 201, 202, 203, 321], we have examined how domain knowledge about human anthropomorphic features can inform occupant detection, counting, and the prediction of occupant thermal comfort preferences, in indoor commercial buildings settings. In [130], we use domain knowledge to understand the dependency structures that characterise the thermal dynamics, for commercial building automation, in order to reduce the severity of intermittent prediction disturbances. In [232], we operationalise expert annotations in order to generate diverse lexical paraphrases for supervised semantic parsing. In [49, 50], we harness domain knowledge about how to decompose sets of non-linear objectives, for optimising thermostatically controlled loads in smart grids. In this thesis, we consider knowledge-enhanced learning in the context of three additional application areas: (i) Neuro-symbolic Commonsense Reasoning (Chapter 4), (ii) Embodied Multimodal Robot Navigation (Chapter 5), and (iii) Autonomous Driving (Chapters 6, 7).

3.1 Neuro-symbolic Commonsense Reasoning

Architectures for Commonsense Question Answering

Non-extractive commonsense QA remains a challenging AI task, as it requires systems to reason about, synthesise, and gather disparate pieces of information, in order to generate responses to queries. Recent approaches on such tasks show increased performance, only when models are either pre-trained with additional information or when domain-specific heuristics are used, without any special consideration regarding the knowledge resource type. In this chapter, we perform a survey of recent commonsense QA methods and we provide a systematic analysis of popular knowledge resources and knowledge-integration methods, across benchmarks from multiple commonsense datasets. Our results show that attention-based injection is a preferable choice for commonsense knowledge integration and that the degree of domain overlap, between knowledge bases and datasets, plays a crucial role in determining model success.

Data Construction for Zero-shot Evaluation, in Commonsense Question Answering

Recent developments in pre-trained neural language modelling have led to leaps in accuracy on commonsense question-answering benchmarks. However, whereas large-capacity neural systems are able

to model individual datasets, there is increasing concern that large-capacity language models overfit to specific tasks, without learning to utilise external knowledge or perform general semantic reasoning. In contrast, zero-shot evaluations have shown promise as a more robust measure of a model's general reasoning abilities, as models that achieve state-of-the-art performance on individual datasets suffer significant performance-degradation under zero-shot evaluation on new, but similar tasks. In this chapter, we propose a novel neuro-symbolic framework for zero-shot question answering across commonsense tasks. Guided by a set of hypotheses, the framework studies how to transform various pre-existing knowledge resources into a form that is most effective for pre-training models. We vary the set of language models, training regimes, knowledge sources, and data generation strategies, and measure their impact across tasks. Extending on prior work, we devise and compare four constrained distractor-sampling strategies. We provide empirical results across five commonsense question-answering tasks, with data generated from five external knowledge resources. We show that, while an individual knowledge graph is better suited for specific tasks, a global knowledge graph brings consistent gains across different tasks. In addition, both preserving the structure of the task as well as generating fair and informative questions help language models learn more effectively.

3.2 Embodied Multimodal Robot Navigation

Knowledge-driven Scene Priors for Semantic Audio-Visual Navigation

Generalisation to unseen contexts remains a challenge for embodied navigation agents. In the context of semantic audio-visual navigation (SAVi) tasks, generalisation includes *both* generalising to unseen indoor visual scenes as well as generalising to unheard sounding objects. Previous SAVi task definitions do not include evaluation conditions on truly novel sounding objects, resorting instead to evaluating agents on unheard sound clips of known objects; meanwhile, previous SAVi methods do not include explicit mechanisms for incorporating domain knowledge about object and region semantics. These weaknesses limit the development and assessment of models' abilities to generalise their learned experience. In this work, we introduce the use of knowledge-driven scene priors in the semantic audio-visual embodied navigation task: we combine semantic information from our novel knowledge graph that encodes object-region relations, spatial knowledge from dual Graph Convolutional Networks, and background knowledge from a series of pre-training tasks—all within a reinforcement learning framework for audio-visual navigation. We define a new audio-visual navigation sub-task, where agents are evaluated on novel sounding objects, as opposed to unheard clips of known objects.

Learning Representations for Reusable Robot Maneuvers

Vision-Language Navigation (VLN) tasks remain challenging for artificial agents, as they must satisfy complex natural language instructions to navigate within complex photorealistic, partially-observable environments. Recent pursuit of this task has focused on pragmatic decoding, agent progress-monitoring, back-translation, and cross-modal grounding; however, the aforementioned issues with instruction complexity and label-independent variation in the visual features still persist, reducing agents' generalisation performance in unseen environments. In this chapter, we propose a framework for distilling a navigation agent's experience into a representation of reusable maneuvers, where the high-frequency and label-independent variation in the instructions and visual context are removed. Our approach encourages the agent to generalise reusable navigation maneuvers on the basis of similarities across high-level textual instructions, conditioned on past actions, in order to leverage experience in executing familiar

sub-commands from new instructions. We achieve this association through a method called auxiliary variable variation approximation, which allows us to introduce a latent variable for estimating the conditional posterior distribution over all observations and executed trajectories (otherwise intractable for most distributions of interest). The agent then learns how to compose samples from this skill space and couple them with conventional multimodal co-grounding mechanisms, through policy refinement, for improved generalisability and downstream performance. We perform an error analysis to illustrate the robustness of models' generalisation to complex scenarios, and we show improvements from our approach.

3.3 Autonomous Driving: Multimodal Perception & Control

Diverse and Admissible Trajectory Forecasting

Multi-agent trajectory forecasting in autonomous driving requires an agent to accurately anticipate the behaviours of the surrounding vehicles and pedestrians, for safe and reliable decision-making. Due to partial observability over the goals, contexts, and interactions of agents in these dynamical scenes, directly obtaining the posterior distribution over future agent trajectories remains a challenging problem. In realistic embodied environments, each agent's future trajectories should be *diverse* since multiple plausible sequences of actions can be used to reach its intended goals, and they should be *admissible* since they must obey physical constraints and stay in drivable areas. In Section 6.2, we propose a model that fully synthesises multiple input signals from the multimodal world—the environment's scene context and interactions between multiple surrounding agents—to best model all diverse and admissible trajectories. We offer new metrics to evaluate the diversity of trajectory predictions, while ensuring admissibility of each trajectory. Based on our new metrics as well as those used in prior work, we compare our model with strong baselines and ablations across two datasets and show a 35% performance-improvement over the state-of-the-art.

Distribution-aware Goal Prediction for Autonomous Driving

The feasibility of collecting a large amount of expert demonstrations has inspired growing research interests in learning-to-drive settings, where models learn by imitating the driving behaviour from experts. However, exclusively relying on imitation can limit agents' generalisability to novel scenarios that are outside the support of the training data. In this paper, we address this challenge by decomposing the driving task into modular skill primitives, based on the intuition that such architecture is more generalizable and more robust to changes in the environment than a monolithic one. Specifically, we draw inspiration from the trajectory forecasting community and reformulate the learning-to-drive task as goal distribution prediction, model-based planning, and trajectory pruning. Firstly, we learn a multi-modal goal distribution by imitating the expert conditioned on reaching the target location. Next, we ground candidate trajectory predictions on vehicle kinematics and road geometry. At each time step, we sample multiple goals and at the same time prune the predictions that are spurious. Under the CARLA simulator, we report new state-of-the-art results on the new CARNOVEL benchmark for assessing model robustness to out-of-distribution scenarios.

Safety-aware Policy Optimisation via Constraint Functions

To be viable for safety-critical applications, such as autonomous driving and assistive robotics, autonomous agents should adhere to safety constraints, throughout interactions with their environments. Furthermore,

the simulated environments themselves must be realistic enough to facilitate training that induces the desired qualities of agents. Whereas much of the existing research in autonomous driving focuses on urban or highway driving, simulated *Formula*-style track racing represents new challenges for artificial agents that must learn complex driving behaviour, safely and sample-efficiently, in rapid time-evolving scenarios. Contemporary works in simulation struggle to capture realism in the visual rendering, vehicular dynamics, and task objectives, inhibiting the transfer of learning agents to real-world contexts. Meanwhile, current solutions are split between more classical approaches that demand privileged information, require significant parameter-tuning, and are limited in their performance capacity; versus approximate driving methods that provide no guarantees of safety and are prone to overfitting on the training scenarios.

In Section 7.2 we study how to address both critical challenges in this task. First, to address the lack of realistic simulators and tasks for studying high-speed driving, we release a new simulator and OpenAI-gym compliant training environment for simulated *Formula*-style racing, which enables agents to learn to race on high-precision models of real-world tracks (e.g., the famed Thruxton Circuit and the Las Vegas Motor Speedway) and to use a suite of rich multimodal sensory information. We also propose the `Learn-to-Race` (`L2R`) AI task with challenging metrics, inspired by learning-to-drive challenges, *Formula*-style racing, and vehicle trajectory forecasting; we provide an official `L2R` task dataset of expert demonstrations and a series of baseline experiments and reference implementations. Next, to address the challenges in safe policy optimisation, we propose the use of safety constraints for autonomous racing, inspired by the theoretical foundations of Hamilton-Jacobi (HJ) reachability analysis in optimal control. Here, we define a safety controller that intervenes whenever an agent approaches bad states, and we show that even an agent that generates actions at random is guaranteed to stay on the drivable area; we further show that an arbitrary learning policy that is coupled with this safe controller is able to learn performant driving behaviour, both safely and sample-efficiently. Finally, we demonstrate that the HJ safety value can be learned and updated directly from vision context, thereby expanding HJ reachability to applications where high-fidelity dynamics models may not be available. While not necessary for convergence, we warm-start the safety value function using values pre-computed with a nominal model. The safety value is updated directly on transitions of ego-agent's frontal camera view and vehicle speed. As a first experiment, we evaluate our approach on alongside strong baselines, in two environment and agent configurations, on the OpenAI Safety Gym framework; we report the minimum number of safety infractions, compared to state-of-the-art CMDP approaches. In the second experiment, we evaluate our approach on `Learn-to-Race` (`L2R`) [117], a recently-released high-fidelity autonomous racing environment, which requires the vehicle to make safety-critical decision in a fast changing environment. We obtain state-of-the-art results on `L2R` and show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

Chapter 4

Learning with Common Sense

for Neuro-symbolic Reasoning

4.1 Motivation

Understanding how to represent and incorporate symbolic commonsense knowledge in learning-based systems is crucial for their improved optimisation and generalisability. Whereas recent developments in large-capacity, pre-trained neural language modelling have led to leaps in accuracy on commonsense question-answering benchmarks, there is increasing concern that models overfit to specific tasks, without learning to utilise external knowledge or perform general semantic reasoning [183]. Even when commonsense knowledge is considered by prior work in neural language modelling, many works take a coarse-grained view of commonsense, without considering the subtle differences across the various knowledge types and resources. Such differences have been discussed at length in AI by philosophers, computational linguists, cognitive psychologists (see for instance [70]): at the high level, we can identify *declarative commonsense*, whose scope encompasses factual knowledge, e.g., 'the sky is blue', 'Paris is in France'; *taxonomic knowledge*, e.g., 'football players are athletes', 'cats are mammals'; *relational knowledge*, e.g., 'the nose is part of the skull', 'handwriting requires a hand and a writing instrument'; *procedural commonsense*, which includes prescriptive knowledge, e.g., 'one needs an oven before baking cakes', 'the electricity should be off while the switch is being repaired' [120]; *sentiment knowledge*, e.g., 'rushing to the hospital makes people worried', 'being in vacation makes people relaxed'; and *metaphorical knowledge* (e.g., 'time flies', 'raining cats and dogs'). We believe that it is important to identify the most appropriate commonsense knowledge type required for specific tasks, in order to get better downstream performance. Once the knowledge type is identified, we can then select the appropriate knowledge-base(s) and the suitable neural integration mechanisms.

Many options exist for incorporating commonsense as domain knowledge in a reasoning task, such as commonsense question answering and constrained text generation. Furthermore, many challenges would befall methods that do not take care to utilise the knowledge effectively: jointly learning from context-free knowledge embeddings or directly concatenating knowledge-grounded triples as additional context (Section 2.1.1) can often have negative effects on models' learned representations. Because this information can vary with high frequency, with respect to the task context, models may learn to ignore the knowledge

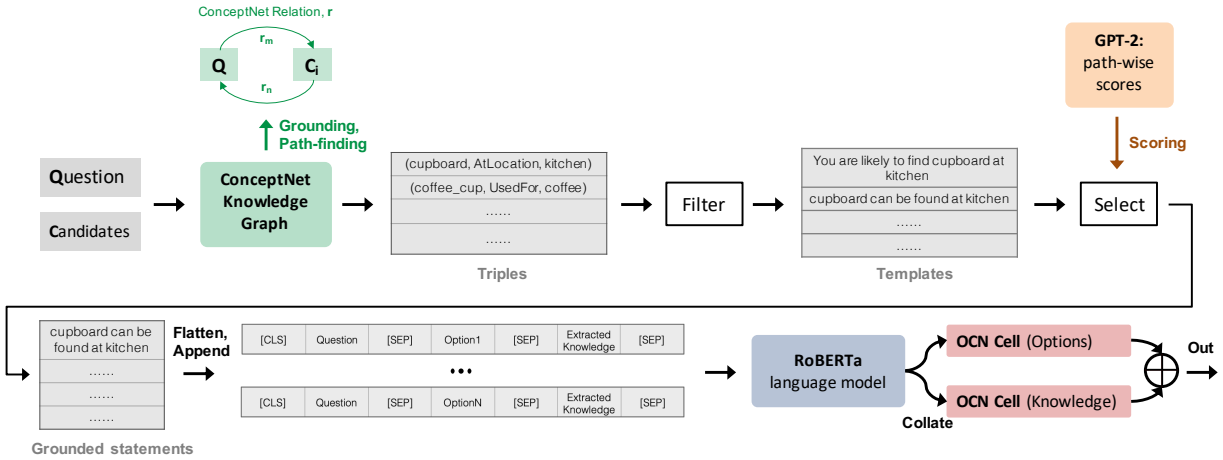


Figure 4.1: Knowledge extraction pipeline.

in favour of the comparatively lower-frequency learning signal in the samples. In this chapter, we consider various strategies for utilising commonsense knowledge for training learning-based systems, in the context of commonsense question-answering and constrained text generation. In Section 4.2, we study the neural architectural mechanisms for leveraging common sense, then consider the importance of knowledge-task alignment. We characterise knowledge utilisation in two stages, knowledge-extraction (from, e.g., an external resource) and knowledge-injection (into, e.g., a neural language model; via attention mechanisms), then provide experiments, ablations, and error analysis accordingly. In Section 4.3, we study a different usage of commonsense knowledge: that is, using it to generate synthetic datasets that can serve as strong pre-training bases for generalisation. We describe our synthetic QA set generation procedure and provide empirical results, across five commonsense question-answering tasks, with data generated from five external knowledge resources, comparing four QA distractor-sampling strategies.

4.2 Neuro-symbolic Architectures for Commonsense QA

With the recent success of large pre-trained language models [74, 175, 229, 318], model performance has reached or surpassed human-level capability on many previous question-answering (QA) benchmarks [118, 154, 230]. However, these benchmarks do not directly challenge model reasoning capability, as they require only marginal use of external knowledge to select the correct answer, i.e., all the evidence required to solve questions in these benchmarks is explicit in the context lexical space. Efforts have been made towards building more challenging datasets that, by design, require models to synthesise external commonsense knowledge and leverage more sophisticated reasoning mechanisms [212, 327], showing that the previous state-of-the-art models often struggle to solve these newer tasks reliably. As a result, commonsense has received a lot of attention in other areas as well, such as natural language inference [323, 325] and visual question answering [324]. Despite the importance of commonsense knowledge, however, previous work on QA methods takes a coarse-grained view of commonsense, without considering the subtle differences across the various knowledge types and resources.

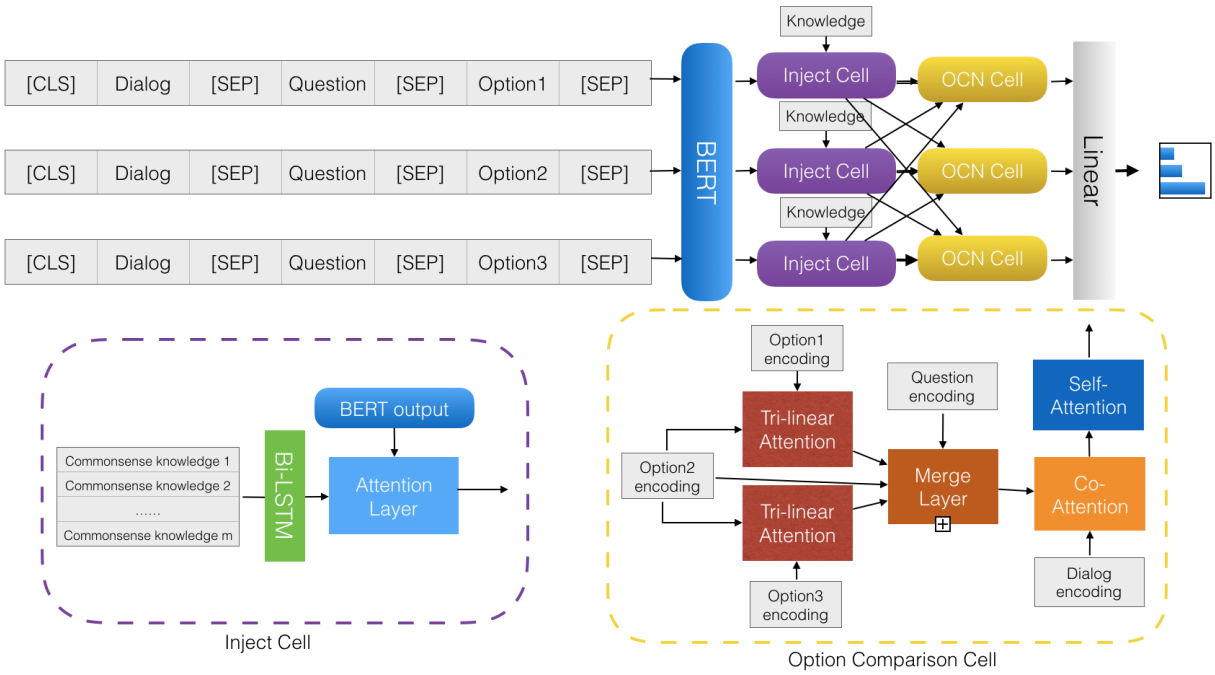


Figure 4.2: Option Comparison Network with Knowledge Injection

4.2.1 Neuro-symbolic Structures

The BERT model [74] has been applied to numerous QA tasks and has achieved very promising performance, particularly on the `DREAM` and `CommonsenseQA` datasets. When utilising BERT on multiple-choice QA tasks, the standard approach is to concatenate the dialogue context and the question with each answer-option, in order to generate a list of tokens which is then fed into BERT encoder; a linear layer is added on top, in order to predict the answer. One aspect of this strategy is that each answer-option is encoded independently: from a cognitive perspective, this aspect contradicts how humans typically solve multiple-choice QA tasks, namely by *weighing* each option to find correlations within them, in addition to correlations with respect to the question. To address this issue, Ran et al. [231] introduced the *Option Comparison Network* (OCN) that explicitly models pairwise answer-option interactions, making OCN better-suited for multiple-choice QA task structures. We re-implemented OCN while keeping BERT as its upstream encoder. Specifically, given a dialogue D, a question Q, and an answer-option O_k, we concatenate them and encode with BERT to get hidden representation $T_{enc} \in \mathbb{R}^{n \times d}$:

$$T_{enc} = \text{BERT}(D; Q; O_k) \tag{4.1}$$

Where d is the size of BERT's hidden representation and n is the total number of words. Next, the dialogue encoding $D_{enc} \in \mathbb{R}^{n_d \times d}$, question encoding $Q_{enc} \in \mathbb{R}^{n_q \times d}$, and answer-option encoding $O_{k,enc} \in \mathbb{R}^{n_o \times d}$ are separated from T_{enc}. Here, option-encoding consists both of question and option, i.e. $Q_{enc} \subseteq O_{k,enc}$ and $n_d + n_o = n$, as suggested by Ran et al. [231]. Given a set of options O_k ($k = 1, 2, ...$), these options are compared, pairwise, using standard tri-linear attention [254]:

$$\text{Att}(u, v) = W_1 \cdot u + W_2 \cdot v + (W_3 \circ v) \cdot u \tag{4.2}$$

Where, $W_1, W_2, W_3 \in \mathbb{R}^d$ are trainable weights and $u \in \mathbb{R}^{x \times d}$, $v \in \mathbb{R}^{y \times d}$ are input matrices; x and y here are generic placeholder for input lengths; matrix multiplication and elementwise multiplication are

Table 4.1: An example from the DREAM dataset, which assesses models' abilities to perform commonsense reasoning. The asterisk (*) denotes the correct answer.

Dialogue:
M: I hear you drive a long way to work every day.
W: Oh, yes. it's about sixty miles. but it doesn't seem that far, the road is not bad, and there's not much traffic.
Question:
How does the woman feel about driving to work?
Answer choices:
A. She doesn't mind it as the road conditions are good.*
B. She is unhappy to drive such a long way everyday.
C. She is tired of driving in heavy traffic.

Table 4.2: An example from the CommonsenseQA dataset, which assesses models' abilities to perform commonsense reasoning. The asterisk (*) denotes the correct answer.

Question:
A revolving door is convenient for two direction travel, but it also serves as a security measure at a what?
Answer choices:
A. Bank*; B. Library; C. Department Store; D. Mall; E. New York

denoted by $(\cdot)$ and $(\circ)$, respectively. Next, we gather information from all other options, to form a new option representation $O_{k,new} \in \mathbb{R}^{n_o \times d}$. Formally, given option $O_{k,enc}$ and another option $O_{l,enc} \in \mathbb{R}^{n_l \times d}$, $O_{k,new}$ is computed as follows:

$$O_k^l = O_{l,enc} \cdot \text{Att}(O_{l,enc}, O_{k,enc}) \tag{4.3}$$

$$\widetilde{O_k^l} = [O_{k,enc} - O_k^l; O_{k,enc} \circ O_k^l] \tag{4.4}$$

$$O_{k,new} = \tanh(W_c \cdot [O_{k,enc}; \{\widetilde{O_k^l}\}_{l \neq k}]) \tag{4.5}$$

Where, $W_c \in \mathbb{R}^{(d+2d(|O|-1)) \times d}$, $|O|$ denotes total number of options and n_l denotes the number of words in the compared option. Then, a gating mechanism is used to fuse the option-wise correlation information $O_{k,new}$ with the current option-encoding $O_{k,enc}$. Gating values are computed as:

$$G = \text{sigmoid}(W_g[O_{k,enc}; O_{k,new}; \widetilde{Q}]) \tag{4.6}$$

$$\widetilde{Q} = Q_{enc} \cdot \text{softmax}(Q_{enc} \cdot V_a)^T \tag{4.7}$$

$$O_{fuse} = G \circ O_{k,enc} + (1 - G) \circ O_{k,new} \tag{4.8}$$

Here, $W_g \in \mathbb{R}^{3d \times d}$ and $V_a \in \mathbb{R}^{d \times 1}$. Co-attention [311] is applied to re-read the dialogue, given the fused option-correlation features:

$$A_{do} = \text{Att}(D_{enc}, O_{fuse}) \tag{4.9}$$

$$A_{od} = \text{Att}(O_{fuse}, D_{enc}) \tag{4.10}$$

$$O_d = A_{od} \cdot [D_{enc}; A_{do} \cdot O_{fuse}] \tag{4.11}$$

$$\widetilde{O_d} = \text{ReLU}(W_p([O_d; O_{fuse}])) \tag{4.12}$$

Here, $W_p \in \mathbb{R}^{3d \times d}$. Finally, self-attention [298] is used to compute final option representation $\widetilde{O_f} \in \mathbb{R}^{n_o \times d}$:

$$O_s = \widetilde{O_d} \cdot \text{Att}(\widetilde{O_d}, \widetilde{O_d}) \tag{4.13}$$

$$O_f = [\widetilde{O_d}; O_s, \widetilde{O_d} - O_s; \widetilde{O_d} \circ O_s] \tag{4.14}$$

$$\widetilde{O_f} = \text{ReLU}(W_f \cdot O_f) \tag{4.15}$$

Unlike the vanilla BERT model, which takes the first token to predict the answer, max-pooling is applied on the sequence dimension of $\widetilde{O_f} \in \mathbb{R}^{n_o \times d}$, in order to generate the final prediction.

4.2.2 Knowledge Bases

The first knowledge-base we consider for our experiments is ConceptNet [264]. ConceptNet contains over 21 million edges and 8 million nodes (1.5 million nodes in the partition for the English vocabulary), generating triples of the form $(C1, r, C2)$: the natural-language concepts $C1$ and $C2$ are associated by commonsense relation r, e.g., *(dinner, AtLocation, restaurant)*. Thanks to its coverage, ConceptNet is one of the most popular semantic networks for commonsense. ATOMIC [250] is a knowledge-base that focuses on procedural knowledge: triples are of the form *(Event, r,* {*Effect*|*Persona*|*Mental-state*}), where head and tail are short sentences or verb phrases and r represents an *if-then* relation type. An example would be: *(X compliments Y, xIntent, X wants to be nice)*. Since both `DREAM` and `CommonsenseQA` datasets are open-domain and require general commonsense, we think these knowledge-bases are most appropriate for our investigation.

4.2.3 Knowledge Elicitation from External Resources

ConceptNet. For the `DREAM` dataset, we find ConceptNet relations that connect dialogues and questions to the answer-options. The intuition is that these relation paths would provide explicit evidence that would help the model find the answer. Formally, given a dialogue D, a question Q, and an answer-option O, we find all ConceptNet relations *(C1, r, C2)*, such that $C1 \in (D + Q)$ and $C2 \in O$, or vice versa. This rule works well for single-word concepts. However, a large number of concepts in ConceptNet are actually phrases, and finding exactly matching phrases in $D/Q/O$ is much harder. To fully utilise phrase-based ConceptNet relations, we relaxed the exact-match constraint to the following:

$$\frac{\text{\# words in C} \cap \text{S}}{\text{\# words in C}} > 0.5 \tag{4.16}$$

Here, S represents $D/Q/O$, depending on which sequence we try to match the concept C to. Additionally, when the part-of-speech (POS) tag for a concept is available, we make sure it matches the POS tag of the

Table 4.3: Extracted ConceptNet relations for sample shown in Table 4.2.

Options	**Extracted ConceptNet triples**
Bank	(revolving door *AtLocation* bank) **(bank RelatedTo security)**
Library	(revolving door *AtLocation* library)
Department Store	(revolving door *AtLocation* store) (security IsA department)
Mall	(revolving door *AtLocation* mall)
New York	(revolving door *AtLocation* New York)

Table 4.4: Sample generated ATOMIC relations for sample shown in Table 4.1.

Input sentence	**Generated ATOMIC relations**
Utterance 1	(xAttr dedicated) (xWant to get to work)
Utterance 2	(xAttr far) **(xReact happy)** (xWant to get to their destination)
Option A	**(xAttr calm)** (xWant to avoid the road)
Option B	**(xAttr careless)** (xReact annoyed) (xEffect get tired)
Option C	**(xAttr frustrated)** (xEffect get tired) (xWant to get out of car)

corresponding word in $D/Q/O$. For `CommonsenseQA`, we use the same procedure to find ConceptNet relations for each answer-option, except that only Q is present and used. Table 4.3 shows the extracted ConceptNet triples for the `CommonsenseQA` example in Table 4.2. It is worth noting that we are able to extract the original ConceptNet sub-graph that was used to create the question, along with some extra triples. Although not perfect, the bold ConceptNet triple does provide some clue that could help the model resolve the correct answer.

ATOMIC. We observe that many questions in `DREAM` inquire about agent's opinion and feeling. Superficially, this particular question type seems well-suited for ATOMIC, whose focus is on folk psychology and related general implications; we could frame our goal as evaluating whether ATOMIC can provide relevant knowledge to help answer these questions. However, one challenge to this strategy is that heads and tails of knowledge triples in ATOMIC are short sentences or verb phrases, while rare words and person-references are reduced to blanks and PersonX/PersonY, respectively. This calls for a new matching procedure, different from the ConceptNet extraction strategy, for eliciting ATOMIC-specific relations: we rely on the recently-published COMET model [33] to generate new ATOMIC relations, with intermediate phrasal resolutions. In particular, we first segmented all dialogues, questions, and answer-options into sentences. We further segment long sentences into sub-sentences, using commas as seperators. Because only verb-phrases satisfy the definition of an "event" in ATOMIC (i.e., relations are only invoked by verbs), we remove all sentences/sub-sentences that do not contain any verb. Next, we use a pre-trained COMET model [33] to generate all possible ATOMIC relations, for all candidate sentences/sub-sentences and we use greedy-decoding to take the 1-best sequences. Table 4.4 shows the sample ATOMIC relations, generated using the `DREAM` example in Table 4.1. It is interesting to note that the reaction for the woman agent (second utterance) is identified as *happy*, since she said that 'the road is not bad.' If we compare the identified attributes for answer-options, the one from correct answer seems to be sematically closer than the other two.

4.2.4 Knowledge Injection into Neural Models

Given previously extracted/generated knowledge triples, we need to integrate them with the OCN model. Inspired by Bauer et al. [22], we propose to use attention-based injection. For ConceptNet knowledge triples, we first convert concept-relation tokens into regular tokens, in order to generate a pseudo-sentence. For example, "*(book, AtLocation, library)*" would be converted to "book at location library." Next, we use the BERT embedding layer to generate an embedding of this pseudo-sentence, with C denoting a ConceptNet relation:

$$H_C = \text{BiLSTM}(C) \tag{4.17}$$

If we let $H_C \in \mathbb{R}^{1\times 2l}$ be the concatenation of the final hidden states and l be the number of hidden units in the LSTM layer, then m ConceptNet relations would yield the commonsense knowledge matrix $H_M \in \mathbb{R}^{m\times 2l}$. We adopt the attention mechanism used in QAnet [320] to model the interaction between H_M and the BERT encoding output T_{enc} (from Equation 4.1):

$$\widetilde{H}_M = H_M \cdot W_{proj} \tag{4.18}$$

$$\mathcal{S} = \text{Att}(H_M, T_{enc}) \tag{4.19}$$

$$A_m = \text{softmax}(\mathcal{S}) \cdot \widetilde{H}_M \tag{4.20}$$

$$A_t = \text{softmax}(\mathcal{S}) \cdot \text{softmax}(\mathcal{S}^T) \cdot T_{enc} \tag{4.21}$$

$$T_C = [T_{enc}; A_m; T_{enc} \circ A_m; T_{enc} \circ A_t] \tag{4.22}$$

$$T_{out} = \text{ReLU}(T_C \cdot W_a) \tag{4.23}$$

Specifically, H_M is first projected into the same dimension as T_{enc}, using $W_{proj} \in \mathbb{R}^{2l\times d}$. Then, the similarty matrix $\mathcal{S} \in \mathbb{R}^{n\times m}$ is computed using tri-linear attention, as in Equation 4.2. We then use $\mathcal{S}$ to compute text-to-knowledge attention $A_m \in \mathbb{R}^{n\times d}$ and knowledge-to-text attention $A_t \in \mathbb{R}^{n\times d}$. Finally, the knowledge-aware textual representation $T_{out} \in \mathbb{R}^{n\times d}$ is computed, where $W_a \in \mathbb{R}^{4d\times d}$. T_{out} is fed to subsequent layers (in place of T_{enc}), in order to generate the prediction. The model structure with knowledge-injection is summarized in Figure 4.2.

For ATOMIC knowledge triples, the injection method is slightly different. Because heads of these knowledge triples are sentences/utterances and the tails contain attributes of the persons (i.e., subject and object of the sentence), it is not possible to directly inject the knowledge triples, as-is. We replace the heads of the ATOMIC knowledge triples with the corresponding speaker for dialogues and leave as blank for the answer-options. Next, we convert the special relation tokens into regular tokens, e.g., "xIntent"⇒"intent" and "oEffect"⇒ "others effect", to make pseudo-sentences. As a result, an ATOMIC relation "*(the road is not bad, xReact, happy)*" would be converted to "*(W, react, happy)*." Moreover, as the ATOMIC knowledge triples are associated with dialogues and answer-options, independently, we inject option relations into $O_{enc} \in \mathbb{R}^{n_o\times d}$ and dialogue relations into D_{enc}, respectively, using the injection method described above.

4.2.5 Knowledge Pre-training

Pre-training large-capacity models (e.g., BERT, GPT [229], XLNet [318]) on large corpora, then fine-tuning on more domain-specific information, has led to performance improvements on various tasks. Inspired by this, our goal in this sub-section is to observe the effect of pre-training BERT on commonsense knowledge and refining the model on task-specific content from the `DREAM` and `CommonsenseQA` corpora. Essentially, we would like to test if pre-training on our external knowledge resources can help the

Table 4.5: Results on DREAM; the asterisk (*) denotes results taken from leaderboard.

Models	Dev Acc	Test Acc
BERT Large(*)	66.0	66.8
XLNet(*)	-	**72.0**
OCN	70.0	69.8
OCN + CN injection	**70.5**	69.6
OCN + AT injection	69.6	**70.1**
OCN + OMCS pre-train	64.0	62.6
OCN + ATOMIC pre-train	60.3	58.8

Table 4.6: Results on CommonsenseQA; the asterisk (*) denotes results taken from leaderboard.

Models	Dev Acc
BERT + OMCS pre-train(*)	68.8
RoBERTa + CSPT(*)	**76.2**
OCN	64.1
OCN + CN injection	67.3
OCN + OMCS pre-train	65.2
OCN + ATOMIC pre-train	61.2
OCN + OMCS pre-train + CN inject	**69.0**

model acquire commonsense. For the ConceptNet pre-training procedure, pre-training BERT on pseudo-sentences formulated from ConceptNet knowledge triples does not provide much gain on performance. Instead, we trained BERT on the *Open Mind Common Sense* (OMCS) corpus [263], the originating corpus that was used to create the ConceptNet resource. We extracted 930K English sentences from OMCS and randomly masked out 15% of the tokens; we then fine-tuned BERT, using a masked language model objective. Then we load this fine-tuned model into OCN and trained on DREAM and CommonsenseQA tasks. As for pre-training on ATOMIC, we again use COMET to convert ATOMIC knowledge triples into sentences; we created special tokens for 9 types of relations as well as blanks. Next, we randomly masked out 15% of the tokens, only masking out tail-tokens. We use the same OMCS pre-training procedure.

4.2.6 Experiments

Datasets

We choose to evaluate our hypotheses using the DREAM and CommonsenseQA datasets, because some / all questions require commonsense reasoning and because there remains a large gap between state-of-the-art models and human performance. DREAM is a dialogue-based multiple-choice QA dataset, introduced by Sun et al. [270]. It was collected from English-as-a-foreign-language examinations, designed by human experts. The dataset contains 10,197 questions for 6,444 dialogues in total, and each question is associated with 3 answer-options. The authors point out that 34% of questions require commonsense knowledge to answer, which includes social implication, speaker's intention, or general world knowledge. CommonsenseQA is a multiple-choice QA dataset that specifically measure commonsense reasoning

Table 4.7: Accuracies for each DREAM question type: **M** means *Matching*, **S** means *Summary*, **L** means *Logic inference*, **C** means *Commonsense inference*, and **A** means *Arithmatic inference*. Numbers beside types denote the number of questions of that type.

Models	**M(54)**	**S(15)**	**A+L(11)**	**L(228)**	**C+L(122)**	**C(14)**	**C+S(60)**
OCN	88.9	86.7	27.3	75.9	60.7	71.4	70.0
OCN + CN injection	83.3(-5.6)	86.7(+0.0)	18.2(-9.2)	76.8(+0.9)	59.8(-0.9)	64.3(-7.1)	78.3(**+8.3**)
OCN + AT injection	88.9(+0.0)	80.0(-6.7)	27.3(+0.0)	75.9(+0.0)	66.4(**+5.7**)	71.4(+0.0)	75(**+5.0**)
OCN + OMCS pre-train	70.4(-18.5)	73.3(-13.4)	45.4(+18.1)	69.7(-6.2)	48.4(-12.3)	57.1(-14.3)	68.3(**-1.7**)
OCN + ATOMIC pre-train	66.6(-22.3)	86.7(+0.0)	18.2(-9.2)	64.0(-11.9)	51.6(-9.1)	42.9(-28.5)	70.0(**+0.0**)

[275]. This dataset is constructed based on ConceptNet [264]. Specifically, a source concept is first extracted from ConceptNet, along with 3 target concepts that are connected to the source concept, i.e., a sub-graph. Crowd-workers are then asked to generate questions, using the source concept, such that only one of the target concepts can correctly answer the question. Additionally, 2 more distractor concepts are selected by crowd-workers so that each question is associated with 5 answer-options. In total, the dataset contains 12,247 questions. For CommonsenseQA, we evaluate models on the development-set only, since test-set answers are not publicly available.

Training Details

For ease of comparison, we borrow hyperparameter settings from Pan et al. [214]; we used the BERT Whole-Word Masking Uncased model [74] for all experiments. For DREAM experiments, we used a max sequence-length of 512, batch-size of 24, learning rate of $1e^{-5}$, and we trained the model for 16 epochs. For CommonsenseQA, we used a max sequence length of 60, batch-size of 32, learning rate of $1e^{-5}$, and trained for 8 epochs. For pre-training on OMCS, we used max sequence length of 35, batch-size of 32, learning rate of $3e^{-5}$, and trained for 3 epochs. For pre-training on ATOMIC, the max sequence length is changed to 45, other hyperparameters remain the same, and we only use the ATOMIC training set. When using OCN on CommonsenseQA, since there is no dialogue, we compute co-attention with Q_{enc}, in place of D_{enc}, in order to keep the model structure consistent.

Results

DREAM results are shown in Table 4.5, and CommonsenseQA results are shown in Table 4.6. For all of our experiments, we run 3 trials with different random seeds and we report average scores in the tables. Evaluated on DREAM, our OCN model got a significant performance boost (+3.0%), compared to BERT-large from previous work. We think the reasons are that OCN is better-suited for the task and that we used BERT Whole-Word Masking Uncased model. OCN with ConceptNet knowledge-injection achieves slightly better results on the development-set, while ATOMIC knowledge-injection helps achieve a small improvement on the test-set. However, we recognise that these improvements are very limited; to our surprise, OCN pre-trained on OMCS or ATOMIC got significantly lower performance. As for results on CommonsenseQA, ConceptNet knowledge-injection provides a significant performance boost (+2.8%), compared to the OCN baseline, suggesting that explicit links from question to answer-options help the model find the correct answer. Pre-training on OMCS also provides a small performance boost to the OCN baseline. Since both ConceptNet knowledge-injection and OMCS pre-training are helpful, we combine both approaches with OCN and we are able to achieve further improvement (+4.9%). Finally, similar to the results on DREAM, OCN pre-trained on ATOMIC yields a significant performance drop.

Table 4.8: Accuracies for each CommonsenseQA question type: **AtLoc.** means *AtLocation*, **Cau.** means Causes, **Cap.** means *CapableOf*, **Ant.** means *Antonym*, **H.Pre.** means *HasPrerequiste*, **H.Sub** means *HasSubevent*, **C.Des.** means *CausesDesire*, and **Des.** means *Desires*. Numbers beside types denote the number of questions of that type.

Models	AtLoc.(596)	Cau.(194)	Cap.(109)	Ant.(92)	H.Pre.(46)	H.Sub.(39)	C.Des.(28)	Des.(27)
OCN	64.9	66.5	65.1	55.4	69.6	64.1	57.1	66.7
+CN inj,	67.4(+2.5)	70.6(+4.1)	66.1(+1.0)	60.9(+5.5)	73.9(+4.3)	66.7(+2.6)	64.3(+7.2)	77.8(+11.1)
+OMCS	68.8(+3.9)	63.9(-2.6)	62.4(-2.7)	60.9(+5.5)	71.7(+2.1)	59.0(-5.1)	64.3(+7.2)	74.1(+7.4)
+ATOMIC	62.8(-2.1)	66.0(**-0.5**)	60.6(-4.5)	52.2(-3.2)	63.0(-6.6)	56.4(-7.7)	60.7(**+3.6**)	74.1(**+7.4**)
+OMCS+CN	71.6(+6.7)	71.6(+5.1)	64.2(+0.9)	59.8(+4.4)	69.6(+0.0)	69.2(+5.1)	75.0(+17.9)	70.4(+3.7)

Error Analysis

To better understand when a model performs better or worse with knowledge-integration, we analysed model predictions. DREAM dataset provides annotations for about 1000 questions: 500 questions in the development-set and 500 in the test-set. Specifically, questions are manually classified into 5 categories: Matching, Summary, Logic inference, Commonsense inference, and Arithmetic inference; and each question can be classified under multiple categories. We refer readers to Sun et al. [270] for additional category information. We extracted model predictions for these annotated questions in test-set and grouped them by types. The accuracies for each question-group are shown in Table 4.7. Note that we omitted 2 categories that have less than 10 questions. For the ConceptNet and the ATOMIC knowledge-injection models, we can see that they did better on questions that involve commonsense (last 3 columns in the table), and the performance on other types are about the same or slightly worse, compared to baseline OCN. As for models pre-trained on OMCS corpus or ATOMIC knowledge-base, we already saw that these model performances drop, compared to the baseline. When we look at the performance difference in each question type, it is clear that some categories account for the performance drop more than others. For example, for both the OMCS pre-trained model and the ATOMIC pre-trained model, performance drops significantly for Matching questions, in particular. On the other hand, for questions that require both commonsense inference and summarization, both models' performances only dropped slightly or did not change. Based on these results, we infer that commonsense knowledge-injection with attention is making an impact on models' weight distributions. The model is able to do better on questions that require commonsense but is losing performance on other types, suggesting a direction for future research in developing more robust (e.g., conditional) injection methods. Moreover, pre-training on knowledge-bases seems to have a larger impact on models' weight distributions, resulting in inferior performance. This weight distribution shift also favors of commonsense, as we see that commonsense types are not affected as much as other types.

We also conducted similar analysis for CommonsenseQA. Since all questions in CommonsenseQA require commonsense reasoning, we classify questions based on the ConceptNet relation between the question concept and correct answer concept. The intuition is that the model needs to capture this relation in order to answer the question. The accuracies for each question type are shown in Table 4.8. Note that we have omitted question types that have less than 25 questions. We can see that with ConceptNet relation-injection, all question types got performance boosts, for both OCN model and OCN pre-trained on OMCS, suggesting that knowledge is indeed helpful for the task. In the case of OCN pre-trained on ATOMIC, although the overall performance is much lower than OCN baseline, it is interesting to see that performance for the "Causes" type is not significantly affected. Moreover, performance for "CausesDesire" and "Desires" types actually got much better. As noted by [250], "Causes" in ConceptNet is similar

to "Effects" and "Reactions" in ATOMIC; and "CausesDesire" in ConceptNet is similar to "Wants" in ATOMIC. This result also correlates with our findings from our analysis on DREAM, wherein we found that models with knowledge pre-training perform better on questions that fit knowledge domain but perform worse on others. In this case, pre-training on ATOMIC helps the model do better on questions that are similar to ATOMIC relations, even though overall performance is inferior. Finally, we noticed that questions of type "Antonym" appear to be the hardest ones. Many questions that fall into this category contain negations, and we hypothesize that the models still lack the ability to reason over negation sentences, suggesting another direction for future improvement.

4.2.7 Related Work

It has been recognised that many recent QA tasks require external knowledge or commonsense to solve, and numerous efforts have been made in injecting commonsense in neural models. Bauer et al. [22] introduced a pipeline for extracting grounded multi-hop commonsense relation paths from ConceptNet and proposed to inject commonsense knowledge into neural models' intermediate representations, using attention. Similarly, Mihaylov and Frank [188] also proposed to extract relevant knowledge triples from ConceptNet and use Key-Value Retrieval [190] to gather information from knowledge to enhance the neural representation. Zhong et al. [332] proposed to pre-train a scoring function using knowledge triples from ConceptNet, to model the direct and indirect relation between concepts. This scoring function was then fused with QA models to make the final prediction. Pan et al. [213] introduced an entity discovery and linking system to identify the most salient entities in the question and answer-options. Wikipedia abstracts of these entities are then extracted and appended to the reference documents to provide additional information. Weissenborn et al. [303] proposed a strategy of dynamically refining word embeddings by reading input text as well as external knowledge, such as ConceptNet and Wikipedia abstracts. More recently, Lin et al. [168] proposed to extract sub-graphs from ConceptNet and embed the knowledge using Graph Convolutional Networks [142]. Then the knowledge representation is integrated with word representation through an LSTM layer and hierarchical attention mechnism. Lv et al. [179] introduced graph-based reasoning modules that takes both ConceptNet knowledge triples and Wikipedia text as inputs to refine word representations from a pretrained language model and make predictions.

Commonsense knowledge integration has also received a lot of attention on many other tasks. Tandon et al. [279] proposed to use commonsense knowledge as hard/soft constraints to bias the neural model's prediction on a procedural text comprehension task. Ma et al. [185] proposed to used embedded affective commonsense knowledge inside LSTM cell to control the information flow in each gate for sentiment analysis task. Li and Srikumar [161] presented a framework to convert declarative knowlegde into first-order logic that enhance neural networks' training and prediction. Peters et al. [222] and Levine et al. [160] both tried to injecting knowlegde into language models by pretraining on knowledge bases.

Previous works only focus on using external knowledge sources to improve model performance on certain tasks, disregarding the type of commonsense knowledge and how the domain of the knowledge resource affects results on downstream tasks. In this paper, we examine the roles of knowledge-base domain and specific integration mechanisms on model performance.

4.3 Knowledge-driven Data Construction for Zero-shot Evaluation in CSQA

Common sense is key to efficient communication in everyday situations, as it enables natural language understanding through contextual reasoning. As McCarthy argued in his seminal work [186], an artificial system has commonsense, *if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows*: it follows that machine commonsense can be assessed only by eliciting whether commonsense knowledge and knowledge-based inferences are used by an artificial system in executing a given task. Machine question answering (QA) benchmarks, like `SocialIQA` [251] and `PhysicalIQA` [27], are effective *behavioural* tests of commonsense reasoning in machines, each focusing on different capabilities. Answering a question in `SocialIQA` might require the knowledge that readers typically prefer heroes over villains in fantasy novels; whereas, in `PhysicalIQA`, the knowledge that metal stools can break windows, because windows are made of glass and metal is a more enduring material than glass. Although such tasks had been traditionally difficult for machines, recent developments in pre-trained neural language modelling have led to leaps in accuracy—closing the gap between human and machine performance to single-digit percentage points.[1] However, due to increasing concern that large-capacity neural systems are modelling individual datasets, rather than learning how to perform logical reasoning or to utilise external knowledge effectively [196], focus is shifting to alternative training and evaluation strategies. In particular, *zero-shot evaluation* shows promise as an efficient measure of model generalisability across tasks [164, 260]. Here, models are trained and validated on task $\mathbf{A}$, and tested on a different task $\mathbf{B}$, without access to $\mathbf{B}$'s training data or labels. This leads state-of-the-art models from individual tasks to falter, sometimes by as much as a 50% decrease in performance [260].

Repositories of commonsense knowledge, like `ConceptNet` [264] and `ATOMIC` [250], can be beneficial for commonsense QA, especially when little or no training data is available. Enriching the training data with `ConceptNet` and `ATOMIC` has been shown [182, 196] to improve accuracy on datasets *derived* from these graphs: `CommonSenseQA` [275] and `SocialIQA`. Knowledge bases (KBs) can be used to generate question-answer pairs and distractors automatically, in order to test a model's reasoning ability [223, 240] or provide additional supervision [317, 319]. While KBs have been shown to help in a zero-shot transfer setting recently [17], no comprehensive study exists on the relation between various knowledge, its usage method, and neural models for zero-shot transfer across commonsense tasks. Moreover, while adversarial filtering techniques [34] improve the quality of a manually created question set, their impact on automatically generated questions from a variety of KBs has not been investigated yet.

In this section, (1) we compile a set of hypotheses and design a novel neuro-symbolic framework that investigates the dependency between knowledge sources, question generation techniques, language model (LM) variants, and tasks. Our framework leverages a wide range of KBs, covering visual, social, and concept-based knowledge, to pre-train LMs for zero-shot evaluation on multiple-choice commonsense QA tasks. (2) Recognising that the aspect of question generation is especially understudied, we expand on prior work to devise and test four distractor-sampling strategies for effective question generation. We analyse their impact on model performance across tasks, conditioned on model class and (pre-)training regime, and show that generating questions that are simultaneously fair and informative is difficult but

[1] For example (accessed 4 August, 2020): `https://leaderboard.allenai.org/socialiqa/submissions/public`

beneficial for LM pre-training. (3) We determine which combination of knowledge graphs (KGs), data construction/training, and architectures is most effective and can utilise appropriately rich contexts across five tasks. We observe that diversifying knowledge generally improves performance, under the condition of it being aligned with the task, and that preserving the structure of the task is desired.

4.3.1 Problem Formulation

Given a natural language question Q, and n possible answers $A_1, ..., A_n$, the task is to select the most probable single answer A. We refer to the remaining $n-1$ possible answers: $D_1, ..., D_{n-1}$ as distractors. In a zero-shot QA evaluation mode, the system has no access to the task training or development data. We assume a setup where the system is pre-trained once and then applied across different tasks in a zero-shot manner. Our zero-shot evaluation framework addresses this task by variants of pre-training an LM on an artificial QA set, created from KG data. Next, we describe its covered tasks, sources of knowledge, question generation strategies, LM techniques, and training regimes, in turn.

4.3.2 Synthetic QA Set Generation

We generate questions, answers, and distractor options from five KGs, found in the unified `Commonsense Knowledge Graph` (`CSKG`) [127]: `ATOMIC`, `ConceptNet`, `WordNet`, `VisualGenome` [152], and `Wikidata` [293]. Notably, `ATOMIC` differs from the other KGs in two ways: 1) its relations have a different focus than those of the other sources; and 2) its node labels are longer and are formalised as templates. Due to these considerations, we prepare two sets of QA sets: one based on `ATOMIC` and one based on the remaining four knowledge sources. Figure 4.3 illustrates our question generation pipeline.

Data Partitions

`ATOMIC` expresses pre- and post-states for events and their participants with nine relations. Its head nodes are events, whereas the tail nodes are either events or attributes. Its nodes have two particularities: 1) irrelevant parts of the node text are replaced with blanks ('_'); and 2) references to fictional agents are indicated with special tokens (e.g., *PersonX*). We follow the `SocialIQA`'s `ATOMIC` train/dev/test splits, to ensure that the facts of the dev and test partitions are excluded in training. Our second partition, `CWWV`, covers three other KGs in `CSKG` that express commonsense facts between concepts: `ConceptNet`, `WordNet`, and `Wikidata`. We use them jointly to generate questions, and we enrich them with additional distractors from `VisualGenome`. Treating these four sources as a single one is enabled by their `CSKG` mapping to a single set of relations, defined by `ConceptNet`. We focus on 14 semantic relations that are grounded on strong psycho-linguistic and pragmatic evidence [204], like `/r/Causes` and `/r/HasPrerequisite`. Since there is no pre-defined train/dev/test split for `CSKG`, we randomly sample 5% of generated questions as development set, while the other 95% are used for training, to maximise the coverage of the knowledge.

Generating Questions and Answers

If a triple (h, r, t) has an associated sentence, we directly employ it for question generation; otherwise, we generate a sentence in a lexicalization step, using a set of pre-defined templates. Next, we generate the question Q by removing the tail of the sentence, and extract this tail as the correct answer, A. Here, we ensure that there is no token overlap between the head and the correct answer. For `ATOMIC`, we: 1) compare the keyword tokens instead of all tokens, in order to avoid stop-words; and 2) the agent templates

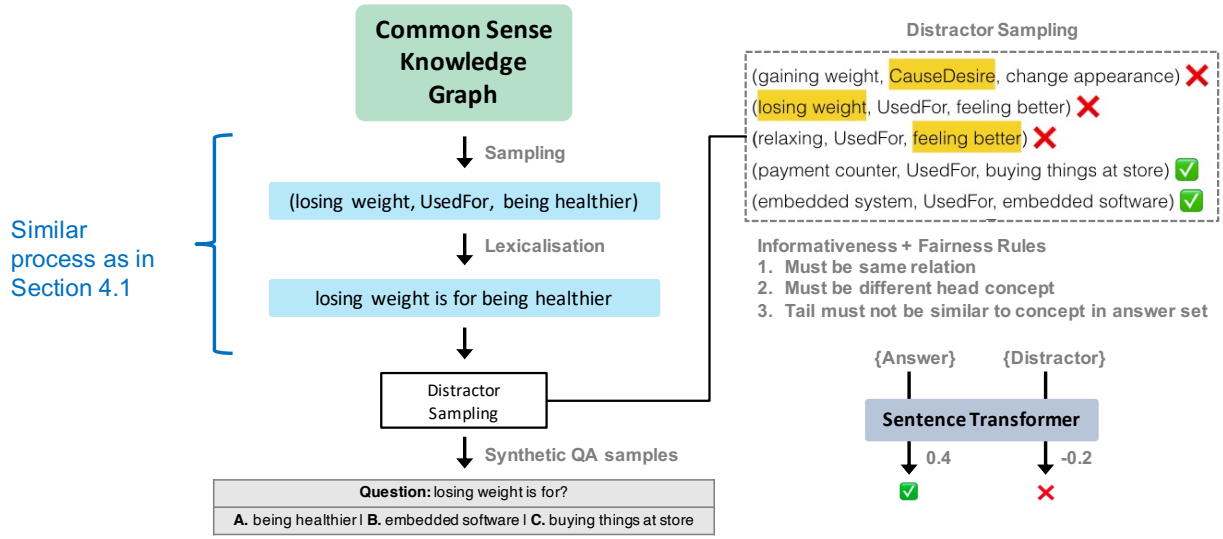


Figure 4.3: An illustration of our question generation pipeline. The first distractor candidate was rejected, as we require distractors to share the same relation as the sample it is predicated on; the second distractor candidate was rejected both because its head overlaps with that of the sample and because its tail is part of the correct answer set. Finally, the third distractor sample was rejected, because its tail is part of the correct answer set.

(e.g., 'PersonX') are replaced with randomly sampled gender-neutral names from a pre-defined set. For CWWV, we filter out questions where either the head or the tail are not common concepts or they are named entities. We use corpus frequency as a proxy for commonness,[2] while named entities are filtered by removing all concepts whose labels start with a capital letter.

Generating Negative Samples: Distractors

We seek to generate distractor options that satisfy two criteria: *informativeness* and *fairness*. Namely, a good distractor has semantic relatedness with the context (informative), while being relatively easy to discriminate from the correct answer (fair). We create the pool of distractors D for every sample as follows: (i) The distractor candidates are the tails of knowledge triples (h', r', t') with the same relation $r' = r$, randomly sampled from the KGs. This would ensure that the distractors can fill the same semantic role as the correct answer. (ii) The head h' of the sampled triples does not have non-stop word overlap with h. (iii) The distractor tail t' is not part of the correct answer set, i.e., there exist no triples, (h, r, t') Considering the example in Figure 4.3, the triple *(gaining weight, CausesDesire, change appearance)* will be filtered out by rule (1), *(losing weight, UsedFor, feeling better)* will be ruled out by both (2) and (3), and *(relaxing, UsedFor, feeling better)* will be ruled out by (3). Here, we replace any references to fictional ATOMIC agents in the distractors with the same names used in the question. We then randomly select two distractors (D_1, D_2) from D. We refer to this distractor pooling strategy as *random*, and propose three alternative strategies in the Sub-section 4.3.3. Example questions with each partition are shown in Table 4.9. For ATOMIC, this procedure generates 535K QA pairs for training and 60K for development. For CWWV, the training set contains 157K and the development set has 8K QA pairs.

[2] https://pypi.org/project/wordfreq/ (Accessed 9 Sept. 2020)

Table 4.9: Generated questions from ATOMIC (top) and CWWV (bottom). (*) denotes the correct answer.

Question: Robin takes the fifth. As a result, Robin wanted to
A1: go to the cinema.
A2: withhold information. (*)
A3: hear what they think.
Question: losing weight is for
A1: being healthier. (*)
A2: embedded software.
A3: buying things in store.

4.3.3 Distractor Sampling

Existing data generation procedures are likely to introduce annotation artefacts in datasets [249, 325]. Models may exploit these artefacts to achieve spuriously strong performance during training, at the expense of degradation in robustness. To generate more challenging QA pairs from KGs and to alleviate potential biases in our synthetic sets, we test two other distractor sampling strategies in addition to the *random* strategy: 1) we select distractors that are as similar as possible to the answer, while being under a certain threshold (*adv-answer*); and 2) we select distractors that are as similar as possible to the question, while being under a certain threshold (*adv-question*). Here we define similarity of two nodes to be their proximity in the embedding space, measured by cosine similarity. The intuition is that, by generating more challenging QA pairs for the models, we could achieve better generalisation across tasks. We use the RoBERTa sentence embedding model [234] to compute embeddings for all KG nodes. For these two strategies, we set an upper bound on the similarity score to avoid unfair distractors, i.e., paraphrases of the correct answer. Based on manual observations, we set their distractor similarity upper bound to be 0.6 for CWWV and 0.4 for ATOMIC.

Sample Filtering

Besides these distractor sampling strategies, we test another condition (3), where we select the distractors randomly, but only keep the questions whose distractors are sufficiently challenging at training time (*adv-filter*). The intuition is that QA pairs generated using the aforementioned methods might still be too easy for the models, thus we would like to only keep the most challenging subset to train our models. We employ the AFLite algorithm [249] for our purpose. Given a *train* and *dev* split of our synthetic QA set, we use 5% of the *train* set to finetune a RoBERTa model with a classification head (4% training, 1% validation). These 5% are discarded from *train* after this step. We then compute the fixed embeddings for the remaining 95% of *train* and the entire *dev*, denoted as *Trn* and *Dev*. Next, we feed *Trn* and *Dev* along with their labels to the AFLite algorithm, which iteratively filters out easy examples using an ensemble of linear classifiers. Finally, we retain (101K training, 11K dev) samples for ATOMIC and (29K training, 1.5K dev) samples for CWWV subset.

4.3.4 Language Models

We consider 2 types of language models: auto-regressive language models and masked language models (MLM). Specifically, we use GPT-2 and RoBERTa to select the best answer candidate. Given a context C,

a question Q, and a list of answer options ($A_1, A_2...$), we concatenate C and Q with each answer option to build input sequences ($T_1, T_2...$). We also use templates to convert a sequence T into a natural language sentence following [260]. For example, we transform the sequence: *[C] What will X want to do next? [A_i]* into: *[C], as a result, X want to [A_i]*. The score S for the resulting sequence using an auto-regressive LM is computed as follows:

$$\mathrm{S}_{\mathrm{LM}}(T) = -\frac{1}{n}\sum_{i=1}^{n}\log P\left(t_i \mid t_1 \ldots t_{i-1}\right) \tag{4.24}$$

where n is the number of tokens in the sequence and P is the conditional probability provided by the LM. To evaluate MLMs, we mask out one token at a time and compute its loss [334]. We repeat this process for every token in the sequence. The final MLM score is:

$$\mathrm{S}_{\mathrm{MLM}}(T) = -\frac{1}{n}\sum_{i=1}^{n}\log P\left(t_i \mid \ldots t_{i-1}, t_{t+1} \ldots\right) \tag{4.25}$$

The predicted option is the one with the lowest score.

Language Model Fine-tuning

In the typical model architecture for fine-tuning LM for multiple-choice tasks, a linear layer is added on top of the LM encoder to predict the answer. The model inputs are separated by a model-specific delimiter. However, as this architecture introduces randomly initialised parameters, it may not be able to fully utilise the pre-trained weights [277]. Instead, we re-use the GPT-2 and RoBERTa with LM head for fine-tuning. By keeping the model intact, we can reuse the same converting templates and scoring functions. To train the model, given the scores computed for each answer candidate $S_1, S_2, ...S_m$, we use the marginal ranking (MR) loss defined as:

$$\mathcal{L} = \frac{1}{m}\sum_{\substack{i=1 \\ i\neq y}}^{m}\max\left(0, \eta - S_y + S_i\right) \tag{4.26}$$

Here, η represents the margin and y is the index of the correct answer. For a MLM model, the computation cost for the scoring function scales in quadratic complexity with the input length. To make the training more efficient, we only mask out non-stop tokens in the head and tail nodes.

Training Regimes

In order to disentangle the contribution of the KGs from the structure of the QA pairs, we consider different training methods for augmentation of language models with KGs. Specifically, we compare marginal ranking (MR) training with masked language modelling (MLM) training. For MLM, we directly concatenate the question and the correct answer in our synthetic QA set and then train RoBERTa on the these sentences using the MLM objective.

4.3.5 Experiments

Reasoning Tasks

We select commonsense tasks based on two criteria. Firstly, we strive to cover a diverse set of tasks, both in terms of their format (question answering, pronoun resolution, natural language inference), as

Table 4.10: Zero-shot evaluation results with different combinations of models and knowledge sources, across five commonsense tasks. CSKG represent the combination of ATOMIC and CWWV. We run our experiments three times with different seeds and report average accuracy with 95% confidence interval. SMLM (*) used OMCS for CSQA, ROCStories [199] for aNLI and ATOMIC for SIQA as knowledge resources.

Model		**KG**	**aNLI**	**CSQA**	**PIQA**	**SIQA**	**WG**
Majority		-	50.8	20.9	50.5	33.6	50.4
GPT2-L		-	56.5	41.4	68.9	44.6	53.2
RoBERTa-L		-	65.5	45.0	67.6	47.3	57.5
Self-talk	[260]	-	-	32.4	70.2	46.2	54.7
COMET-DynaGen	[32]	ATOMIC	-	-	-	50.1	-
SMLM	[17]	*	65.3	38.8	-	48.5	-
GPT2-L (MR)		ATOMIC	59.2(±0.3)	48.0(±0.9)	67.5(±0.7)	53.5(±0.4)	54.7(±0.6)
GPT2-L (MR)		CWWV	58.3(±0.4)	46.2(±1.0)	68.6(±0.7)	48.0(±0.7)	52.8(±0.9)
GPT2-L (MR)		CSKG	59.0(±0.5)	48.6(±1.0)	68.6(±0.9)	53.3(±0.5)	54.1(±0.5)
RoBERTa-L (MR)		ATOMIC	**70.8(±1.2)**	64.2(±0.7)	72.1(±0.5)	63.1(±1.5)	59.6(±0.3)
RoBERTa-L (MR)		CWWV	70.0(±0.3)	**67.9(±0.8)**	72.0(±0.7)	54.8(±1.2)	59.4(±0.5)
RoBERTa-L (MR)		CSKG	70.5(±0.2)	67.4(±0.8)	**72.4(±0.4)**	**63.2(±0.7)**	**60.9(±0.8)**
RoBERTa-L (supervised)		-	85.6	78.5	79.2	76.6	79.3
Human		-	91.4	88.9	94.9	86.9	94.1

well as their type of knowledge (e.g., social or physical knowledge). Secondly, we prefer larger task datasets that are manually constructed. For this reason, we do not include datasets like COPA [104], or HellaSwag [325]. We opt for the following five task datasets:

1. Abductive NLI (aNLI) [25] is posed as a natural language inference task. Given the beginning and the ending of a story, the task is to choose the more plausible hypothesis out of two options. The dataset consists of nearly 170k entries.

2. CommonsenseQA (CSQA) [275] evaluates a broad range of common sense aspects. Each entry contains a question and 5 answer candidates. The questions are crowd-sourced based on a subgraph from ConceptNet. The answer candidates combine ConceptNet nodes with additional crowd-sourced distractors.

3. PhysicalIQA (PIQA) [27] is a two-choice question answering dataset which focuses on physical reasoning. Given a question, the system (or human) is asked to pick the more plausible out of two possible continuations.

4. SocialIQA (SIQA) [251] is a question-answering dataset which requires reasoning about social interactions. Each entry contains a context, a question, and 3 answer candidates. The context is derived from the ATOMIC knowledge graph, the questions are generated based on nine templates (corresponding to the relations in ATOMIC), and the answers are crowd-sourced.

5. WinoGrande (WG) [249] contains 44 thousand pronoun resolution problems. Each entry consists of a context description with an emphasized pronoun, and two options are offered as its possible references.

Baselines

We compare our results with the following baselines. *Majority* answers each question with the most frequent option in the entire dataset. *'Vanilla' versions of the language models* are used in order to understand the impact of further tuning. Here we directly uses the LMs to score the QA pairs without any fine-tuning. We also show the results of other unsupervised systems that leverage KGs: *Self-talk*, *COMET-DynaGen*, and *SMLM*. To indicate the upper bound of this work, we include results of a supervised fine-tuned RoBERTa system and of human evaluation. For the LM baselines, we directly load the weights from the Transformers library [306] and evaluate on the downstream tasks. The fine-tuned LMs are trained for a single epoch on our synthetic QA set. For Adv-filter, we train the models for 5 epochs to compensate for less training data. We use our synthetic development set to select the best model.

Hypotheses

Based on individual prior findings and understanding of different components of our framework, we put forward a set of hypotheses which will be validated in our experiments:

H1 *RoBERTa would have better performance than GPT-2.* This is in line with prior findings that RoBERTa has the advantage of bi-directional context [334].

H2 *Pre-training a language model with artificially created question-answer sets enhances zero-shot performance.* This is also supported in previous study about unsupervised QA [164]

H3 *The impact of more knowledge depends on the alignment between KGs and the task*, partial evidence for which is provided by [182, 196].

H4 *Adding diverse knowledge (from different KGs) improves performance*. This is the initial motivation behind the creation of `CSKG` [127], but has not been investigated in detail.

H5 *When selecting negative samples for a question, it helps to use an adversarial strategy that ensures the question is not trivial for a language model.* H5 is inspired by adversarial filtering, which has not been investigated in detail for automatically-generated questions and across KGs.

H6 *Preserving the task structure when generating synthetic data leads to better accuracy.* This is implicitly assumed in prior data augmentation work [143].

H7 *The automatically created questions are notably easier for humans than they are for machines* - a general assumption made by commonsense task creators and typically correct for any existing, human-generated benchmark.

Results

We evaluate various combinations of: knowledge sources, question generation strategies, LMs, training regimes, and tasks. We use accuracy as a metric. All our experiments are performed in a zero-shot setting, i.e., the models do not leverage the official training data of the task. We report results on the dev sets of these tasks, as the official test sets are not publicly available. We note that, since we did not use the tasks' dev sets for hyperparameter tuning or checkpoint selection, the dev sets can be used effectively as test sets. Table 4.10 shows that GPT-2 and RoBERTa outperform the majority baseline by a large margin on all tasks, indicating that the LMs have already learned relevant knowledge during pre-training. Despite being

Table 4.11: Comparison of different QA generation strategies.

RoBERTa-L	Strategy	aNLI	CSQA	PIQA	SIQA	WG
+ATOMIC	Random	**70.8**(±**1.2**)	**64.2**(±**0.7**)	72.1(±0.5)	**63.1**(±**1.5**)	59.6(±0.3)
+ATOMIC	Adv-answer	70.4(±0.8)	62.3(±0.9)	**72.6**(±**1.8**)	61.6(±0.3)	60.5(±0.5)
+ATOMIC	Adv-question	70.8(±0.6)	55.6(±0.9)	70.6(±0.8)	51.6(±0.8)	58.5(±0.3)
+ATOMIC	Adv-filter	68.6(±1.8)	46.4(±1.5)	67.9(±1.1)	51.8(±1.2)	**60.8**(±**0.6**)
+CWWV	Random	**70.0**(±**0.3**)	67.9(±0.8)	72.0(±0.7)	**54.8**(±**1.2**)	59.4(±0.5)
+CWWV	Adv-answer	69.5(±1.1)	**68.5**(±**0.8**)	**72.7**(±**0.3**)	53.8(±0.6)	**60.7**(±**0.7**)
+CWWV	Adv-question	68.3(±2.3)	60.9(±2.3)	69.6(±0.6)	47.0(±2.0)	59.0(±1.4)
+CWWV	Adv-filter	69.7(±0.7)	64.7(±2.3)	72.0(±1.3)	50.1(±1.0)	59.4(±1.4)

a smaller model, RoBERTa outperforms GPT-2 on 4 out of 5 tasks without pre-training, and on all tasks when pre-training over different synthetic QA sets. This shows the advantage of leveraging bi-directional context, and confirms our hypothesis H1. As expected (H2), training RoBERTa on our ATOMIC or CWWV synthetic sets brings notable performance gain on all 5 tasks. We observe that models trained on ATOMIC sets have a large advantage on SIQA compare to models trained on CWWV, while CWWV brings advantage on the CSQA task. This is not surprising as these two tasks are derived from ConceptNet and ATOMIC, respectively. The difference between ATOMIC and CWWV on the remaining three tasks is relatively small. This supports our hypothesis H3: knowledge alignment is crucial for obtaining better performance.

Training on the combined question set (CSKG) is mostly able to retain the best of its both partitions. Training on CSKG leads to best performance on three out of five tasks, showing that a global commonsense resource is able to bring consistent gain across different tasks. This supports our hypothesis H4: adding more diverse knowledge is beneficial for language models. Finally, even with this knowledge, we recognize that there is still a large gap between our model's accuracy and that of the supervised RoBERTa model.

Comparison of QA Generation Strategies. Table 4.11 shows the results with different sampling strategies, thus addressing H5. The best performing adversarial algorithm, *Adv-answer*, yields comparable accuracy to the *random* strategy, revealing that distractors sampled with a more sophisticated strategy are not necessarily more informative for the LMs. *Adv-question* and *Adv-filter* typically lead to declines in accuracy. Considering *Adv-question*, this could be due to the similarity of the distractors to the question, which might guide the model to learn to pick the most dissimilar candidate as the correct answer, which is an artifact of our question generation and cannot be expected to work well for downstream tasks. Our manual inspection of the remaining questions prefered by *Adv-filter* indicates that many questions are unfair, as some distractors are also correct answers, which is a consequence of the incompleteness of the KGs. Adv-filter prioritizes these questions as they are "difficult" for LMs, however, training on them might teach the LM incorrect knowledge and harm downstream accuracy.

Comparison of Training Regimes. Table 4.12 presents results with two different training regimes. In comparison to the baseline without additional training, MLM training on ATOMIC only improves on the SIQA task, and harms on the rest. With CWWV, it brings large gain on CSQA and small improvements on SIQA and WG. At the same time, marginal ranking training on either question set consistently outperforms MLM training by a large margin, suggesting that preserving the task structure is beneficial in

Table 4.12: Comparison between MLM and MR training.

RoBERTa-L	Train	aNLI	CSQA	PIQA	SIQA	WG
baseline	-	65.5	45.0	67.6	47.3	57.5
+ ATOMIC	MLM	62.9	43.8	65.8	53.9	55.5
+ ATOMIC	MR	70.8	64.2	72.1	63.1	59.6
+ CWWV	MLM	65.3	57.3	67.2	49.3	59.4
+ CWWV	MR	70.0	67.9	72.0	54.8	59.4

Table 4.13: LM and human accuracy on our synthetic QA sets.

Model	ATOMIC	CWWV
GPT2-L	43.2	69.5
RoBERTa-L	45.9	64.5
Human	78.0	80.7

addition to the question content and validating H6.

Difficulty of the Synthetic QA Sets. Ideally, the generated question-answer pairs should be challenging for the models but easy for humans to solve (H7). Here, we probe this hypothesis by assessing the difficulty of our synthetic QA sets both by humans and 'vanilla' LMs. We evaluated both models on the dev sets of our synthetic data. For human evaluation, we randomly sample 50 questions from ATOMIC and 50 questions from CWWV. A total of five researchers were asked to first *provide the correct answer*, then *rate the question difficulty*. For the latter, the annotator chose between easy, moderate, hard, or non-sensical - as a guideline, nonsensical questions have unfair distractors and cannot be easily understood. Following this procedure, we obtained three judgements for each question.

The inter-annotator agreement on selecting the correct answer is 0.62 using Fleiss Kappa score, which is substantial agreement. The Kripendorf alpha [151] for rating question difficulty is 0.35, which is fair agreement. The results of the baseline LMs and human performance (Table 4.13) show that the ATOMIC subset presents a harder challenge for both models, as well as for humans. Overall, the results support our hypothesis H7: the synthetic questions are relatively easy for humans to solve and much harder for models. However, the annotation pointed to several directions for improving the synthetic QA sets. A number of questions generated from ATOMIC are ungrammatical, which makes them harder to understand, while some questions from CWWV were rated as unfair. For example, all answer options for the question *A person can* are valid: (a) *cost several thousand dollars* (b) *expressing boredom* (c) *check snow level*. As discussed earlier, this is due to the incompleteness of our KGs, and the current lack of understanding on how to generate fair, yet informative, distractors.

4.3.6 Related Work

Knowledge injection. Strong performance on standard multiple-choice QA benchmarks, like SocialIQA and PhysicalIQA, has been achieved by fine-tuning a task-specific prediction layer, placed atop pre-trained LMs, such as BERT [74], RoBERTa [175], and GPT [229]. As shown by Ma et al. [182] and Mi-

tra et al. [196], combining neural methods with structured background knowledge from ConceptNet, WordNet [191], and ATOMIC works well for commonsense datasets that have been partially derived from these resources, such as SocialIQA and CommonSenseQA. Here, the structured knowledge, formalised as lexicalised task-targeted evidence paths, is injected into an LM, either via an attention mechanism [22] or through an auxiliary training objective [310]. Graph and relation networks can also be used to score answer candidates, by informing the graph structure with data from LMs [168, 333]. Finally, complete KGs can be incorporated directly in training by introducing additional modeling objectives, to teach a model about general commonsense regardless of the task at hand [160, 174, 222, 276, 329]. This line of work resembles our approach of including background knowledge in a general, task-agnostic way; however, it still relies on the task training data and has generally not been tested in a zero-shot regime.

Generating commonsense questions and answers. Richardson and Sabharwal [240] use links in WordNet to generate question-answer pairs, then leverage the resulting dataset to evaluate language models. Petroni et al. [223] prompt the skills of language models by sentences instead of questions, generated from sources like ConceptNet and SQuAD [230]. Previous works have generated synthetic QA sets to complement existing training data. Ye et al. [319] proposed an 'align-mask-select' method to generate questions using ConceptNet and Wikipedia. Kocijan et al. [143] constructed a large set of pronoun resolution questions using Wikipedia sentences. Yang et al. [317] generate QA pair and distractors using generative models. Unlike our focus on zero-shot evaluation, these efforts use questions to augment the task training data. Assuming low availability of training data, the self-training method by [207] enhances the small set of golden labels with noisy evidence labels, and iteratively uses the pseudo evidence predictions as extra supervision in next iterations. An iterative approach is also used by [164], who use a knowledge graph to generate pre-training data for SQuAD; however, this approach has not been evaluated on commonsense tasks. Regarding zero-shot evaluation, the Self-Talk model of [260] generates clarification prompts based on a template prefix, which are leveraged to elicit knowledge from another LM, which is used jointly with the original context and question to score each answer candidate.

Given a task context, one can use COMET [32], a generative model trained on commonsense KGs, to generate background knowledge statements, and to compute scores for each answer candidate based on the context, question, and generated knowledge. Banerjee and Baral [17] pre-train the LM with three representation learning functions which aim to complete a knowledge triple given two of its elements. These functions jointly compute the distance for each answer candidate. The ambition of this paper is to provide a comprehensive framework for such prior efforts on zero-shot QA with KGs. By covering a wider set of KGs, question generation techniques, and tasks, we can systematically investigate the effect of using different KGs, generation methods, and techniques across tasks.

4.4 Conclusion

In this chapter, we highlighted symbolic commonsense knowledge as a dominant form of domain knowledge, for grounding the predictions of large-capacity neural language models (LMs) in natural language processing tasks, such as non-extractive multiple-choice commonsense question answering. In the first section, we were reminded that these tasks remain challenging, because (i) systems are required to reason about, synthesise, and gather disparate pieces of information, in order to generate responses to queries; and because (ii) recent approaches on such tasks show increased performance, only when models are either pre-trained with additional information or when domain-specific heuristics are used, unless special consideration is given regarding the knowledge resource type to be used. We showed that, by identifying the most appropriate type of commonsense knowledge for a particular problem (e.g., from declarative, taxonomic, relational, procedural, sentiment, and metaphorical common sense), we can consistently improve downstream performance and cross-domain generalisation. We introduced a new model that performs commonsense knowledge grounding+lexicalisation (for extraction) and trilinear attention-based combination with neural context (injection). We evaluate our models on the `DREAM` [270] and `CommonsenseQA` [275] datasets, and we show that: (i) our approach is preferable for knowledge integration and (ii) that the degree of domain overlap, between knowledge-base and task, is vital to model success. In the second section, we address concerns that large-capacity LMs overfit to specific tasks, without learning to utilise external knowledge or perform general semantic reasoning, despite their recent advances and significant gains on in-domain task performance. We identify zero-shot evaluation as a more robust measure of models' general reasoning abilities, as models that achieve state-of-the-art performance on individual datasets suffer significant performance-degradation under zero-shot evaluation on new, but similar tasks. We propose a novel neuro-symbolic framework for zero-shot question answering across commonsense tasks; the framework studies how to transform various pre-existing knowledge resources into a form that is most effective for pre-training models. We vary the set of language models, training regimes, knowledge sources, and data generation strategies, and we measure their impact across tasks; we devise and compare four (4) constrained distractor-sampling strategies, and we provide empirical results across five (5) commonsense question-answering tasks, with data generated from five (5) external knowledge resources. We show that, while an individual knowledge graph is better suited for specific tasks, a global knowledge graph brings consistent gains across different tasks. In addition, both preserving the task structure and generating fair and informative questions are crucial for learning.

Chapter 5

Learning with Primitives

in sequential decision-making tasks

5.1 Motivation

One of the long-term goals in artificial intelligence is that of developing autonomous *embodied* agents that can flexibly interact with their environments. Embodied multimodal robot navigation is an instance of this objective, wherein the agent must flexibly perceive its visual context, understand directives given to it in natural language, and execute the steps necessary for carrying out the instructed task(s). Whereas it may be trivial for a human to understand and follow natural language instructions, such as *Go up the steps. Go into the house ... to the fridge and wait there*, this can be a particularly challenging task for artificial agents. Complexities include partial observability over the state space, label-independent variation in the indoor scenes, complexity in the natural language instructions from varying lengths and lexicons, sparse reward structures over the span of an episode (thereby limiting task progress-monitoring capability), and biased coverage of the provided expert demonstrations. Various approaches pursue solutions to these challenges, through pragmatic inference, back-translation, and progress-monitoring based on distance-to-goal, but still struggle to ground agents' actions in some domain invariant context, such as a navigation graph, which would otherwise encourage the agent to learn stopping and recovery actions and avoid bad states. Furthermore, the aforementioned issues with instruction complexity and label-independent variation in the visual features still persist, reducing agents' generalisation performance in unseen environments.

In Section 5.2, we consider the semantic audio-visual navigation (S-AVN) task proposed by [53]. We introduce the use of knowledge-driven scene priors in the semantic audio-visual embodied navigation task: we combine semantic information from our novel knowledge graph that encodes object-region relations, spatial knowledge from dual Graph Convolutional Networks, and background knowledge from a series of pre-training tasks—all within a reinforcement learning framework for audio-visual navigation. We define a new audio-visual navigation sub-task, where agents are evaluated on novel sounding objects, as opposed to unheard clips of known objects. We show improvements over strong baselines in generalisation to unseen regions and novel sounding objects, within the Habitat-Matterport3D simulation environment, under the SoundSpaces task.

In Section 5.3, we propose a framework for distilling a navigation agent's experience into a representation of reusable maneuvers, where the high-frequency and label-independent variation in the instructions and visual context are removed. Our approach encourages the agent to generalise reusable navigation maneuvers on the basis of similarities across high-level textual instructions, conditioned on past actions, in order to leverage experience in executing familiar sub-commands from new instructions. We achieve this association through a method called auxiliary variable variational approximation, which allows us to introduce a latent variable for estimating the conditional posterior distribution over all observations and executed trajectories (otherwise intractable for most distributions of interest). The agent then learns how to compose samples from this skill space and couple them with conventional multimodal co-grounding mechanisms, through policy refinement, for improved generalisability and downstream performance. We perform an error analysis to illustrate the robustness of models' generalisation to complex scenarios, and we show improvements from our approach.

5.2 Knowledge-driven Scene Priors for Audio-Visual Navigation

5.2.1 Problem Formulation

We consider the semantic audio-visual navigation (S-AVN) task proposed by [53]. In this task, the agent is initialised at a random location, in an unmapped 3D house environment, containing a sounding object (e.g., piano). The agent's task is to reach the sounding object using its sensory inputs, consisting of visual and audio sensors. Two assumptions are made in this task: 1) the target sound has a variable length and may not be available at each time step, so the sound may stop during navigation (e.g., telephone ringing sound stops after some time); 2) the sounding object has a visual embodiment, which is semantically meaningful (e.g., the sound produced by a spoon dropping is associated with the spoon). These assumptions are realistic because sound events have a variable length in the real world based on the semantics of the sounding object. For example, the sound produced by a glass jar breaking would usually be shorter than a telephone ringing sound. Due to the variable length nature of the sound, the agent cannot rely on the audio signal alone to reach the sounding object. Instead, the agent needs to use the audio signal to predict its location and understand the sounding object's semantics. Moreover, the agent also needs to use the visual cues for associating it with the sound semantics and reason about the object and region semantics to navigate effectively.

We further extend the S-AVN task by evaluating the agent on unheard sounding objects. In the initial task [53], the agent was evaluated on unheard clips of the known sounding objects, whereas in our task, the agent is evaluated on completely unknown sounding objects. More formally, let $\mathcal{H}$ be the set of houses, let $\mathcal{O}$ be the set of sounding objects (e.g., shower, tv monitor), and let $\mathcal{R}$ be the set of regions (e.g., bathroom, living room). A house $h_i \in \mathcal{H}$ has a set of regions $\{r_{i1}, r_{i2}, \ldots, r_{ij}\}$ and a set of objects $\{o_{i1}, o_{i2}, \ldots, o_{ik}\}$, where there are k objects placed in j regions of the house h_i. Note that there are multiple instances of each sounding object $o \in \mathcal{O}$ and region $r \in \mathcal{R}$ across all houses $\mathcal{H}$. We divide the total set of possible houses $\mathcal{H}$ into two mutually exclusive subsets: $\mathcal{H}_{seen}$ and $\mathcal{H}_{unseen}$. Similarly, we divide sounding objects $\mathcal{O}$ into two subsets: $\mathcal{O}_{heard}$ and $\mathcal{O}_{unheard}$. The houses in $\mathcal{H}_{seen}$ and the sounding objects in $\mathcal{O}_{heard}$ are only experienced by the agent during the training phase; the agent is evaluated on unheard sounding objects $\mathcal{O}_{unheard}$. Thus, the agent must learn to reason about the novel sounds based on prior knowledge to solve this task. Our work aims to enable the agent to reach the sounding object it has never experienced before.

5.2.2 SAVEN-agent: Semantic Audio-Visual Embodied Navigation

We introduce a knowledge-driven approach for semantic audio-visual embodied navigation (K-SAVEN). K-SAVEN incorporates scene priors in knowledge graph form and extracts relational features using Graph Convolutional Network (GCN) [142] for audio and visual modalities. GCN provides the agent reasoning capability using prior knowledge and dynamically updates its belief according to the current observation, specific to the current environment. Our model also incorporates Scene Memory Transformer (SMT) [84] that captures long-term dependencies by recording visual features in memory and locating the goal by attending to acoustic features. We use visual observations to compute visual features, including vision-based semantic knowledge vector and features encoded from the vision encoder. Similarly, we use audio observations to compute acoustic features, including audio-based semantic knowledge vector and location prediction from location predictor. Thus, the prior knowledge-driven reasoning capability using GCNs with the memory-based attention mechanism using SMT allows the agent to generalise to novel houses and sounding objects, exploit spatio-temporal dependencies, and navigate to the goal efficiently.

Illustrated in Fig. 5.1, the K-SAVEN policy consists of 5 modules: 1) Pre-trained models that, given the audio and visual observations from the environment, predict objects and regions; 2) Graph Convolutional Networks that compute audio-semantic and visual-semantic feature embeddings; 3) Vision Encoder that projects the visual observations at each step to an embedding space; 4) Location Predictor that, given the acoustic signal from the sounding object, predicts its relative distance and direction from the agent; 5) Scene Memory Transformer that uses an attention-based policy network, which computes a distribution over actions, given the encoded observations in scene memory and the acoustic observation that captures goal information from acoustic events. In the following sections, we discuss each module in detail.

Modular Pre-training. In our task, the agent relies on audio observations to set its goal and uses visual observations to navigate to that goal. Therefore, the agent must detect objects and regions in a given observation. To this end, we trained audio (f_c^b) and vision (f_c^v) classification models to predict classification scores for objects and regions in a given observation. More specifically, f_c^b and f_c^v predict a score for each object $o \in \mathcal{O}$ (the likelihood that the object o produced the observation) and region $r \in \mathcal{R}$ (the likelihood that the observation correspond to region r). These models are used as a backbone of the other models in our proposed framework. The acoustic event has variable length and may not be present at each time step, so the agent cannot rely on the current audio observation alone as a persistent signal. Thus, our model aggregates the current prediction $\hat{c}_t^b$ with the previous prediction c_{t-1}^b, $c_t^b = f_\lambda(\hat{c}_t^b, c_{t-1}^b) = (1-\lambda)\hat{c}_t^b + \lambda c_{t-1}^b$, where λ is the weighting factor set to 0.5. When the acoustic event stops (i.e., zero sound intensity), the agent uses its latest estimate c_t^b.

Knowledge Graph Construction. Our knowledge graph captures spatial relationships between object-to-object, object-to-region, and region-to-region. This prior knowledge about how objects are placed in regions of houses enables the agent to reason about where to find novel-sounding objects for efficient navigation; more precisely, this prior knowledge enables the reasoning path, *Sound* $\rightarrow$ *Object* $\rightarrow$ *Region*, which is crucial to the task of audio-conditioned visual navigation. For example, suppose the squeaky sound produced by a chair is novel to the agent, and it knows that chairs are usually kept close to tables or cushions and found in living rooms, or offices. In that case, it may decide to navigate to regions that usually have chairs and objects usually placed close to chairs, which would lead to finding the chair faster than not knowing such spatial and semantic relationships between objects and regions.

Our knowledge graph is denoted by an undirected graph $G = (V, E)$, where V and E denote vertices and edges, respectively. Each vertex denotes an object or region, and each edge denotes the relationship between a pair of vertices. To compute these relationships, we use Matterport3D dataset [43] as it contains

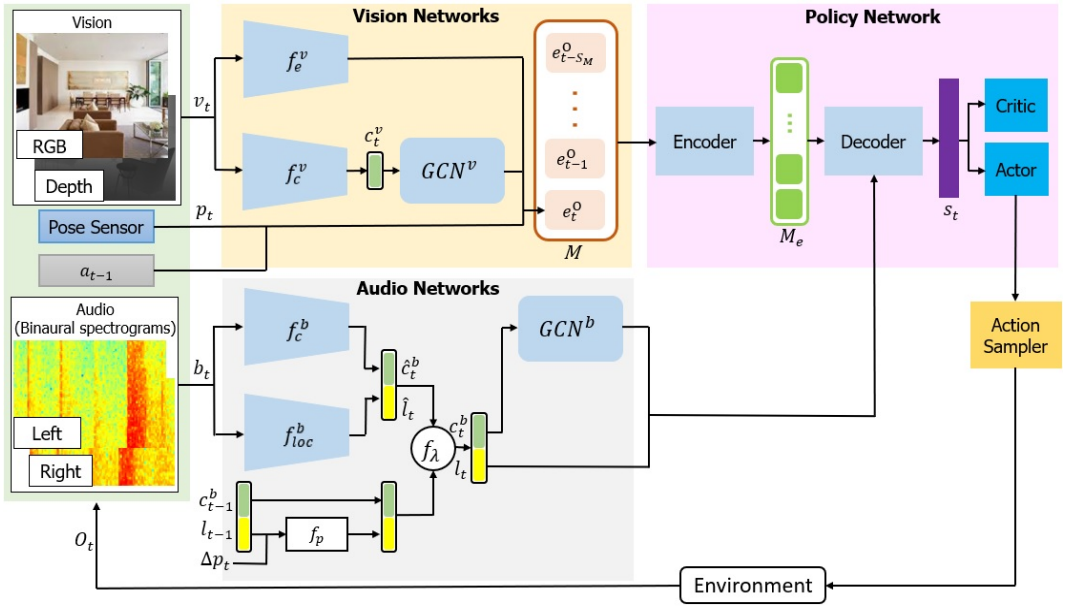


Figure 5.1: K-SAVEN's system overview. Visual observation v_t is fed to two modules: vision encoder f_e^v, which encodes the visual observation, and pre-trained vision model f_c^v, which, given the visual observation, predicts classification scores c_t^v for objects and regions. These scores are used by the vision-based graph convolutional network GCN^v to compute visual-semantic feature embeddings. The outputs of these two models are stored in memory M. Audio observation b_t is also fed to two models: location predictor f_{loc}^b, which predicts distance and direction of the sounding object from the agent (l_t), and pre-trained audio model f_c^b, which, given the audio observation, predicts classification scores c_t^b for objects and regions. These scores are used by the audio-based graph convolutional network GCN^b to compute audio-semantic feature embeddings. The attention-based policy network conditions the encoded visual information M_e on the acoustic information, enabling the agent to associate visual cues with acoustic events and predict the state representation s_t, which contains spatial and semantic cues helpful to reach the goal faster. The actor-critic network, given the state s_t, predicts the next action a_t. When the agent executes the action in the environment, it receives a reward and observations.

semantic labels of 42 objects and 30 regions for 90 houses. We only use 21 objects and 24 regions ($|V| = 45$), which were used in the original S-AVN task [53] to build the knowledge graph. More specifically, two objects (object-to-object) are connected with an edge if they are found in the same region, and their frequency of occurrence is above a threshold. This frequency is computed with respect to the most frequency object of that region, and the threshold is set to the maximum value that connects each object with at least one other object. An object and a region (object-to-region) are connected if the region contains other object(s), which are connected with the object based on object-to-object relations. Finally, two regions (region-to-region) are connected if their frequency of containing connected objects based on object-to-object relations is above a threshold. This threshold is set to the maximum value, connecting each region with at least one other region.

The resultant knowledge graphs are provided in Tables 5.1 and 5.2, which can be represented as adjacency matrices, with an indicator of 1 to characterise a co-occurrence edge between objects, other objects, regions, and other regions. Alternatively, these graphs can be represented in the same format as existing large-scale commonsense knowledge resources, such as ConceptNet [264]: i.e., as a collection of head

Table 5.1: Relational knowledge graph for spatial object-object interactions

Sounding objects (21)	**Objects** (21)	**Regions** (22)
bathtub	towel, sink, shower, picture, cabinet, toilet, counter, table, plant	bathroom
bed	chair, picture, table, sink, seating, cushion, cabinet, chest_of_drawers, shower, plant, counter, tv_monitor, towel	spa/sauna, junk, bedroom
cabinet	clothes, chair, towel, seating, shower, toilet, picture, table, sink, cushion, plant, sofa, counter, bed, chest_of_drawers, bathtub, tv_monitor, stool, fireplace	spa/sauna, bathroom, familyroom/lounge, living room, entryway/foyer/lobby, kitchen, office, utilityroom/toolroom, other room, hallway, laundryroom/mudroom, closet
chair	gym_equipment, picture, seating, cushion, table, plant, cabinet, sink, shower, chest_of_drawers, bed, counter, sofa, towel, tv_monitor, stool, fireplace	spa/sauna, familyroom/lounge, living room, junk, entryway/foyer/lobby, kitchen, office, utilityroom/toolroom, bedroom, other room, rec/game, balcony, lounge, porch/terrace/deck, hallway, dining room, meetingroom/conferenceroom, workout/gym/exercise
chest_of_drawers	chair, picture, cushion, table, bed, tv_monitor, cabinet	office, bedroom
clothes	cabinet, picture	closet
counter	towel, cabinet, shower, chair, toilet, picture, sink, cushion, bed, tv_monitor, table, bathtub, plant, stool	bathroom, junk, kitchen, utilityroom/toolroom, laundryroom/mudroom
cushion	chair, picture, seating, table, sink, plant, cabinet, shower, chest_of_drawers, bed, sofa, counter, towel, tv_monitor, stool, fireplace	spa/sauna, familyroom/lounge, living room, junk, entryway/foyer/lobby, office, utilityroom/toolroom, bedroom, other room, rec/game, balcony, lounge, porch/terrace/deck
fireplace	cushion, table, chair, picture, sofa, plant, stool, cabinet	living room
gym_equipment	picture, chair	workout/gym/exercise
picture	clothes, gym_equipment, toilet, chair, seating, shower, cushion, towel, cabinet, table, sink, chest_of_drawers, bed, counter, plant, sofa, bathtub, tv_monitor, stool, fireplace	spa/sauna, bathroom, familyroom/lounge, living room, junk, entryway/foyer/lobby, kitchen, office, utilityroom/toolroom, bedroom, other room, rec/game, lounge, hallway, laundryroom/mudroom, closet, dining room, meetingroom/conferenceroom, toilet, workout/gym/exercise
plant	chair, picture, sink, towel, table, cushion, shower, toilet, seating, cabinet, sofa, counter, bed, bathtub, tv_monitor, stool, fireplace	spa/sauna, bathroom, familyroom/lounge, living room, junk, entryway/foyer/lobby, rec/game, balcony, porch/terrace/deck
seating	chair, table, sink, picture, plant, cabinet, shower, bed, cushion, towel	spa/sauna, entryway/foyer/lobby, other room
shower	chair, sink, towel, table, toilet, seating, cabinet, picture, counter, bed, plant, bathtub, cushion	spa/sauna, bathroom
sink	cabinet, chair, towel, shower, toilet, seating, picture, table, counter, cushion, bed, tv_monitor, plant, bathtub, stool	spa/sauna, bathroom, junk, kitchen, utilityroom/toolroom, laundryroom/mudroom
sofa	chair, picture, cushion, table, plant, cabinet, stool, tv_monitor, fireplace	familyroom/lounge, living room, rec/game, balcony, lounge, porch/terrace/deck
stool	cushion, chair, picture, table, cabinet, counter, sofa, plant, sink, tv_monitor, fireplace	familyroom/lounge, living room, kitchen
table	chair, towel, picture, seating, shower, toilet, cushion, sink, cabinet, plant, bed, chest_of_drawers, counter, sofa, bathtub, tv_monitor, stool, fireplace	spa/sauna, bathroom, familyroom/lounge, living room, entryway/foyer/lobby, kitchen, office, utilityroom/toolroom, bedroom, other room, rec/game, balcony, lounge, porch/terrace/deck, hallway, dining room, meetingroom/conferenceroom
toilet	sink, shower, towel, cabinet, picture, counter, bathtub, table, plant	bathroom, toilet
towel	toilet, chair, sink, table, shower, seating, cabinet, picture, counter, bed, plant, bathtub, cushion	spa/sauna, bathroom, toilet
tv_monitor	chair, picture, table, cushion, sink, plant, sofa, cabinet, counter, bed, chest_of_drawers, stool	familyroom/lounge, junk, office

h / relation **r** / tail **t** triples of the form (**h**, **r**, **t**), with the ConceptNet `LocatedNear` relation for each

Table 5.2: Relational knowledge graph for spatial region-region interactions

Regions (22)	**Objects** (21)	**Other regions** (22)
balcony	chair, plant, cushion, table, sofa	living room, familyroom/lounge, rec/game, porch/terrace/deck
bathroom	towel, sink, shower, picture, cabinet, toilet, counter, bathtub, table, plant	spa/sauna
bedroom	cushion, picture, chest_of_drawers, bed, chair, table	spa/sauna, office
closet	clothes, cabinet, picture	bathroom, hallway, entryway/foyer/lobby, living room, familyroom/lounge, office, kitchen, laundryroom/mudroom, spa/sauna, other room, utilityroom/toolroom
dining room	chair, picture, table	bedroom, hallway, entryway/foyer/lobby, living room, familyroom/lounge, office, kitchen, lounge, rec/game, spa/sauna, other room, utilityroom/toolroom, meetingroom/conferenceroom
entryway/foyer/lobby	picture, chair, table, plant, cabinet, cushion, seating	spa/sauna
familyroom/lounge	cushion, chair, picture, table, plant, sofa, cabinet, tv_monitor, stool	living room
hallway	picture, cabinet, chair, table	entryway/foyer/lobby, living room, familyroom/lounge, office, kitchen, spa/sauna, other room, utilityroom/toolroom
junk	picture, chair, sink, cushion, counter, plant, bed, tv_monitor	spa/sauna
kitchen	cabinet, chair, counter, sink, stool, picture, table	utilityroom/toolroom
laundryroom/mudroom	cabinet, counter, picture, sink	bathroom, kitchen, utilityroom/toolroom
living room	cushion, table, chair, picture, sofa, plant, stool, fireplace, cabinet	familyroom/lounge
lounge	chair, picture, table, cushion, sofa	living room, familyroom/lounge, rec/game
meetingroom / conferenceroom	chair, picture, table	bedroom, hallway, dining room, entryway/foyer/lobby, living room, familyroom/lounge, office, kitchen, lounge, rec/game, spa/sauna, other room, utilityroom/toolroom
office	chair, table, picture, tv_monitor, chest_of_drawers, cabinet, cushion	familyroom/lounge
other room	seating, chair, table, picture, cushion, cabinet	entryway/foyer/lobby, spa/sauna
porch/terrace/deck	chair, plant, table, cushion, sofa	balcony, living room, familyroom/lounge, rec/game
rec/game	chair, table, cushion, picture, sofa, plant	living room, familyroom/lounge
spa/sauna	table, chair, sink, seating, cabinet, shower, picture, bed, plant, towel, cushion	bathroom, entryway/foyer/lobby
toilet	toilet, picture, towel	bathroom
utilityroom/toolroom	cabinet, chair, picture, table, counter, cushion, sink	kitchen, spa/sauna
workout/gym/exercise	gym_equipment, picture, chair	bedroom, hallway, dining room, entryway/foyer/lobby, living room, familyroom/lounge, office, kitchen, lounge, rec/game, spa/sauna, other room, utilityroom/toolroom, junk, meetingroom/conferenceroom

(h, t)=(object, object) instance pair, the `AtLocation` relation for each (h, t)=(object, region) instance pair, and with the `LocatedNear` relation for each (h, t)=(region, region) instance pair—with saliency weights, based on frequency. Some instances can be further expanded with additional relations, such as `UsedFor`, derived from activity annotations in the region labels. The following triples are taken from the first and tenth rows of table 5.1, for example:

(*bathtub*, `LocatedNear`, *towel*)
(*bathtub*, `LocatedNear`, *sink*)
(*bathtub*, `AtLocation`, *bathroom*)

...
(*gym_equipment*, `UsedFor`, *workout*)
(*gym_equipment*, `AtLocation`, *gym*)
(*gym_equipment*, `UsedFor`, *exercise*)

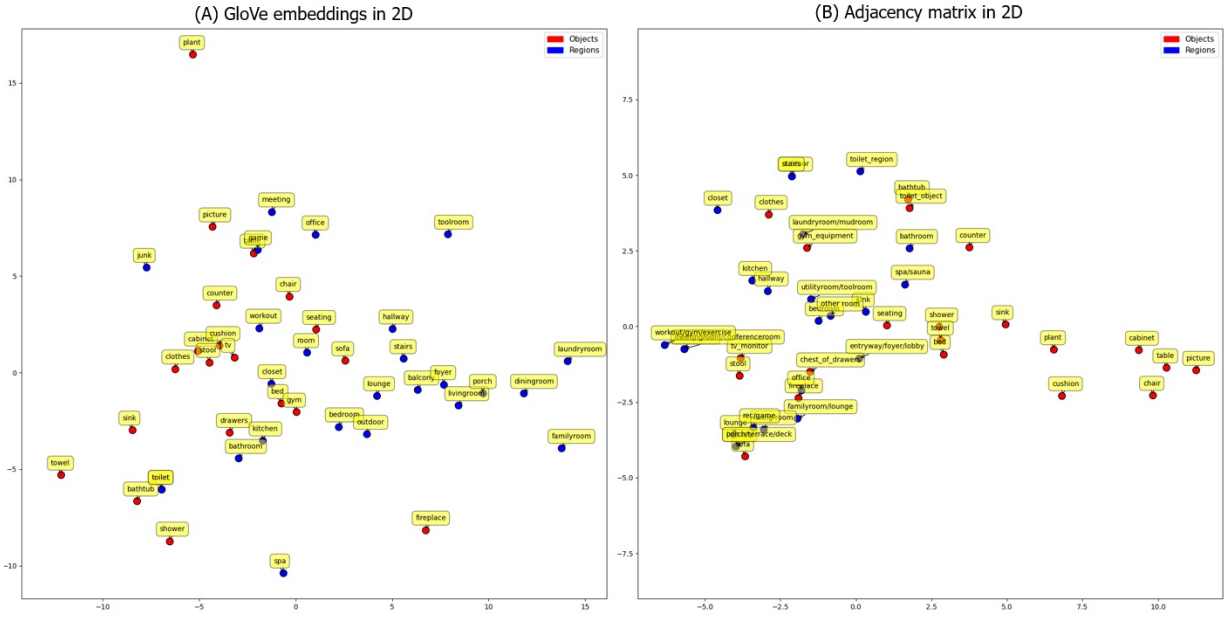


Figure 5.2: Lower-dimensional projections, illustrating object-region similarity. (A) GloVe embeddings for each object and region into 2D space. (B) Adjacency matrix that encodes the relationship between objects and regions in 2D space.

Figure 5.2A illustrates the GloVe embedding space and Figure 5.2B represents the object-region adjacency matrix, both as two-dimensional projections. We reduced the dimension of the GloVe embeddings, for each object and region, into 2 by using ISOMAP [281] (shown in Figure 5.2A). We also reduced the dimension of the vector in the adjacency matrix that encodes the relationship of each object and region with other objects and regions (shown in Figure 5.2B). As shown in Figure 5.2, regions and objects are clustered together, and objects found together in houses, such as tables and chairs, are close together.

Graph Encoder. The goal of our GCNs (GCN^v and GCN^b) is to extract a semantic knowledge vector using the graph $G = (V, E)$. As shown in Fig. 5.3, the input to each vertex v is feature vector x_v, which is a concatenated representation of both semantic cues (i.e., language embeddings) and the visual or acoustic cues (i.e., the classification score for objects and regions based on the current visual image or sound signal). The language embeddings are generated by GloVe [221] (f^l_{enc}) and the classification score is generated by pre-trained vision (f^v_c) or audio (f^b_c) modules. The knowledge graph is represented as a binary adjacency matrix A. Similar to [142], we perform normalisation on A to obtain $\tilde{A}$. Let $X = [x_1, \ldots, x_{|V|}] \in R^{|V| \times D}$ be the inputs of all vertices and $Z = [z_1, \ldots, z_{|V|}] \in R^{|V| \times F}$ be the output of the GCNs, where D and F denote the dimension of the input and output feature.[1] Our GCNs perform the following layer-wise information propagation rule:

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)})$$

[1] D is set to 345 (300 for word embedding and 45 for classification score) and F is set to 254.

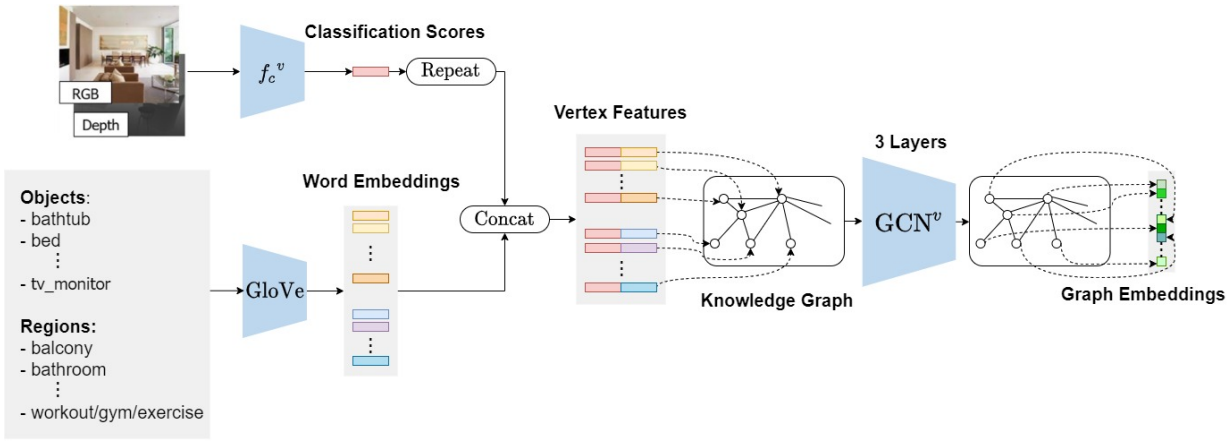


Figure 5.3: Vision-based Graph Convolutional Networks. Each vertex denotes an object or region category. The initial vertex features fed into the GCN^v are initialized with the joint embedding obtained by concatenating word embeddings of object or region names and classification scores of objects and regions based on the current observation. GCN^v performs information propagation through the three layers, and the output of the GCN^v is graph embedding. Note that the audio-based GCN (GCN^b) uses f_c^b and GCN^b instead of f_c^v and GCN^v.

Here, $H^{(0)} = X, H^{(L)} = Z, W^{(l)}$ is the parameter for the l-th layer, L is the number of GCN layers, and σ denote the activation function, which is ReLU in our implementation. We initialise each vertex based on the current observation and then perform information propagation to compute audio-based and vision-based semantic knowledge vectors. The vision-based semantic knowledge vector is stored in memory M, and the audio-based knowledge vector is used to attend to the encoded memory M_e. The output is a graph embedding which serves as a spatial- and semantic-aware representation for policy optimisation.

Vision Encoder and Location Predictor. Our vision encoder f_e^v encodes the visual observations, consisting RGB and depth images from the agent's perspective. We used the pre-trained vision model, described above, as the backbone architecture of vision encoder. The audio observation contains information about the relative distance and direction from the agent to the sounding object. Thus, we trained a location predictor f_{loc}^b to predict a location $\hat{l}_t^b = (\Delta x, \Delta y)$ relative to the current pose p_t of the agent. Similar to the pre-trained audio model, location predictor model also aggregates the current prediction $\hat{l}_t^b$ with the previous prediction l_{t-1}^b, $l_t^b = f_\lambda(\hat{l}_t^b, l_{t-1}^b, \Delta p_t) = (1-\lambda)\hat{l}_t^b + \lambda f_p(l_{t-1}^b, \Delta p_t)$, where $f_p(\cdot)$ transforms the previous location prediction l_{t-1}^b based on the last pose change Δp_t, and λ is the weighting factor set to 0.5. The agent uses its latest estimate $l_t^b = f_p(l_{t-1}^b, \Delta p_t)$ when the acoustic event stops. We used the pre-trained audio model, described above, as the backbone architecture for the location predictor.

Policy Network. We used attention-based transformer architecture for our reinforcement learning policy network, which stores observations in memory M. At each time step, our model encodes each visual observation, $e_t^v = f_e^v(v_t)$ and $e_t^{v-gcn} = GCN^v(f_c^v(v_t))$ to save in the memory. Our model also stores in memory the agent's pose p, defined by its location and orientation (x, y, θ) with respect to its starting pose p_0 in the current episode, and a_{t-1}, the previously executed action. Thus, the encoded observation stored in memory is $e_t^O = [e_t^v, e_t^{v-gcn}, p_t, a_{t-1}]$. The model stores these observation encodings up to time t in memory: $M = \{e_t^O : i = max\{0, t - S_M\}, \ldots, t\}$, where S_M is the memory size.

The transformer uses the memory M stored so far in the episode and encodes these visual observation em-

beddings with a self-attention mechanism to compute the encoded memory $M_e = Encoder(M)$. Then, using the audio observation embeddings, a decoder network attends to all cells in the encoded memory M_e to calculate the state representation $s_t = Decoder(M_e, e_t^{b-gcn}, l_t^b)$, where $e_t^{b-gcn} = GCN^b(f_c^b(b_t))$. Using this attention mechanism, the agent captures long-term spatio-temporal associations between the acoustic-driven goal prediction and the visual observations. Moreover, our model preserves the most relevant information to reach the goal by conditioning visual-semantic embeddings stored in M_e on audio-semantic embeddings computed using current audio observation. The actor-critic network uses s_t to predict the value of the state and action distribution. Finally, the action sampler samples the next action a_t from this action distribution to select the agent's next action.

5.2.3 Learning and Optimisation

To train the vision classification model f_c^v, we collect a dataset using 85 Matterport3D houses, consisting of 82,828 images, each corresponding to a location and rotation angle in the SoundSpaces simulator (see Section 5.2.4). Each image has 128 x 128 resolution and 4 modalities: RGB image, depth image, object semantic image, and region semantic image. We use the binary cross-entropy loss for optimising the vision classification model and train it as a standard multi-label classifier.

To train the audio classification model f_c^b, we use the SoundSpaces simulator to generate 1.5M spectrograms, using different source and receiver positions, each corresponding to a sounding object in one of the 85 Matterport3D houses. One spectrogram corresponds to a sounding object, which could be present in multiple regions. For example, a sink can be present in both kitchen and bathroom regions. Thus, we treat detecting sounding objects as a multi-class classification problem and detecting regions in which that sounding object could be present as a multi-label classification problem. We optimise the audio classification model using cross-entropy loss for sounding object detection and binary cross-entropy loss for detecting regions.

Our vision classification model takes an RGB image as input, and the audio classification model takes 1 second sound clip represented as two 65×26 binaural spectrograms as input. We trained both vision and audio classification models using a ResNet-18 [114] architecture, pre-trained on ImageNet, to predict a score to 21 objects and 24 regions (see Section 5.2.4). These models are pre-trained before and are frozen during during policy training.

For training the location predictor f_{loc}^b to predict a relative location of the sounding object, we use the ResNet-18 architecture and initialise it with the weights of the pre-trained audio classification model. Location predictor is trained during policy training using the same experience collected for policy training. We optimise the location predictor using the mean squared error loss and update it with the same frequency as the policy network.

We train the policy network using the decentralised distributed proximal policy optimisation (DD-PPO) [305], which consists of a value network loss, policy network loss, and an entropy loss to encourage exploration [253]. We adapt the two-stage training procedure proposed in [84] for effectively training the vision networks (f_e^v, GCN^v). In the first stage, the SMT policy is trained without attention by setting the memory size $s_M = 1$ and storing the latest observation embeddings. In the second stage, the memory size is set to $s_M = 150$, and the parameters of the vision networks are frozen. Training SMT requires enormous computational power, and due to limited computational resources, we were not able to complete the second stage of training the SMT policy. Thus, the results for our method and the SAVi baseline correspond to the policy after the 20,000 updates of the first training stage. Moreover, the results for the

rest of the baselines also correspond to the policy after 20,000 updates. We emphasise that this may not be a fair comparison because some policies converge sooner than others.

The input to the vision encoder f_e^v is 64×64 RGB, and depth images cropped from the center. We optimise our model using Adam [138] with a learning rate of 2.5×10^{-4} for the policy network and 1×10^{-3} for the pre-trained audio and vision networks using PyTorch [220].

5.2.4 Experiments

Environment

Simulator and Semantic Sounds. We use SoundSpaces [52], a visually- and acoustically-realistic simulation platform, to simulate an agent navigating in 3D house environments. The simulator renders sounds at any pair of source (sounding object) and receiver (agent) locations on a uniform grid of nodes spaced by 1 meter. While, SoundSpaces supports two real-world environment scans (Replica [266] and Matterport3D [43]), we used Matterport3D as it provides a larger number of houses and object-region semantics therein. We use the same 21 object categories as [53] for Matterport3D: chair, table, picture, cabinet, cushion, sofa, bed, chest of drawers, plant, sink, toilet, stool, towel, tv monitor, shower, bathtub, counter, fireplace, gym equipment, seating, and clothes. These object categories are visually present in the 24 regions (balcony, bathroom, bedroom, closet, dining room, entryway/foyer/lobby, familyroom/lounge, hallway, junk, kitchen, laundryroom/mudroom, living room, lounge, meetingroom/conferenceroom, office, other room, porch/terrace/deck, rec/game, spa/sauna, toilet, utilityroom/toolroom, and workout/gym/exercise) of the 85 Matterport3D houses. We use the publicly available sound clips from the experiment performed by [53], in which audio clips from `freesound.org` database were used. We generate sound by rendering the specific sound that semantically matches the object at the locations in Matterport3D houses. For example, the water-dropping sound will be associated with the sink in the kitchen.

Rewards and Episodes. The agent receives a sparse reward of $+10$ when it reaches the goal successfully, a dense reward of $+1$ for reducing the geodesic distance to the goal, and an equivalent negative reward for increasing it. To encourage trajectory efficiency, we also assign a negative reward of -0.01 per time step. To avoid simpler episodes, in which it is easy to reach goal (e.g., straight paths or short distance), we used 2 conditions while sampling episodes: 1) the ratio of geodesic distance to euclidean distance must be greater than 1.1; 2) the geodesic distance from the start location to the goal location must be greater than 4 meters. We sample 367,155 episodes for training and 1000 episodes for each of the testing settings.

Action space and sensors. There are 4 actions in the agent's action space: *MoveForward*, *TurnLeft*, *TurnRight*, and *Stop*. *MoveForward* changes the agent's current location to the node in front of it only if that node is reachable without collision. *Stop* can be used by the agent to report sounding objects and terminal the episode. The *TurnLeft*, *TurnRight*, and *Stop* actions can always be executed successfully. There are 4 sensory inputs: egocentric binaural sound (two-channel audio waveforms), RGB image, depth image, and the agent's current pose relative to the starting pose of the episode.

Episode specification and success criteria. An episode of semantic audio-visual embodied navigation task is defined by a house, a start location, and rotation angle of the agent, a goal location, a sounding object, and duration of the audio event. In each episode, the start location and rotation of the agent is randomly selected. For selecting the sounding object, an instance of an object category in the house is also chosen randomly. We define a set of viewpoints within 1 meter of the object's boundary for each sounding object. When the agent executes *Stop* action at any of these viewpoints, the episode will be successfully completed.

Baselines

We compare our model, K-SAVEN, against the following baselines:

- **Random walk**, a baseline which uniformly samples one of the three navigation actions and executes *Stop* automatically when the target sounding object is reached within 1m radius.
- **AudioGoal** [52], an end-to-end RL policy based on the PointGoal task [305] based on a Seq2Seq mechanism which uses a GRU state encoder that leverages colour and depth images to navigate the unknown environments. In contrast to PointGoal which uses GPS sensing to guide the agent toward its goal, this baseline uses audio spectrograms.
- **AudioObjectGoal** a Seq2Seq mechanism similar to (2) but the agent is also provided with the semantic label of the target object.
- **SAVi** [53], a transformer-based model that uses a goal descriptor network, which predicts both spatial and semantic properties of the target sounding object. It is the state-of-the-art deep reinforcement learning model for the semantic audio-visual embodied navigation task.

Evaluation Metrics

We follow [53, 54] in reporting agent performance, based on the following metrics: 1) success rate (SR; proportion of successful episodes); 2) success rate weighted by path length (SPL; a proxy for trajectory length-efficiency); 3) success rate weighted by the number of actions (SNA; a proxy for action-efficiency, penalising in-place rotation actions); 4) average distance to goal (DTG) on episode success/termination; and 5) success when silent (SWS; the proportion of successful episodes when the agent reaches the target sounding object after the end of the acoustic event).

We assess model generalisation by evaluating our method on unheard sounding objects. More specifically, we evaluate our model on the following testing settings: 1) test on seen houses with unheard sounding object categories as the navigation target; and 2) test on unseen houses with unheard sounding object categories. We randomly split the houses and sounding objects for training and testing. More specifically, we use 68 seen houses, 17 unseen houses, 16 heard sounding objects, and 5 unheard sounding objects. We average the results over 1,000 episodes for each setting.

Results

The performance comparison between the aforementioned baseline agents, across Seen-House/Heard-Sounds (SH/HS), Seen-House/Unheard-Sounds (SH/US), Unseen-House/Heard-Sounds (UH/HS), and Unseen-House/Unheard-Sounds (UH/US) conditions, is summarised in Table 5.3.

Navigation agents benefit from modular decomposition. We compare K-SAVEN to SAVi and to the end-to-end RL policies, AudioGoal and AudioObjectGoal; we observe DTG improvements in all evaluation conditions, implying higher confidence and accuracy about stopping locations in our approach.

Scene priors enable generalisation to unseen contexts. Recalling that our unique experimental conditions consist of performance evaluation over completely unheard sounds (as opposed to unheard clips from previously-heard objects), we highlight K-SAVEN's improvements over all baselines, on all metrics, in both of the sound-generalisation conditions (SH/US and UH/US). For the scene-generalisation conditions,

Table 5.3: Results of baseline models and our proposed approach on the Semantic Audio-Visual Embodied Navigation (SAVEN) task, evaluated with best checkpoints after 300M iterations.

	SEEN HOUSES, HEARD SOUNDS					SEEN HOUSES, UNHEARD SOUNDS				
Method	SR (↑)	SPL (↑)	SNA (↑)	DTG (↓)	SWS (↑)	SR (↑)	SPL (↑)	SNA (↑)	DTG (↓)	SWS (↑)
Random	—	—	—	—	—	0.07	0.02	0.87	15.20	0.07
AudioGoal [52]	—	—	—	—	—	0.11	0.11	0.08	14.06	0.04
AudioObjectGoal	—	—	—	—	—	0.11	0.11	0.08	14.73	0.03
SAVi [53]	0.67	0.54	0.53	1.65	0.38	0.22	0.16	0.14	6.54	0.12
K-SAVEN (*ours*)	**0.72**	**0.59**	**0.61**	**1.63**	**0.39**	**0.30**	**0.22**	**0.21**	**6.21**	**0.15**
	UNSEEN HOUSES, HEARD SOUNDS					UNSEEN HOUSES, UNHEARD SOUNDS				
Method	SR (↑)	SPL (↑)	SNA (↑)	DTG (↓)	SWS (↑)	SR (↑)	SPL (↑)	SNA (↑)	DTG (↓)	SWS (↑)
Random	0.05	0.01	0.01	16.45	0.05	0.07	0.02	0.01	15.20	0.07
AudioGoal [52]	—	—	—	—	—	0.12	0.12	0.08	14.16	0.04
AudioObjectGoal	—	—	—	—	—	0.15	0.13	0.10	13.06	0.05
SAVi [53]	0.32	0.21	0.18	10.12	**0.18**	0.15	0.11	0.09	9.97	0.08
K-SAVEN (*ours*)	**0.33**	**0.23**	**0.21**	**9.41**	0.16	**0.21**	**0.14**	**0.12**	**9.33**	**0.10**

K-SAVEN enjoys improved performance over all baselines, for all metrics, except for SWS: we hypothesise that this is due to the inclusion of acoustic representations in memory, on the part of SAVi, which we leave as a possible K-SAVEN configuration to explore in future work.

Scene priors enable robustness to silent episodic segments. In general, K-SAVEN performs significantly better in the success when silent (SWS) metric than the baseline methods. This shows that scene priors enable robustness to silent episodic segments. The reasoning capability provided by the scene priors helps the agent focus on the sounding object even when the acoustic signal stops.

5.2.5 Related Work

Modularity in goal-driven robot navigation. Goal-oriented navigation tasks have long been a topic of research in robotics [40, 134, 144, 157]. Classical approaches generally tackle such tasks through non-learning techniques for searching and planning, e.g., heuristic-based search [144] and probabilistic planning [134]. Although classical approaches might offer better generalisation and optimality guarantees in low-dimensional settings, they often assume accurate state estimation and cannot operate on high dimensional raw sensor inputs [105]. More recently, researchers have geared toward data-driven techniques, e.g., deep reinforcement learning [21, 46, 53, 54, 96, 305, 316] and imitation learning [128, 150], to design goal-driven navigation policies. End-to-end mechanisms have proven to be powerful tools for extracting meaningful features from raw sensor data, and thus, are often favoured for the setting where agents are tasked with learning to navigate toward goals in unknown environments using mainly raw sensory inputs. However, as task complexity increases, these types of systems generally exhibit significant performance drops, especially in unseen scenarios and in long-horizon tasks [105, 248].

To address the aforementioned limitations, modular decomposition has been explored in recent embodied

tasks. Chaplot et al. [48] design a modular approach for visual navigation consisting of a module that builds and updates a map of the environment, and a global and local policies to, respectively, predict the next sub-goal using such map and the low-level actions to reach it. Irshad et al. [128] also leverage a hierarchical setup to disentangle Vision-Language Navigation (VLN) [9] into a global policy tasked with grounding the input modalities and predicting the next global step, and a local policy that performs motion control to navigate toward it. Gordon et al. [105] design a hierarchical controller that invokes different low-level controllers in charge of different tasks such as planning, exploration and perception. Similarly, Saha et al. [248] design a modular mechanism for Vision-Language tasks that breaks down the task into multiple sub-tasks that include: mapping, language understanding, modality grounding, and planning. The aforementioned modular designs have shown to increase task performance and generalisability, especially in unexplored scenarios, compared to their end-to-end counterparts. Motivated by the aforementioned, we develop a modular framework for semantic audio-visual navigation, which includes pre-trained and knowledge-enhanced scene priors, enabling improved unseen generalisation.

Knowledge graphs in visual navigation. Combining prior knowledge with machine learning systems remains a widely-investigated topic in various research fields, such as natural language processing [94, 182, 183], due to the improvements in generalisability and sample-efficiency that symbolic representation promises for learning-based approaches. Historically, integrating symbolic knowledge with, e.g., navigation agents has proven non-trivial, yielding a collection of research areas focusing on smaller components of the problem—such as finding the appropriate representation of the knowledge (e.g., logical formalism, knowledge graphs, probabilistic graphical models), the appropriate *type* of knowledge that should be encoded (e.g., spatial commonsense, declarative facts, etc.), and the best knowledge injection mechanism (e.g., graph convolutional networks, grounded natural language, etc.) [182]. Knowledge graphs have gained popularity due to their interpretability and general availability as existing large-scale resources, such as ConceptNet [264] and VisualGenome [152]. Fortuitously, graph processing of structured data has experienced a surge of popularity in deep learning in recent years, leading to renewed interest in this neuro-symbolism [211, 309]. Some works in visual navigation tasks exploit knowledge graphs in the pursuit of generalisation [79, 180, 198, 291, 316]. [316] create knowledge graphs based on the VisualGenome [152] and inject features extracted from the graph as prior knowledge in visual navigation. In similar fashion, [227] provide agents with knowledge of object-object relational semantics. [180] show improvements in goal-directed visual navigation by injection 3D spatial knowledge into learning-based agents. Inspired by these works, we construct a knowledge graph that includes *all* object-object, object-region, and region-region semantics, which enables the more complex reasoning path, *sound* $\rightarrow$ *object* $\rightarrow$ *region*, in audio-visual navigation. Therefore, to our best knowledge, we become the first to study knowledge-driven scene priors for the audio-visual navigation task family.

Generalisation to unseen contexts. Chen et al. [52, 53, 54] leverage the SoundSpaces [52] simulation environment and dataset to design and assess Audio-Visual Navigation policies. The dataset is based on photorealistic indoor environments from the Matterport3D [43] and Replica [266] datasets, to which 102 sound sources commonly found in indoor environments (e.g., household appliances, musical instruments, telephones, etc.) were incorporated. The SoundSpaces dataset is split, such that indoor scenes encountered during testing are not found in the episodes used during the training stage. However, sounds of objects encountered during training may also appear during testing. Gan et al. [96] also explore Audio-Visual Navigation, but using the simulation platform AI2-THOR [146], which contains computer-generated graphical imagery. The authors introduce the Visual-Audio Room (VAR) benchmark consisting of seven different indoor environments—two of which were used for training and five for testing. The VAR benchmark incorporates three different audio categories: ring tone, alert alarm, and clocks. Similar to the AVN task

introduced before, sound sources are found both in the training scenes, as well as and the testing scenes. In this paper, we argue that in the context of Audio-Visual Navigation tasks, generalisation to unseen environments pertains to both generalising to unseen visual scenes, as well as to unheard sounds. Current Audio-Visual benchmarks do not take into consideration the latter. Thus, there is no direct assessment of generalisation performance to unheard sounds. To tackle this limitation, we propose a curated version of the SoundSpaces dataset where we evaluate our agent in two different settings: (1) seen scenes and unheard sounds, (2) unseen scenes and unheard sounds.

5.3 Learning Representations of Reusable Robot Maneuvers

Vision-Language Navigation (VLN) tasks remain challenging for artificial agents, as they must satisfy complex natural language instructions to navigate within complex photorealistic, partially-observable environments. Recent pursuit of this task has focused on pragmatic decoding, agent progress-monitoring, back-translation, and cross-modal grounding; however, issues with instruction complexity and label-independent variation in the visual features still persist, reducing agents' generalisation performance in unseen environments. In this chapter, we propose a framework for distilling a navigation agent's experience into a representation of reusable maneuvers, where the high-frequency and label-independent variation in the instructions and visual context are removed. Our approach encourages the agent to generalise reusable navigation maneuvers on the basis of similarities across high-level textual instructions, conditioned on past actions, in order to leverage experience in executing familiar sub-commands from *new* instructions. We achieve this association through a method called auxiliary variable variation approximation, which allows us to introduce a latent variable for *estimating* the conditional posterior distribution over all observations and executed trajectories (otherwise intractable for most distributions of interest); this is essentially a special case of variationl behaviour cloning, between an expert (a learned distributional prior over robot maneuvers) and a student (a jointly-learned posterior distribution over action+visual trajectories). The agent then learns how to compose samples from this skill space and couple them with conventional multimodal co-grounding mechanisms, through policy refinement, for improved generalisability and downstream performance. We perform extensive baseline error-analysis, illustrating the limits in models' generalisation to complex scenarios, where we show improvements from our approach.

5.3.1 Problem Formulation

We adopt the problem setting of Vision-and-Language Navigation (VLN) [9], wherein our task is to navigate realistic indoor environments, using text-based instructions, visual context, and a partially-observable navigation state graph. Thus, our embodied agent is required to coordinate directions with visual surroundings, execute a trajectory between multiple rooms (or regions of the same room), and to arrive as close as possible to goal state that is referenced by the instruction. Formally, we are given: (i) an undirected graph $G =< V, E >$ that is defined over a set of panoramic viewpoints V and edges E; (ii) an arbitrary initial pose $p_0 = (v_0, \phi_0, \theta_0)$ consisting of a initial spatial position v_0, initial camera heading ϕ_0, and initial camera elevation θ_0; and (iii) a natural language text instruction $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ with average sequence length n of 29 tokens. The task is to select a set of deterministic actions $\{a_1, a_2, ...a_T\} \in A$, such that the agent arrives within 3 meters of the target spatial position, v_{target}. For each state, the agent receives a set of next-step reachable viewpoints, $W_{t+1} \subseteq V$, such that $W_{t+1} = \{v_t\} \cup \{v_i \in V | < v_t, v_i > \in E \wedge v_i \in P_t\}$, and must choose $a_{t+1} = f(v_{t+1}) \in W_{t+1}$ and nominate any changes in camera heading ($\Delta\phi_{t+1}$) or camera elevation ($\Delta\theta_{t+1}$). Here, P_t is the region enclosed by the camera's left and right frustrums, implying that the only actions available to the agent at any given time are next-states defined by the graph that are

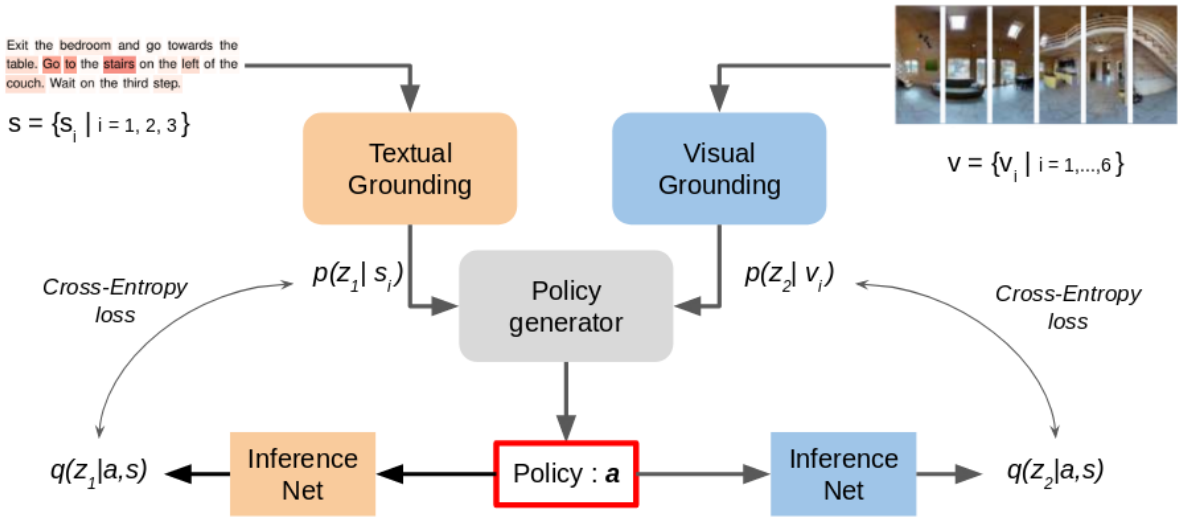


Figure 5.4: The main illustration of our approach, Auxiliary Variable Variational Approximation (AVVA-agent) for Vision-and-Language Navigation.

within the field-of-view of the agent.

5.3.2 AVVA-agent: Auxiliary Variable Variational Approximation

We are reminded that the objective of VLN is to transform high-level natural language navigation instructions into sequences of low-level discretised actions. Here, we inherit the original formulation of the VLN task, as in Room-to-Room Navigation [9], where access to the entire scene navigation graph was not available to the agent (thereby precluding motion-planning based approaches), nor were segmented sub-goal instructions (removing intermediate task supervision), nor fine-grained alignment between instructions and trajectories in the dataset (precluding the use of instruction templates).

Much of the previous methodology leverages a sequence-to-sequence recurrent policy agent (RPA), based on an LSTM decoder as an action generator, posed for the VLN-R2R task by the originating dataset paper [9]. While the approach enjoys considerable simplicity in its architecture and utilisation, the approach is faced with issues in generalisability and limits in modelling capacity. More sophisticated solutions involved posing the R2R-VLN task as a trajectory search problem, wherein the agent needs to find the best trajectory in the environment to navigate from the start location to the goal location, given the natural language instruction [95]. Towards this problem, Fried et al. [95] proposed a model assembly, consisting of an instruction-interpretation module (*Follower*, which maps different instructions to actions sequences) and an instruction-generation module (*Speaker*, which maps action sequences to instructions). The proposed speaker module is leveraged to augment data during training and is also used to perform pragmatic inference to handle ambiguous situations. While this Speaker-Following assembly created promising avenues for data augmentation and self-supervision of navigation agents, this came at the cost of generating long trajectories—sometimes, multiple kilometers in length, limiting the approach's relevance for deployment in constrained-resource scenarios as well as its performance on efficiency-weighted success metrics. Ma et al. [181] introduce the Self-monitoring Agent (SMA), which attempts to rectify generalisability and trajectory length issues by combining visual and text co-grounding streams, for improved multimodal alignment and task progress-monitoring capability. However, empirical error analyses (Sec-

tion 5.3.4) show weakness to various forms of structured perturbation, including visual noise-injection, region-substitution, and navigable action noise—showing that a moderate gap remains, necessitating better learning strategies for multimodal alignment, especially.

In this work, we pose VLN as a multimodal alignment problem, where the agent should learn to associate its step-wise visual context with the textual command it received initially. The agent learns to represent primitive 'skills' in an embedding space z, which is conditioned on task ID embeddings. Departing from previous work in the area of learning robot skills, which used simple one-hot vector representations as task IDs, we seek to use encoded natural language instructions directly. Fundamentally, the problem is to organise a latent space of all solutions to a set of relevant tasks, given some arbitrary task specification. To this end, we hypothesise that similarities in the sub-instructions map to similarities in the sequences of executed actions, and that these action sequences form behaviour primitives, which we call *maneuvers*, that generalise across tasks. Explicitly supervising action trajectories at the sub-instruction level will take a lot of time/effort, will require us to generate additional supervisory datasets, and may not even be generalisable. Exploring all the distinct solutions to all types of tasks has exponential complexity and is therefore intractable. Therefore, we utilise a method called *auxiliary variable variational approximation* to estimate the probability distribution over some embedding space—conditioned on textual, visual, and action trajectory information. We can think of the modes in this embedding space as representing distinct skills or *maneuvers*, learned through an optimisation process. Figure 5.4 illustrates our overall approach.

Predicate decomposition on instructions. We hypothesise that the best natural language articulation, for the maneuvers we wish to represent in the skill embedding space, takes the form of *sub-instructions*—segments of instructions that agents receive at the beginning of episodes—which contain action-oriented predicate clauses. These clauses should map to action sequences that are frequently executed in the environment, across various instructions and even multiple times in the same instruction; these primitive clauses (and the corresponding abstracted action sequences) should also feature in other tasks in the family of Embodied Vision-Language Planning (EVLP) tasks [94], which includes R2R-VLN. Because natural language instructions in EVLP tasks are generally free-form and may not be pre-segmented, we start by decomposing the full instructions in the R2R-VLN task, into predicate clauses. We leverage the AllenNLP semantic role labelling model BERT-based model (SRL-BERT) [100, 257] to perform predicate decomposition on all the instructions in the training split of the R2R-VLN dataset.[2] We then paired the sub-instructions with the original instruction, for each sample in the training set; on average, where, on average, each instruction yields 2-4 sub-instructions.

Embedding network and observation-encoding. At the beginning of the episode, the agent is initialised in the simulated environment and receives an initial observation of state, as a tuple of image feature v_0 and instruction x_0, and must generate an action a_0 to execute in the environment. As a consequence of predicate decomposition on the instruction, the agent also receives a collection of sub-instructions $s = \{s_i \mid i = 1, 2, ..., k\}$. We use an `Instruction Encoder` to embed each sub-instruction, then project the embeddings into different regions of the same latent representation, using a variational autoencoder (`Embedding Network`). In other words, there are shared weights across all k embedded sub-instruction projections. Each projection represents $p(\mathbf{z}^{\text{skill}}|s_i)$ for some sub-instruction s_i, with $1 \leq i \leq k$. Notably, our approach uses each sub-instruction as a skill ID (task specifier) conditioning variable for latent vector $\mathbf{z}^{\text{skill}}$; being an encoding of natural language, s_i is a much noisier and complex signal, compared to the one-hot vector representations used at task IDs in much of the skills literature [112]. It is

[2]We also added the SRL tags to the lexicon, to facilitate mappings to the language embedding space; we found ArgM-LOC, ArgM-DIR, ArgM-MNR to be especially important.

for this reason that we separate the `Instruction Encoder` and `Embedding Network` and can subsequently impose sparsity regularisation on the embedding network's representation, e.g., using an unsupervised contrastive objective [300]. Subsequent to the agent's initial spawning in the environment, and at each timestep t thereafter, the agent receives a tuple of the current image feature (according to the agent's pose) $\mathbf{v}_t$ and generates a summary state representation h_t using a recurrent policy model. Following Ma et al. [181], we construct a visually-grounded full-instruction representation $\hat{\mathbf{x}}_t$ (Eqn. 5.3) and textually-grounded visual representation $\hat{\mathbf{v}}_t$ (Eqn. 5.5):

$$\mathbf{h}_t = \texttt{Policy}([\hat{\mathbf{x}}_t, \hat{\mathbf{v}}_t, \mathbf{a}_{t-1}, \mathbf{z}^{\text{skill}}_{t,s_i}]), \quad \text{where} \tag{5.1}$$

$$z^{\text{text}}_{t,l} = (\mathbf{W}_x \mathbf{h}_{t-1})^\top \texttt{PE}(x_l), \quad \alpha_t = \text{softmax}(\mathbf{z}^{\text{text}}_t) \tag{5.2}$$

$$\hat{\mathbf{x}}_t = \alpha_t^T \mathbf{X} \tag{5.3}$$

$$z^{\text{visual}}_{t,d} = (\mathbf{W}_v \mathbf{h}_{t-1})^\top g(\mathbf{v}_{t,k}), \quad \beta_t = \text{softmax}(\mathbf{z}^{\text{visual}}_t) \tag{5.4}$$

$$\hat{\mathbf{v}}_t = \beta_t^T \mathbf{v} \tag{5.5}$$

Here, "$[,]$" denotes concatenation, $\mathbf{a}_{t-1}$ is the previously-executed action, $z^{\text{text}}_{t,l}$ and $z^{\text{visual}}_{t,k}$ are scalars, $\mathbf{W}_x$ and $\mathbf{W}_v$ are the learnable parameters, $\texttt{PE}(\cdot)$ is a positional encoding transform [288], l indexes a word token, d indexes a navigable direction, and $g(\cdot)$ is a multi-layer perceptron (MLP).

Inference network and action-generation. All k probability distributions, according to the k sub-instructions are sampled from $p(z)$ and used to generate an attention distribution over the set of k latent spaces. The highest-scoring latent space is then fed to a recurrent policy network, along with the historical trajectory information. In this manner, we generate a sequence of action predictions. These action predictions are encoded using an LSTM (`TrajectoryEncoder`), whose output is then projected to another latent space (`Inference Network`); this second projection implements the posterior probability distribution over the latent space, conditioned on historical actions and visual information $p(z|a, v)$. Action-selection is performed via an inner product between the image features on the navigable directions that correlate most with grounded instructions $\hat{\mathbf{x}}_t$ and the current hidden state $\mathbf{h}_t$:

$$a_t \overset{\text{cat}}{\sim} \text{softmax}((\mathbf{W}_a[\mathbf{h}_t, \hat{\mathbf{x}}_t])^\top g(\mathbf{v}_{t,k})), \tag{5.6}$$

where "$\overset{\text{cat}}{\sim}$" denotes categorical sampling, used during training, $\mathbf{W}_a$ are learnable weights (omitting the bias term, for conciseness), and $g(\cdot)$ is the same as in Eqn. 5.4.

5.3.3 Learning and Optimisation

We choose a stochastic recurrent policy $\pi(\cdot)$ that is trained through *student-forcing*, in order to generate actions at each time-step in the roll-out; architecturally, this recurrent policy network as $\texttt{Policy}(\cdot)$, above, which follows from the agents in Fried et al. [95], Ma et al. [181], but with entropy regularisation to encourage the z-space to have reduced sparsity in the local neighbourhoods of task primitives. We want to query posterior skill distribution $p(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)$, however this quantity is intractable, as it must represent all possible configurations of the latent space with respect to action, visual, and textual information. The solution is to define a variational posterior $q(\mathbf{z}^{\text{skill}}|a, \mathbf{h})$ which leads to a variational lower bound ($\mathcal{L}_{var}$) on the entropy $\mathcal{H}(\cdot)$. Note that we avoid conditioning the variational term on the task ID, as we want to encourage identifiability of the latent skill vector $\mathbf{z}^{\text{skill}}$ from the trajectory $\mathbf{h}$, which also contains co-grounded visual- and positionally-encoded, attended textual information. Maximising this variational

lower bound would indirectly maximise the entropy:

$$\mathcal{H}(\pi(a|\mathbf{h}, s_i)) = \mathbb{E}_{\pi}(-log(\pi(a|\mathbf{h}, s_i))) \tag{5.7}$$

$$\geq \mathbb{E}_{\pi(a,\mathbf{z}^{\text{skill}}|\mathbf{h},s_i)}[log(\frac{q(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)}{\pi(a, \mathbf{z}^{\text{skill}}|\mathbf{h}, s_i))})] \tag{5.8}$$

$$= -\mathbb{E}_{\pi_\theta(a|\mathbf{h},s_i)}[\texttt{CE}[p(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)||q_\psi(\mathbf{z}^{\text{skill}}|a, \mathbf{h})]] \tag{5.9}$$

$$+ \mathcal{H}[p_\phi(\mathbf{z}^{\text{skill}}|s_i)] + \mathbb{E}_{p_\phi(\mathbf{z}^{\text{skill}}|s_i)}[\mathcal{H}[\pi_\theta(a|\mathbf{h}, \mathbf{z}^{\text{skill}})]] \tag{5.10}$$

$$= \mathcal{L}_{\text{var}} \tag{5.11}$$

The Cross-Entropy $\texttt{CE}(\cdot)$ between $p(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)$ and $q_\psi(\mathbf{z}^{\text{skill}}|a, \mathbf{h})$ aligns the two distributions, as an auxiliary training objective, similar to Hausman et al. [112]. Remembering that $p(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)$ is intractable, we utilise a sample-based evaluation of CE:

$$\mathbb{E}_{\pi_\theta(a|\mathbf{h},s_i)}[\texttt{CE}[p(\mathbf{z}^{\text{skill}}|a, \mathbf{h}, s_i)||q_\psi(\mathbf{z}^{\text{skill}}|a, \mathbf{h})]] = \mathbb{E}_{\pi_\theta(a,\mathbf{z}^{\text{skill}}|\mathbf{h},s_i)}[-log\ q_\psi(\mathbf{z}^{\text{skill}}|a, \mathbf{h})] \tag{5.12}$$

The two latent space projections, `Embedding Network` and `Inference Network`, are the encoders of two separately instantiated VAEs. These VAE-encoders are implemented as two-layer MLPs, that map a 1024-dimensional input to a latent space of dimension 512 with ReLU activations, in between. The sub-instruction embeddings are of size 256; the TrajectoryEncoder produces an output of size 512. The full pipeline was trained on a GPU cluster with two Titan X Pascals and two GTX 1080s, using Adam optimisation, over 300 epochs, with a learning rate that decayed from $1 \cdot e^{-4}$ to $1 \cdot e^{-6}$.

5.3.4 Experiments

Dataset

The MatterPort3D dataset [43] provides 194,400 RGB-D images, sampled from 10,800 panoramic views in 90 building-scale scenes. This dataset also provides an undirected navigation graph for state enumeration, consisting of an average position separation of 2.25 meters and an average vertex degree of 4.1, where the images were sampled in 6 degree-of-freedom at each node. Anderson et al. [9] provide 21,567 crowd-sourced, Room-to-Room (R2R) navigation instructions that are paired with paths/trajectories from the original dataset. The verbal instructions in the R2R dataset define navigation tasks that agents must perform within indoor scenes in the Matterport dataset. The average navigation graph trajectory length for each instruction is 10 meters, and the average instruction sequence length is 29 words.

There are a few notable challenges in performing the VLN-R2R task: the first is that the VLN task does not provide access to the entire navigation graph in a scene, precluding the use of common motion-planning techniques [94]. The second challenge is that the VLN task only offers agents a sparse reward, of task completion, with no intermediate supervision (e.g., sub-goals, waypoints) provided. As a result, agents are susceptible to the classic temporal credit assignment problem [272], where, upon completion or failure of the downstream task, they are not given information about what intermediate decisions led to the eventual outcome. This limits the model's implicit progress-monitoring capability and prevents the model from generalising short-term behaviours across various instructions and scenes. The third major challenge is in understanding how to leverage the agent's own navigation history for cross-modal alignment between the textual and visual contexts.

Evaluation Metrics

We adopt the standard metrics, originally defined for the VLN-R2R task [9]:

- *Trajectory Length* (TL) — defined in terms of the agent's average trajectory length in metres.
- *Navigation Error* (NE) — average distance between the end-location predicted by the agent and the true route's end-location.
- *Success Rate* (SR) — percentage of predicted end-locations within 3m of the true end- locations.
- *Oracle Success Rate* (OSR) — the closest your trajectory is to the goal point when choosing the ideal stopping time rather than the policy stopping time.
- *Success rate weighted by normalised inverse Path Length* (SPL) — trades-off SR against TL giving importance to solving VLN tasks with efficient trajectories.

Baseline Error-analysis

In this section, we report the results obtained by re-implementing the RPA-Seq2Seq [9] and Speaker-Follower [95] baselines on the validation seen (val-seen), validation unseen (val-unseen) and test sets. We also include a comparison between Look-Before-You-Leap [299] and Speaker-Follower. We report performance in terms of Navigation Error (NE), Success rate (SR), and Oracle Success Rate (OSR).

Recurrent Policy Agent. Our implementation of the Recurrent Policy Agent (RPA) and downstream performance follow those of the Seq2Seq baseline achieved comparable performance with the one reported in the originating paper [9]. As seen in Table 5.4, the trained model achieved a success rate of 37.84, 21.03 and 20.27 on val-seen, val-unseen and test splits respectively. The trained model performed slightly better for the OSR of val-unseen than the reported score but performs a little worse on the other metrics.

In Figure 5.5 (Student-forcing RPA-Seq2Seq), we can see that a majority of the instructions (approx. 60% for val-seen and approx. 80% for val-unseen) have navigation error more than 3m with more failures in case of unseen environment (SR of 21.03). Thus, the Seq-2-Seq model doesn't generalise to the unseen environments. Moreover, from Table 5.4, we can see that OSR is more that SR, thus, it is evident that the agent also does not learn to generate stop action or learn when to stop. Another observation from Figure 5.7 (left) is that the trajectory length for the Seq-2-seq model is comparable to the trajectory length corresponding to the shortest path (Figure 5.6, left). Surprisingly, the results for the number of trajectory steps for the seq2seq model is maximum around 20 (see Figure 5.8, left) as opposed to the ground truth shortest path (see Figure 5.6, right). This analysis suggests that the seq-2-seq model has just learnt to stop after taking 20 actions (or trajectory steps) irrespective of the environment and the natural language instruction. This experiment also point towards the dataset bias where agent might achieve higher scores irrespective of the visual modality.

Speaker-Follower Agent. Our re-implementation of the Speaker-Follower model [95] results in the same scores as reported by the authors in their paper. The model achieves better scores in the validation-seen environment but performs slightly worse in validation-seen and test environment in the order of decimals. As seen in Table 5.5, the model achieves a success rate of 53.3% which is approx. 30 points higher than RPA-Seq2Seq model (absolute terms).

One important observation is that the Speaker-Follower model is able to achieve a high OSR of 96% but nearly 42 points lower SR (of 54%) (see Table 5.5). From this, we can clearly infer that for about 96%

of the instructions, the agent is able to reach within 3m of the goal location but does not learn to stop about 42% of them. Thus, degrading the performance in terms of the SR. This calls for the need of a technique which helps the agent identify its proximity to the goal and predict a stop action. Further, from Figure 5.5 (Speaker Follower), we can see that a majority of the instructions for val-seen and val-unseen have navigation error less than 3 meters. In contrast to the Seq-2-Seq model, the Speaker Follower model generalises to the unseen environment. Moreover, from Figure 5.7 and Figure 5.8, it is clear that the Speaker-Follower model explores a lot before gaining in terms of the SR as compared to the actual ground truth shortest path statistics (see Figure 5.6).

Examples: RPA-Seq2Seq versus Speaker-Follower. We compared the Recurrent Policy Agent (RPA) with the Speaker-Follower Model, on the basis of a handful of randomly-sampled instructions from the VLN-R2R dataset. We show two such instances, below:

> (i) *Go up the steps. Go into the house using the sliding doors on the left. Go straight until you get to the fridge and wait there.*
> (ii) *Walk forward down the hall, and take a left at the hall. Enter the bedroom at the end of the hall and stop once you are on the rug next to the bed.*

In the first example (i), the Speaker-Follower Model takes 540 steps, traversing 1268.22 meters, while the ground-truth shortest path consists of 13.75 meters. For the same sample, the RPA-Seq2Seq model takes 12 steps with an trajectory length of 29.166 meters. In the second example (ii), The Speaker Follower Model takes 674 steps, traversing 1676.967 meters, while the RPA-Seq2Seq takes around 15 steps to explore the same area with trajectory length of 24.33 meters. These examples imply that models actually get confused when they are not able to recognise objects, during the navigation. Below, we analyse the Self-Monitoring Agent [181], which attempted to abate these issues in multimodal grounding.

Self-monitoring Agent. The Self-monitoring Agent (SMA; [181]) was one of the first instances of a new class of VLN agents, which attempted multimodal alignment and progress-monitoring, through cross-modal attention. The intention was to correct previously-faced issues in multimodal grounding and agents' lack of understanding of when to stop execution. The grounding mechanisms introduced by this agent make the method particularly suitable as a baseline, for helping to characterise the importance of each modality on the VLN-R2R task. We do this through ablation: we inject visual feature noise, perturb navigation directions, and perform structured token-substitution in the instructions, in order to understand the baseline's unimodal dependence and robustness characteristics.

In Table 5.6, we compare the unaltered baselines of Recurrent Policy Agent (RPA), Speaker-Follower Agent (SFA), and Self-monitoring Agent (SMA) to perturbed SMA variants, on the Validation-Seen (1021 instructions) and Validation-Unseen (2349 instructions) splits of the VLN-R2R dataset. For `visual noise-injection (random)` perturbations, we replace {*noise-rate*}% of the instructions in the respective dataset split with random visual feature vectors: the goal is understand the dependence of the model on the visual modality in the VLN-R2R task. For the `visual noise-injection (average)` perturbation, we replace {*noise-rate*}% of the instructions with the average visual feature vector, computed across the respective dataset split: the goal is to understand the extent to which the model's action-generation component has simply memorised the average action sequence learned from the training set; if this is the case, we expect for *average* visual noise-injection to yield better results than the *random* visual noise-injection. Additionally, we want to evaluate the strength of the model's cross-modal grounding mechanism, to understand the extent to which the agent actually pays attention to region references; we want to empirically observe how much the model is affected, after making structured pertur-

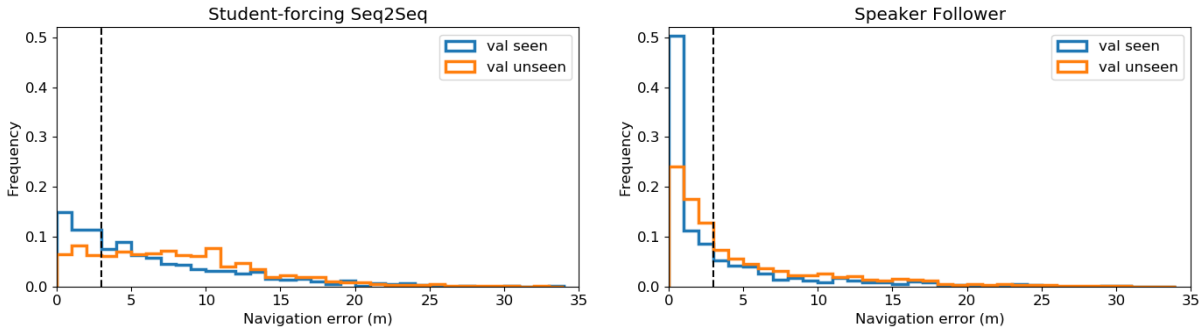


Figure 5.5: Plot showing the comparison between the navigation error histogram of val-seen and val-unseen environments for RPA-Seq2Seq and Speaker-Follower models.

bations to its natural language input. One way to achieve this is by making substitutions to the region-mentions in the instructions. For this, we consider the most frequently-mentioned regions in the VLN-R2R dataset, namely, *bedrooms* and *bathrooms*; these are also the regions in which the agent is most frequently spawned, during the initialisation of the simulated task environment. Indeed, the `structured region-substitution` perturbation is of such that, for {*noise-rate*}% number of instructions that include either bedroom or bathroom references, we substitute one for the other and measure downstream task performance. Finally, we want to empirically evaluate the dependence agents have on the initial heading, for downstream task performance; we use this as a proxy for determining the amount of exploration agents perform, versus attempts to directly execute actions according to the natural language instructions. For `Navigable action noise`, we perturb the first action taken by the agent, in {*noise-rate*}% of the instructions, for the tasks whose ground-truth trajectories have a two-way branching decision; we report downstream performance on Validation-Seen and Validation-Unseen.

Both expected and unexpected insights result from the above structured perturbations. In the case of `visual noise-injection (random)`, we see the expected degradation trend in SR and SPL; we also confirm the increased (relative) performance from average-feature visual perturbations, compared to random-feature visual perturbations. Regarding the `structured region-substitution`, however we observe that even substantially different noise rates actually have minimal effect on the downstream performance, suggesting that models could benefit from a language-encoding stream that better captures the nuanced distinctions between the various regions. Considering the `Navigable action noise` we also see that the substantially different noise rates actually have minimal effect — this time, suggesting that the action-generation component (LSTM-based recurrent policy) may be overfitting, i.e., memorising the dataset average from training. Here, a higher-capacity policy architecture may be important to better represent grounded actions.

Analysis of Visual Modality

Effectiveness of feature representation. We conducted several experiments comparing the effect of the representation space. Speaker-Follower computes features on the training set using the Resnet architectural backbone, *a priori*. Already, the authors are augmenting a direct sequential architecture (like VGG, for example) with residual layers; we compare this against the architectures proposed in Huang [125] and Lin [169], inspired by the competitive improvements from He et al. [113] and Hariharan [111]. Maintaining the same number of training iterations and holding the post-feature-extraction process identical

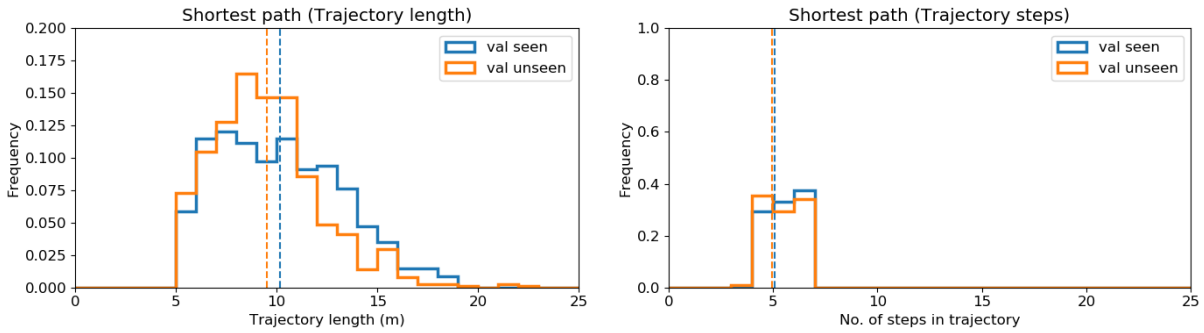


Figure 5.6: Plot showing the comparison between the trajectory length and the number of trajectory steps histogram of val-seen and val-unseen environments for shortest path.

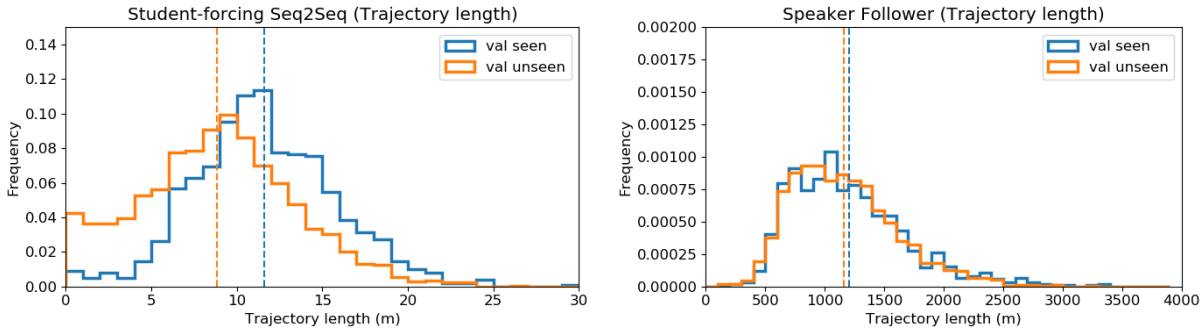


Figure 5.7: Plot showing the comparison between the trajectory length histogram of val-seen and val-unseen environments for RPA-Seq2Seq and Speaker-Follower models.

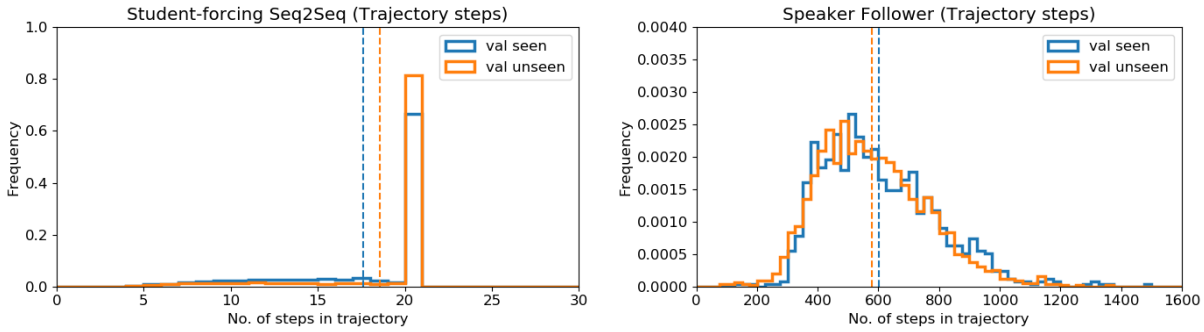


Figure 5.8: Plot showing the comparison between the number of trajectory steps histogram of val-seen and val-unseen environments for RPA-Seq2Seq and Speaker-Follower models.

Table 5.4: Baseline re-implementation results comparison of the Recurrent Policy Agent (RPA), on the Validation-Seen, Validation-Unseen and Test (Unseen) splits of the Vision-Language Navigation Room-to-Room (VLN-R2R) dataset [9].

	VALIDATION-SEEN			VALIDATION-UNSEEN			TEST (UNSEEN)		
Method	NE (↓)	SR (↑)	OSR (↑)	NE (↓)	SR (↑)	OSR (↑)	NE (↓)	SR (↑)	OSR (↑)
RPA [9]	11.33	38.60	52.90	7.81	21.80	28.40	7.85	20.40	26.60
RPA (*ours*)	11.59	37.84	51.86	8.03	21.03	29.08	7.86	20.27	26.45

Table 5.5: Baseline re-implementation results comparison of the Speaker-Follower Agent (SFA; [95]), on the Validation-Seen, Validation-Unseen and Test (Unseen) splits of the Vision-Language Navigation Room-to-Room (VLN-R2R) dataset [9].

	VALIDATION-SEEN			VALIDATION-UNSEEN			TEST (UNSEEN)		
Method	NE (↓)	SR (↑)	OSR (↑)	NE (↓)	SR (↑)	OSR (↑)	NE (↓)	SR (↑)	OSR (↑)
SFA [95]	3.08	70.01	78.30	4.83	54.60	65.20	4.87	53.50	96
SFA (*ours*)	3.04	70.68	78.40	4.90	54.02	64.40	4.89	53.30	95.60

to the authors' initial process, our results can be seen as a direct statement on the representation power of image-based CNNs, especially in domain-adaptive settings. We relied on the default settings of the learning algorithm after introducing hypercolumns and using reasonable pre-trained weights taken from ImageNet. We measure in terms of convergence time and final performance. By evaluating on observed and unobserved scenes, the experiment is designed to look at the generalizability of the underlying architecture in addition to its power in-domain. Notably, He et al. [113] notices an improvement of about 5-10% in its similar adaption of these networks in the image segmentation domain, though the difference of metric precludes direct comparison between the literature.

The Self-monitoring Agent naturally reports validation accuracy both in seen and unseen scenarios. We report improvement with respect to training speed and final performance on the success-rate metric in both scenarios. By default, and to maintain an injective comparison, Speaker-Follower measures over a fixed number of iterations. We observed that both new architectures, DCN and FPN, converged at roughly 70% of the total number of iterations required, as opposed to requiring the full convergence horizon in the original architecture. The success rate with Resnet50 was 70.1% on seen data and 54.6% on unseen data. FPN performed slightly better, converging to 74.6% in seen data and 58.6% in unseen data. DCN performed the best, adding additional improvement that brought the seen success rate to 76.1% and the unseen rate to 60.0% (note these numbers are slightly worse than reported in the midterm report, in which a bug in the evaluation script caused the number to be misreported). Directly, the results offer an initial evidenciary basis of support towards the notion that denser representation can improve performance in this domain; there is an apparent lack of support for the claim that this benefit is exaggerated in an unseen case.

Visual Domain Invariance. One of the persistent challenges to reinforcement learning arises from its inability to generalise well across different environments. The difference between the environments could be due to different reasons such as simulation vs. real environments [278] or different types of real

environments. With respect to the VLN task, we are interested in the real-to-real type transfer of policies. Consider an example where a VLN agent trained on indoor scenes is deployed in an industrial warehouse setup. The underlying distribution of the environment in the form of object types, object sizes, visual appearance of the floor, walls or ceilings could be significantly different under the two settings. This leads to a domain gap in the (vision based) observation space. Ideally, we would like to learn navigation policies which are invariant to such domain gaps. To realise this objective, we experiment with both supervised and unsupervised techniques. In parallel, we also experiment with state-of-the-art unsupervised methods like $\beta-$VAE [36] which strive to learn disentangled latent representations of the observed scenes. Such representations have been speculated to be useful for knowledge transfer tasks [155]. Our goal is to estimate the optimal dimension of the latent representation which could allow sufficient disentanglement of the covariates and, by extension, successful policy transfer across scenes.

Research Questions

We observed several challenges from the task and from the limitations of the prior art, based on our extensive error analysis in Section 5.3.4. Because agents have only partial observability over the state space (known transitions, unknown node locations), beyond just the adjacent navigable nodes, planning based approaches are not feasible. Furthermore, we observed that prior approaches tend to overfit to the common trajectories that were traversed in the training set, limiting generalisability to new layouts that require novel maneuvers. Next, the sparse reward structure of the task severely limits progress-monitoring capability, where even agents that attempt to assess task progress through multimodal grounding still face challenges from the complexity of the visual context.

In this work, we hypothesised that similarities in the (sub-)instructions map to similarities in the sequences of actions to execute in the environment. However, while we would ideally want to supervise action trajectories at the sub-instruction level, this requires significant expert time (manual annotation effort) and may not generalise naturally across scene layouts and environments. We want to evaluate the effect of our agent's ability to distil its experience into a set of behaviours (skills, primitives, maneuvers) to sample from and compose at test-time; we also wanted to encourage the agent to learn skills in a way that maximises its ability to generalise to unseen environments.

Results. We compare our approach with the baseline policy, Self-monitoring Agent (SMA) [181], on the basis of VLN-R2R task success metrics (SR, SPL, OSR) and task error metrics (NE, TL, and distance-from-goal[3])—all on the Validation-Unseen split of the dataset (see figures 5.9 and 5.10). We observe improvements to asymptotic performance on all success metrics and most error metrics.

5.3.5 Related Work

Vision, language, and navigation. Many works have pursued multimodal vision and language tasks, such as in visual question-answering (VQA) [11, 129, 165] and in textual image-caption generation [292]. Borrowing from the human-robot interaction domain, recent tasks have included a spatial navigation component to the problem [9]. This special pairing of language grounding with task-orientation is applicable to our focus in this paper: the embodied agent shall receive complex text-based instructions and must navigate a real-world (i.e., non-simulated) environment in real-time, to arrive at a goal. Mei et al. [187] look at neural mapping of instructions to sequence of actions, along with input from bag-of-word features extracted from the visual image. Robot navigation, conditioned on text-based instructions, has been the

[3]Distance-from-goal is defined as the stopped distance, away from the goal location, on episode completion.

focus of some recent work [68, 195], but only in simulated environments. Our task pertains to real-world environments, where: (1) we must deal with an arbitrary set of objects in each scene with complex human-curated language instructions, rather than templated queries as in Chaplot et al. [45], Misra et al. [195]; and (2) we must develop an agent to deal with a partially-observable environment, where only the adjacent nodes in the navigation graph are seen, compared to contemporary environments wherein knowledge of the entire state-space is given *a priori*, as in Das et al. [68], Wu et al. [308]. Chaplot et al. [45] proposed an end-to-end trainable architecture for task-oriented language grounding that combines image and text representation and learns a policy on top of this. These approaches mainly deal with synthetic environments and relatively short language instructions while our task is based on real-world environments with lengthy natural language instructions ($\sim$29 average word count).

Pre-existing work on the VLN-R2R task. Anderson et al. [9] introduced the R2R-VLN task and proposed the first baseline. They proposed to model the agent with a recurrent policy, using an LSTM-based sequence-to-sequence (Seq2Seq) encoder-decoder architecture, with an attention mechanism. More sophosticated solutions involved posing the R2R-VLN task as a trajectory search problem, wherein the agent needs to find the best trajectory in the environment to navigate from the start location to the goal location, given the natural language instruction [95]. Towards this problem, Fried et al. [95] proposed a model assembly, consisting of two modules: instruction intrepretation module (*follower*, which maps different instructions to actions sequences) and an instruction generation module (*speaker*, which maps action sequences to instructions). The proposed speaker module is leveraged to augment data during training and is also used to perform pragmatic inference to handle ambiguous situations. Inspired by Weber et al. [301], Wang et al. [299] propose an alternative solution to the R2R-VLN task, by combining model-free and model-based reinforcement learning. Their core contribution is the *Look-Ahead* module, which circumvents the issue of having a partially-observable state space, by estimating the cumulative sum of future rewards – given a trajectory rollout, based on an implicit transition model that is learned exclusively from the visual context. Due to the complexity of the visual scenes, however, their model suffers from generalisability problems – where the model experiences a significant performance gap across `seen` and `unseen` validation splits. Ma et al. [181] attempt to rectify this generalisability issue by combining visual and text grounding, in order to maintain an understanding of which part of instruction has been and is yet to be executed, and choosing where to go based on images from the navigable paths seen. In addition to cross-modal grounding between the textual and visual contexts, methods from the related domains of scene-graph reasoning [208] and multi-task learning [10] suggest that further conditioning the multimodal representation on an available (/predicted) navigation trajectory graph allows the model to inherit valuable *global* information about the interdependencies across the state distribution. Indeed, we hypothesise that the nature of the R2R-VLN task lends itself well to visual and textual co-grounding, based on information reused from previously-executed state trajectories.

Learning robot skills. Recent work in robot learning considers the task of learning behaviour primitives (or *skills*), for improved generalisability to new tasks. Sermanet et al. [255] provide an algorithm that enables an agent to learn manipulation tasks (e.g., pouring) from multi-perspective human demonstrations and to directly imitate human pose. The authors achieve this by learning an embedding space, using triplet loss – organises the latent space according to groups of similar observations. The intuition is that various frames that are taken from expert task demonstrations are projected into a lower-dimension embedding space representation. Within the representation, the model is encouraged to repel examples that are different (negative example) from a given sample (anchor) and to attract examples that are similar (positive example). Whereas similar-looking observations are clustered in the learned representation, there is no explicit constraint in their objective for making the learned space reusable or versatile; moreover, the

representation only captures behaviours from one modality (visual context), without jointly representing other modalities (textual or navigational context), as VLN would require. Moreover, their model requires that the observation sequences that are projected into the same representation sub-space also have ground-truth time-synchronisation, which is impractical in the context of VLN. Hausman et al. [112] provide an algorithm for learning an embedding space that represents distributions of solutions for a set of robot manipulation tasks; this would enable more efficient task transfer, as an agent would be able to solve a control problem in the more compact and information-dense embedding space as opposed to exploring in the raw action space. The authors discuss their contribution in the context of hierarchical (i.e., multi-task), entropy-regularised, off-policy reinforcement learning. The focus of this work is primarily on encouraging an agent to implicitly represent a variety of solutions for the same or similar tasks and to also learn across this distribution of solutions. However, the tasks that are highlighted in this work are quite simple and are enumerated by simple one-hot encodings of numerical task IDs, whereas VLN tasks are expressed through natural language instruction, which are significantly more complex and ambiguous. Moreover, the control tasks discussed in this work are only in the context of robot manipulation, as opposed to navigation trajectory-optimisation as in VLN.

Deep RL for natural language navigation. Deep Reinforcement Learning is often used in Vision-and-Language Navigation tasks. Mei et al. [187], Mirowski et al. [194] proposed a sequence-to-sequence model to map the language to navigation actions. Misra et al. [195] formulate navigation as a sequential decision-making process, in a contextual bandit setting. In the same environment, Xiong et al. [312] propose a scheduled training mechanism which yields more efficient exploration and achieves better results. Fried et al. [95] proposed a speaker-follower model, where a speaker model encodes language instructions into action sequences while a follower model estimates the probability of a language instruction given the action sequence. This approach has been optimized to increase success rate but this is achieved at the cost of significantly longer trajectory lengths due to the exploration of possible candidate paths. For the task we are working with, most approaches proposed so far are based on supervised learning Fried et al. [95]. While these approaches achieve very good success rates, they have the issue of exorbitant trajectory lengths. We believe formulating our task as a motion-planning under uncertainty problem with multi-modal reward-shapping that penalizes large trajectory lengths can counter this superfluous exploration. Wang et al. [299] proposed planned-ahead hybrid reinforcement learning model that combines model-free and model-based reinforcement learning to bring together the best of both worlds. Our approach, on the other hand, is completely model-free and is based on the popular on-policy policy-gradient algorithm A3C [197]. While Chaplot et al. [45] work with a similar problem in a synthetic environment and give their agent a positive reward only upon successful completion of task (a very sparse reward), this quickly becomes impractical for a large state-space. Robot navigation, conditioned on text-based instructions, has been the focus of some recent work [45, 68, 195], but only in simulated environments. Our task pertains to real-world environments. Unlike Chaplot et al. [45], Misra et al. [195], we do not rely on templated queries. In contrast to Das et al. [68], Wu et al. [308], our method is applicable to environments where only adjacent nodes on navigation graph are seen (as opposed to having knowledge of the entire map).

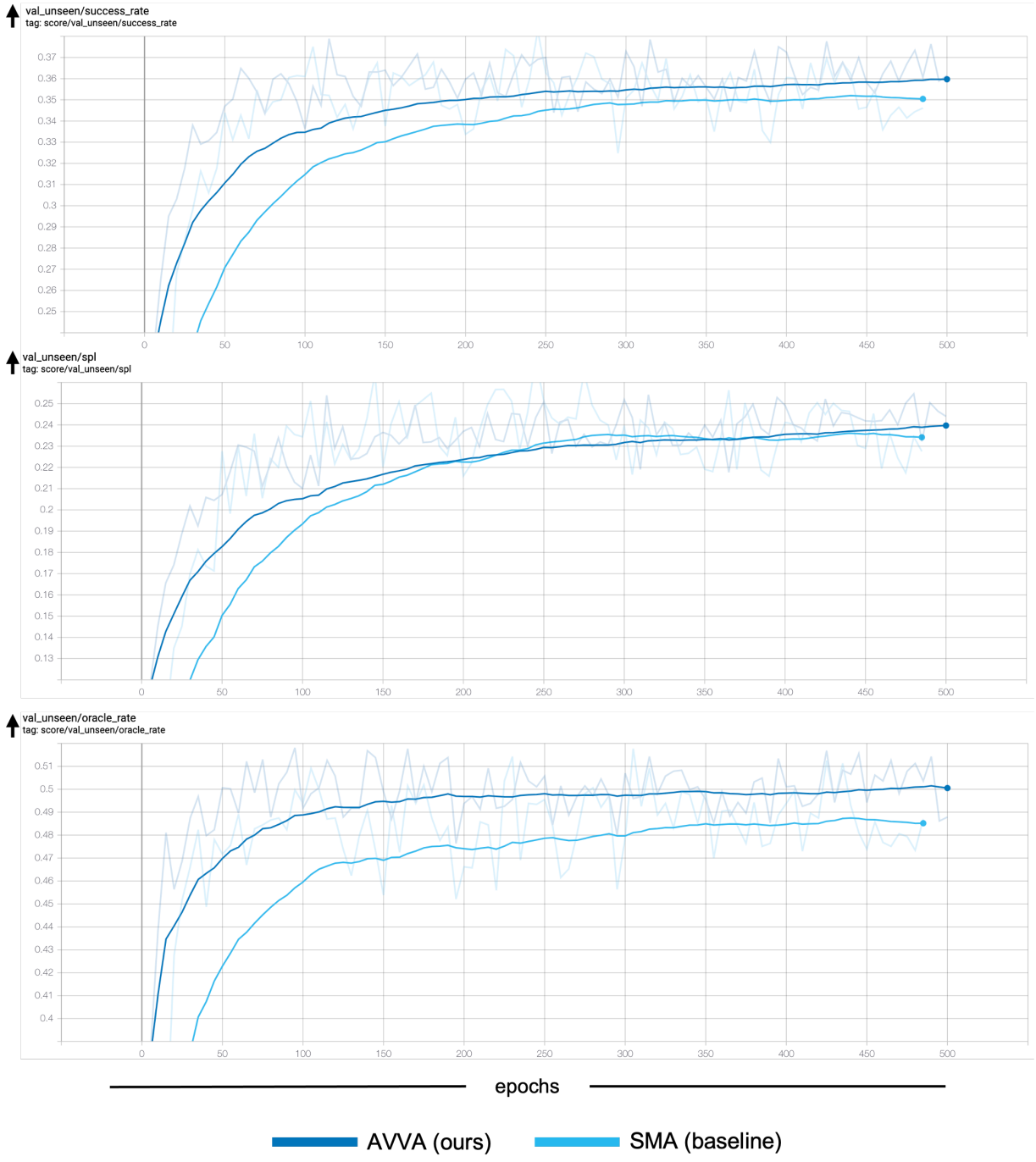


Figure 5.9: Learning with primitives: AVVA-agent main experimental results (1/2).

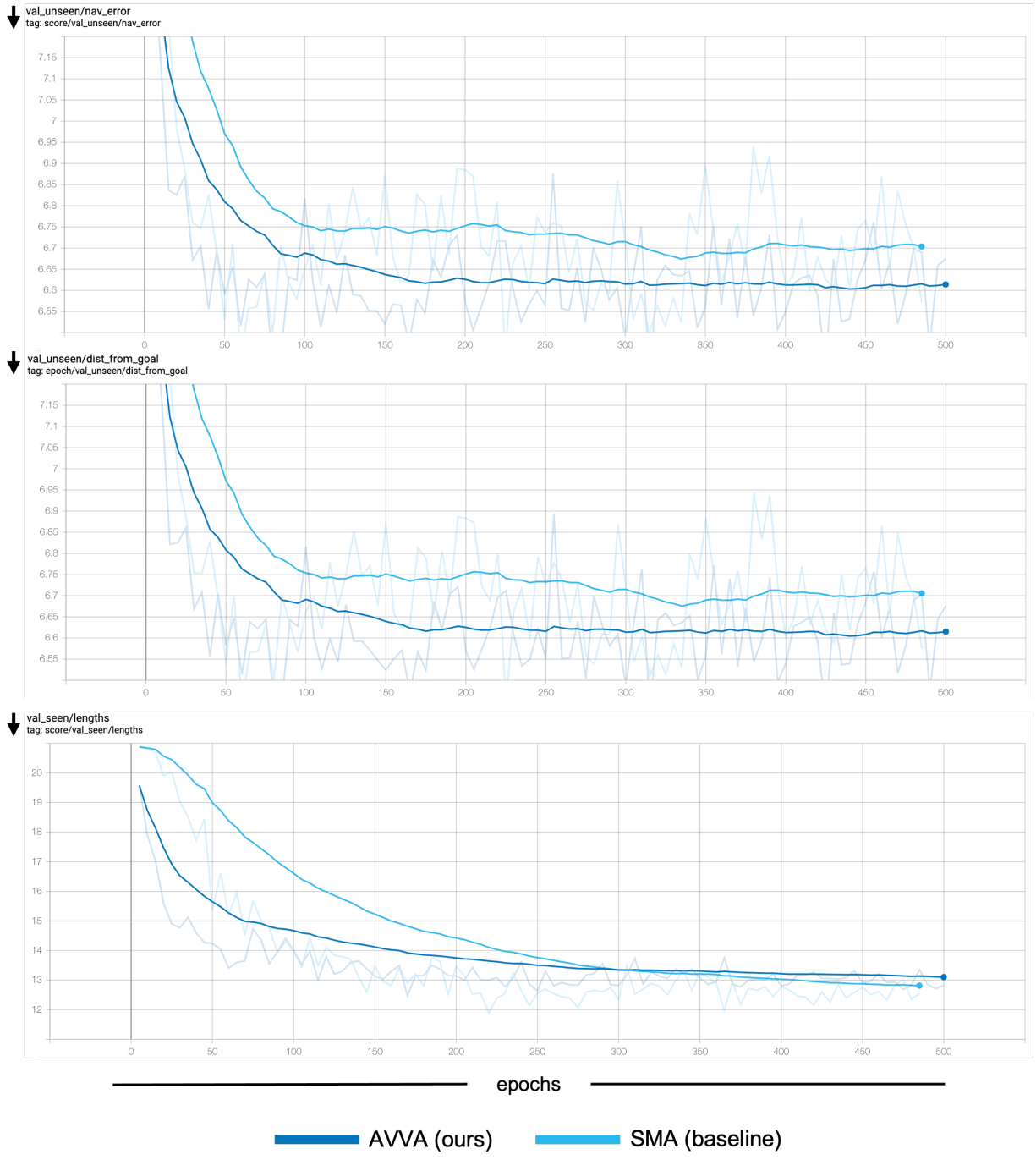


Figure 5.10: Learning with primitives: AVVA-agent main experimental results (2/2).

Table 5.6: Results of noise-injection experiments with the Self-Monitoring Agent (SMA; [181]) baseline, on the VLN-R2R task ([9]); the goal is to understand the unimodal dependence and robustness characteristics of the class of agents that perform multimodal alignment and progress-monitoring, through cross-modal attention.

Standard Baselines											
		VALIDATION-SEEN					VALIDATION-UNSEEN				
Method	**Noise Rate**	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)
Random	0.0	1.58	0.67	16.86	6.75	6.94	2.04	0.87	15.20	6.64	0.0
RPA [9]	0.0	1.58	0.67	16.86	6.75	6.94	2.04	0.87	15.20	6.64	0.0
SFA [95]	0.0	1.58	0.67	16.86	6.75	6.94	2.04	0.87	15.20	6.64	0.0
SMA [181]	0.0	1.58	0.67	16.86	6.75	6.94	2.04	0.87	15.20	6.64	0.0
Visual Noise-injection (*random*)											
		VALIDATION-SEEN					VALIDATION-UNSEEN				
Method	**Noise Rate**	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)
SMA (*ours*)	10%	0.5814	0.6725	0.5228	4.10	12.45	0.3883	0.5053	0.2859	6.16	15.23
SMA (*ours*)	50%	0.3392	0.4196	0.2840	6.43	13.69	0.2205	0.3044	0.16	7.61	15.46
Visual Noise-injection (*average*)											
		VALIDATION-SEEN					VALIDATION-UNSEEN				
Method	**Noise Rate**	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)
SMA (*ours*)	10%	0.6402	0.7390	0.56	3.52	12.71	0.4253	0.5394	0.3	5.97	16.03
SMA (*ours*)	50%	0.5059	0.6686	0.37	4.99	16.21	0.3580	0.5095	0.2251	6.68	17.47
SMA (*ours*)	100%	0.2666	0.5009	0.135	8.11	20.011	0.2337	0.4525	0.122	8.23	20
Structured region-substitution											
		VALIDATION-SEEN					VALIDATION-UNSEEN				
Method	**Noise Rate**	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)
SMA (*ours*)	∼10%	0.6422	0.7333	0.582	3.71	11.97	0.4282	0.54	0.31	5.94	15.49
SMA (*ours*)	∼25%	0.6294	0.7353	0.5667	3.76	12.07	0.4125	0.5262	0.2966	5.98	15.56
SMA (*ours*)	∼50%	0.598	0.7019	0.54	4.18	12.03	0.404	0.52	0.293	6.187	15.64
Navigable action noise											
		VALIDATION-SEEN					VALIDATION-UNSEEN				
Method	**Noise Rate**	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)	SR (↑)	OSR (↑)	SPL (↑)	NE (↓)	TL (↓)
SMA (*ours*)	∼5%	0.6529	0.7422	0.59	3.59	11.94	0.4274	0.54	0.314	5.88	15.39
SMA (*ours*)	∼15%	0.6471	0.7373	0.585	3.7	11.89	0.4168	0.5334	0.3068	5.98	15.41
SMA (*ours*)	∼30%	0.6275	0.7206	0.5677	3.89	12.089	0.4032	0.5126	0.295	6.24	15.45

5.4 Conclusion

This chapter covered the use of priors, primitives, and skills, in the context of robot learning for navigation tasks. We were reminded that various approaches pursue solutions to the challenges in goal-directed and instruction-following tasks, such as: partial observability over the state space, label-independent variation in visual scene rendering, sparse reward structures over the span of an episode, and biased coverage in the provided expert demonstrations. Despite some recent advances—e.g., through pragmatic inference, back-translation, and progress-monitoring—methods still struggle to ground agents' actions to domain invariant context and to effectively align information from multiple modalities.

In the first section, we introduce the use of knowledge-driven scene priors for semantic audio-visual embodied navigation: we combine semantic information from our novel knowledge graph that encodes object-region relations, spatial knowledge from dual Graph Convolutional Networks, and background knowledge from a series of pre-training tasks—all within a reinforcement learning framework for audio-visual navigation. We define a new audio-visual navigation sub-task, where agents are evaluated on novel sounding objects, as opposed to unheard clips of known objects; and we show improvements over strong baselines in generalisation to unseen regions and novel sounding objects, within the Habitat-Matterport3D simulation environment, under the SoundSpaces task. We recognise future improvements of our work, e.g., in the selection of the knowledge resource used for encouraging scene priors in the semantic audio-visual navigation task: we would consider constructing a knowledge resource that characterises sound-object relations, more befitting of pre-training the acoustic GCN stream.

In the second section, we propose a framework for distilling a navigation agent's experience into a representation of reusable maneuvers, which is learned alongside the downstream navigation tasks, where the high-frequency and label-independent variation in the instructions and visual context are reduced. Our approach encourages the agent to generalise reusable navigation maneuvers on the basis of similarities across high-level textual instructions, conditioned on past actions, in order to leverage experience in executing familiar sub-commands from new instructions. We achieve this association through a method called auxiliary variable variational approximation, which allows us to introduce a latent variable for estimating the conditional posterior distribution over all observations and executed trajectories (otherwise intractable for most distributions of interest). The agent then learns how to compose samples from this skill space and couple them with conventional multimodal co-grounding mechanisms, through policy refinement, for improved generalisability and downstream performance. e perform many experiments to analyse the visual and textual modalities, individually, and we show our full model's performance on the overall VLN task, against strong baselines. We reserve additional language modality exploration (e.g., hierarchical co-attention) for future work. We would also explore different policy network structures (e.g., mixture of experts), possibly enabling generalisation to other embodied vision-language planning tasks.

Chapter 6

Learning with Distribution-awareness

for grounded prediction from priors

6.1 Motivation

The availability of expert demonstrations and simulation environments has enabled recent advances in autonomous driving, where agents are trained to forecast the future actions of other agents on the road and/or to generate plans concerning their own behaviour, conditioned on their scene context. However, due to partial observability over the goals and intentions of other agents, as well as the visual complexity in these dynamical scenes, directly querying the posterior distribution over future ego-agent trajectories remains a challenging problem. Conventionally, models have relied on imitation-based objectives, in order to inherit the experience captured by expert demonstrations. However, exclusively relying on imitation-based objectives can limit agents' generalisability to novel scenarios that are outside the support of the training distribution. Furthermore, handling the training instances as i.i.d. samples can leave the model sensitive to small variations across driving scenes and can, therefore, lead to causal confusion in the downstream predictions.

This challenge of generalisability has thus became a common theme: *How can we effectively utilise the expert's prior experience (e.g., in the form of expert demonstrations), while also achieving generalisability to novel scenarios?* Some recent works from the trajectory forecasting community formulate a dual-objective optimisation, coupling an imitation objective with a likelihood density estimation term [219, 237, 239], arguing that the two ideals of using prior expert experience and generalising can be unified. A common issue with this formulation is that models are incentivised to *trade-off* the two objectives, rather than inherit their individual benefits. Samples from the likelihood density may not be sufficiently diverse, if the expert demonstrations did not provide sufficient coverage over the modes in the distribution over all possible predictions. Furthermore, predictions may not be admissible, discussed by Park et al. [219], without some bias to adhere to, e.g., known physical constraints, as in *Verlet integration* [290] or classical control.

In Section 6.3, we propose a model that fully synthesises multiple input signals from the multimodal world—the environment's scene context and interactions between multiple surrounding agents—to best

model all diverse and admissible trajectories. We offer new metrics to evaluate the diversity of trajectory predictions, while ensuring admissibility of each trajectory. Based on our new metrics as well as those used in prior work, we compare our model with strong baselines and ablations across two datasets and show a 35% performance-improvement over the state-of-the-art.

In Section 6.3, we draw inspiration from the trajectory forecasting community, and we frame a learning-to-drive task as goal prediction, model-based planning, and trajectory refinement. First, we define a series of module primitives, based on insights about the decomposable nature of the environment. We thereby assign modules with the aforementioned specialised roles, improving both the tractability of their respective tasks and their complementarity towards the shared downstream task. Next, we pursue model generalisability by coupling an imitation prior objective with a goal likelihood term, enabling the agent to leverage expert knowledge, while modelling more diverse modes in the underlying distribution over all trajectory futures. Next, we ground candidate trajectory predictions on conformant model-based vehicle kinematics, while learning to prune the predictions that are spurious. Finally, we provide both qualitative and quantitative analysis on multiple benchmarks under the CARLA simulation environment, achieving new state-of-the-art results.

6.2 Diverse and Admissible Trajectory Forecasting through Multimodal Context Understanding

Trajectory forecasting is an important problem in autonomous driving scenarios, where an autonomous vehicle must anticipate the behaviour of other surrounding agents (e.g., vehicles and pedestrians), within a dynamically-changing environment, in order to plan its own actions accordingly. However, since none of the goals, contexts, or interactions are directly observed, predicting future trajectories is a challenging problem [239, 280, 331]. It necessitates both the estimation of plausible agent actions based on observable environmental features (e.g., road structures, agent interactions) as well as the simulation of agents' hypothetical future trajectories toward their intended goals. In realistic embodied environments, there are multiple plausible sequences of actions that an agent can take to reach its intended goals. However, each trajectory must obey physical constraints (e.g., Newton's laws) and stay in the statistically plausible locations in the environment (i.e., the drivable areas). In this paper, we refer to these attributes as *diverse* and *admissible* trajectories, respectively, and illustrate some examples in Fig. 6.1. Achieving diverse and admissible trajectory forecasting for autonomous driving allows each agent to make the best predictions, by taking into account all valid actions that other agents could take. In addition, it allows each agent to assess the surrounding situation to ensure safety and prevent accidents.

To predict a diverse set of admissible trajectories, each agent must understand its *multimodal* environment, consisting of the scene context as well as interactions between multiple surrounding agents. While the scene context gives direct information about regions an agent can drive in, observation of other agents' trajectories can provide additional environmental context. For example, conceptual constraints over the agent's motion (e.g., traffic laws, road etiquette) may be inferred from the motion of the surrounding agents. Therefore, the model's ability to extract and meaningfully represent multimodal cues is crucial.

Concurrently, another challenging aspect of trajectory forecasting lies in encouraging models to make diverse predictions about future trajectories. However, due to high-costs in data collection, most public datasets are not explicitly annotated for multiple future trajectories [44, 149, 184]. Vanilla predictive models that fit future trajectories based only on the existing annotations would severely underestimate

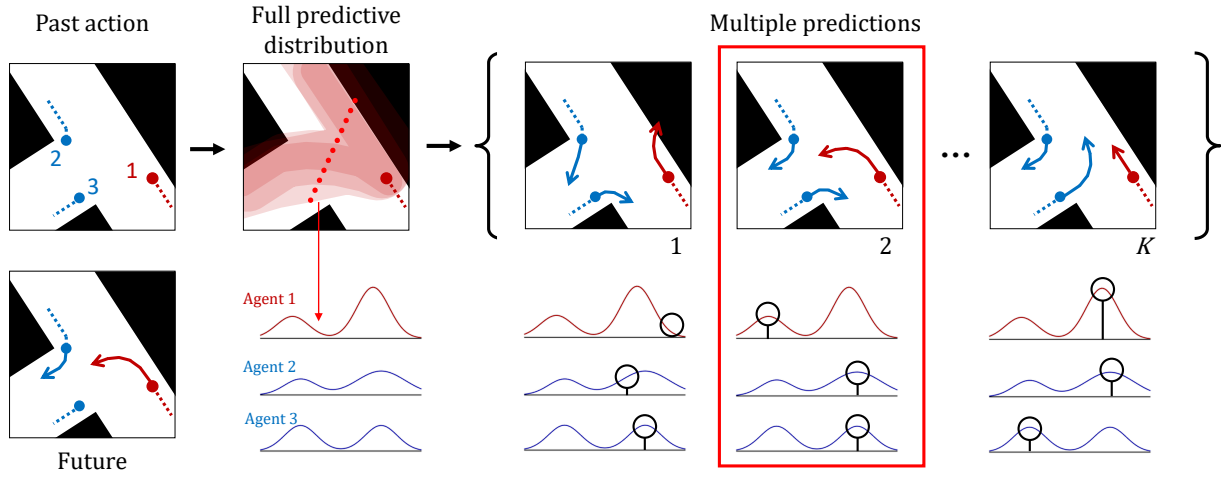


Figure 6.1: Diverse and admissible trajectory forecasting. Based on the existing context, there can be multiple valid hypothetical futures. Therefore, the predicted future trajectory distribution should have multiple modes representing multiple plausible goals (*diversity*) while at the same time assigning low density to the implausible trajectories that either conflict with the other agents or are outside valid drivable areas (*admissibility*).

the diversity of all possible trajectories. In addition, measuring the quality of predictions using existing annotation-based measures (e.g., displacement errors [246]) does not faithfully score diverse and admissible trajectory predictions.

As a step towards multimodal context understanding for diverse trajectory forecasting, our contribution in this section is *four*-fold.

1. We propose a model that addresses the lack of diversity and admissibility for trajectory forecasting through the understanding of the multimodal environmental context. As illustrated in Fig. 6.2, our approach explicitly models agent-to-agent and agent-to-scene interactions through "self-attention" [288] among multiple agent trajectory encodings, and a conditional trajectory-aware "visual attention" [313] over the map, respectively. Together with a constrained flow-based decoding trained with symmetric cross-entropy [237], this allows our model to generate diverse and admissible trajectory candidates by fully integrating all environmental contexts.

2. We propose a new approximation of the true trajectory distribution based on a differentiable drivable-area map. This approximation is used when evaluating our posterior likelihood. Previous approximation methods [237] utilise ground-truth (GT) trajectories to model the real distribution. However, only one GT annotation is available per agent. Our approximation method does not rely on GT samples and empirically facilitates greater diversity in the predicted trajectories while ensuring admissibility.

3. We propose a new metric, Drivable Area Occupancy (DAO), to evaluate the diversity of the trajectory predictions while ensuring admissibility. This new metric utilises the drivable-area map, without requiring multiple annotations of trajectory futures. We couple this new metric with standard metrics from prior art, such as Average Displacement Error (ADE) and Final Displacement

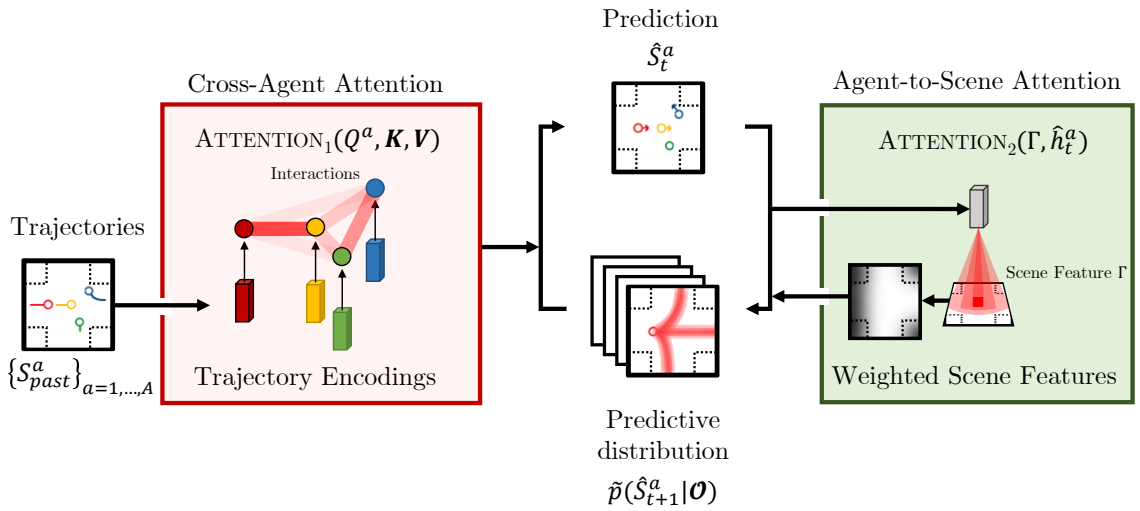


Figure 6.2: Overview of our multimodal attention approach. Best viewed in color. The cross-agent attention module (left) generates an attention map, based on the encoded trajectories of nearby agents. The agent-to-scene attention model (right) generates an attention map over the scene, based on the posterior approximations.

Error (FDE), to compare our model with existing baselines.

4. We provide a programmatic set of procedures to convert the NUSCENES [37] tracking data to a new dataset for trajectory forecasting. The procedure includes trajectory association, smoothing, imputation, and generation of the drivable-area features. These features are used for approximation of the real trajectory distribution and for calculating our new metrics. We set new state-of-the-art performance for multi-agent trajectory forecasting, wherein our model enjoys a 35% performance-improvement over the current baselines.

6.2.1 Problem Formulation

We define the terminology that constitutes our problem. An *agent* is a dynamic on-road object that is represented as a sequence of 2D coordinates, i.e., a spatial position over time. We denote the position for agent a at time t as $S_t^a \in \mathbb{R}^2$. By writing $S_{t_1:t_2}^a$, we represent the sequence of its positions, between t_1 and t_2. $\boldsymbol{S}^a$ (bold) to denote full sequence of positions for agent a. We set $t = 0$ as *present*, $t \leq 0$ as *past*, and $t > 0$ as *prediction* or simply, *pred*. We often split the sequence into two parts, with respect to the *past* and *pred* sub-sequences: we denote these as $\boldsymbol{S}_{past}^a$ and $\boldsymbol{S}_{pred}^a$, respectively. A *scene* is a high-dimensional structured data that describes the present environmental context around the agent. For this, we utilise a bird's eye view (BEV) array, denoted $\boldsymbol{\Phi} \in \mathbb{R}^{H \times W \times C}$, where H and W are the sizes of field around the agent and C is the channel size of the scene, where each channel consists of distinct information such as the drivable area, position, and distance encodings.

Combining the *scene* and all *agent* trajectories yields an *episode*. In an episode $\mathcal{X}$, there is a variable A number of agents, each of which plays for different time periods from between the variable *start time* t_s^a and *final time* t_f^a. As a result, the episode is the set $\{\boldsymbol{S}^1, \boldsymbol{S}^2, ..., \boldsymbol{S}^A, \boldsymbol{\Phi}\}$ where $\boldsymbol{S}^a \equiv S_{t_s^a:t_f^a}^a$. In the combined setting, we often use bold $\boldsymbol{S} \equiv \{\boldsymbol{S}^1, \boldsymbol{S}^2, ..., \boldsymbol{S}^A\}$ to denote the agents subset of the episode $\mathcal{X}$

and write $\boldsymbol{S}_{\text{past}}$ or $\boldsymbol{S}_{\text{pred}}$ to represent the set of past or predicted segments of A agents. Since $\boldsymbol{S}_{\text{past}}$ and $\boldsymbol{\Phi}$ serve as the observed information cue used for the prediction, they are often called *observation* simply being denoted as $\mathcal{O} \equiv \{\boldsymbol{S}_{\text{past}}, \boldsymbol{\Phi}\}$. Finally, we may add the subscript $n = 1, 2, ..., N$ to all the notations, such as $\mathcal{X}_n$, $\mathcal{O}_n$, $\boldsymbol{\Phi}_n$, $S^a_{t,n}$, $S^a_{t_1:t_2,n}$, $\boldsymbol{S}^a_n$, or $\boldsymbol{S}_n$ to distinguish the information from different episodes.

We define *diversity* to be the level of coverage in a model's predictions, across modes in a distribution representing all possible future trajectories. We denote the model distribution as $q(\boldsymbol{S}_{\text{pred}}|\mathcal{O})$ and want the model to generate K candidates or *hypotheses*, denoted as $\hat{\boldsymbol{S}}_{pred} \equiv \{\hat{\boldsymbol{S}}^1_{pred}, \hat{\boldsymbol{S}}^2_{pred}, ..., \hat{\boldsymbol{S}}^K_{pred}\}$. We interpret $\hat{\boldsymbol{S}}_{pred}$ as a set of independent hypotheses that might have happened, given the same *observation*. Instead of generating samples from one mode, which we refer to as *perturbation*, we expect to build a model that generates multiple hypotheses that cover multiple modes.

Finally, we acknowledge that encouraging a model's predictions to be diverse, alone, is not sufficient for accurate and safe output; the model predictions should lie in the support of the real future trajectory distribution p, i.e., predictions should be $admissible$. Given the observation $\mathcal{O}$, it is futile to predict samples around regions that are physically and statistically implausible to reach. In conclusion, our task is *diverse and admissible multi-agent motion forecasting* by modelling multiple modes in the posterior distribution over the *pred* trajectories, given the observation: $p(\boldsymbol{S}_{\text{pred}}|\mathcal{O})$.

6.2.2 Admissible Prediction from Diverse Modes

We hypothesise that future trajectories of human drivers should follow distributions of multiple modes conditioned on the scene context and social behaviours of agents. Therefore, we design our model to explicitly capture both agent-to-scene interactions and cross-agent interactions with respect to each agent of interest. Through our objective function, we explicitly encourage the model to learn a distribution with multiple modes by taking into account past trajectories and attended scene context.

As illustrated in Fig. 6.3, our model consists of an encoder-decoder architecture. The encoder has two modules to capture cross-agent interactions and existing trajectories. The decoder has three modules: the local scene extractor, the agent-to-scene interaction module, and the flow-based decoding module. Please refer to Fig. 6.4 for a detailed illustration of our main proposed modules.

The *encoder* extracts past trajectory encoding for each agent, then calculates and fuses the interaction features among the agents. Given a set of past trajectories $\boldsymbol{S}_{\text{past}}$ in an observation $\mathcal{O}$, we encode each agent's past trajectory $S^a_{\text{past}} \in \boldsymbol{S}_{\text{past}}$ by feeding it to the agent trajectory encoding module. The module utilises a recurrent neural network (RNN) to summarise the past trajectory. It iterates through the past trajectory with Eq. (6.1) and its final output h^a_0 (at *present* $t = 0$) is utilised as the agent embedding. Collecting the embeddings for all agents, we get $\boldsymbol{h}_0 \equiv \{h^1_0, h^2_0, ..., h^A_0\}$. We then pass $\boldsymbol{h}_0$ to the cross-agent interaction module, depicted in Fig. 6.4(a), which uses *self-attention* [288] to generate a cross-agent representation. We linearly transform each agent embedding to get a query-key-value triple, (Q^a, K^a, V^a). Next, we calculate the interaction features through self-attention with $\text{ATTENTION}_1(Q^a, \boldsymbol{K}, \boldsymbol{V})$, where $\boldsymbol{K}, \boldsymbol{V} \equiv \{K^1, ..., K^a\}, \{V^1, ..., V^a\}$. Finally, the fused agent encoding $\tilde{\boldsymbol{h}} \equiv \{\tilde{h}^1, \tilde{h}^2, ..., \tilde{h}^A\}$ is calculated by adding the features to each agent embedding (see Eq. (6.2) and Fig. 6.4(b)).

$$h^a_t = \text{RNN}_1(S^a_{t-1}, h^a_{t-1}) \tag{6.1}$$

$$\tilde{h}^a = h^a_0 + \text{ATTENTION}_1(Q^a, \boldsymbol{K}, \boldsymbol{V}) \tag{6.2}$$

The *decoder* takes the final encodings $\tilde{\boldsymbol{h}}$ and the scene context $\boldsymbol{\Phi}$ as inputs. We first extract the scene feature through a *ConvNet*, $\boldsymbol{\Gamma} = \text{CNN}(\boldsymbol{\Phi})$. The decoder then generates the future position $\hat{S}^a_t$, in an auto-

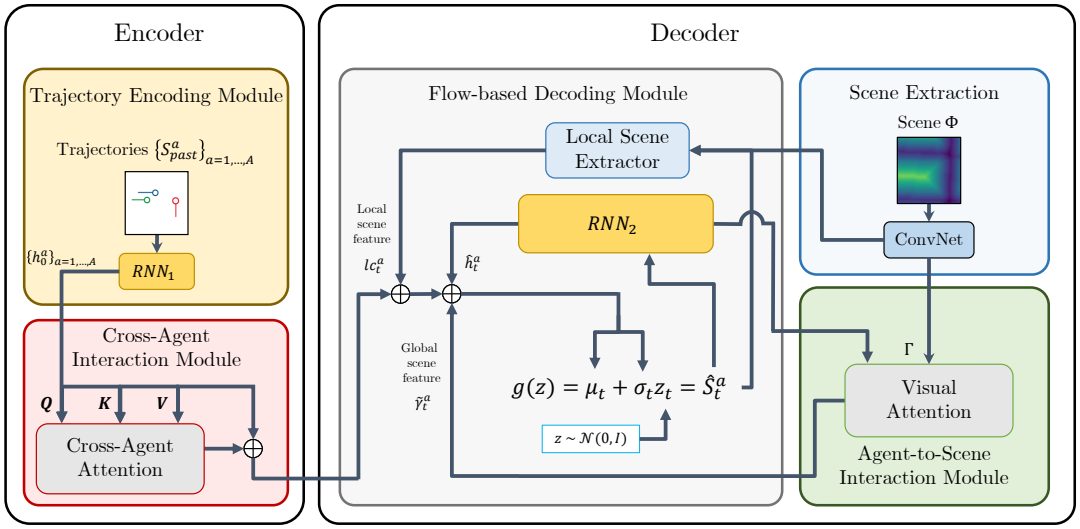


Figure 6.3: Model Architecture. The model consists of an encoder-decoder architecture: the encoder takes as past agent trajectories and calculates cross-agent attention, and the flow-based decoder predicts future trajectories by attending scene contexts for each decoding step.

regressive manner, while referring to both the local scene context γ^a_t and the global scene context $\tilde{\gamma}^a_t$ from the agent-to-scene interaction module. The local scene feature γ^a_t is gathered using bilinear interpolation on 2×2 crop of $\mathbf{\Gamma}$ corresponding to the physical position $\hat{S}^a_{t-1}$. Then, the feature is concatenated with the encoding $\tilde{h}^a$ and processed thorough fully-connected layers to make the "local context" lc^a_t. We call this part the local scene extractor. The global scene feature $\tilde{\gamma}^a_t$ is calculated using *visual-attention* [313] to generate weighted scene features, as shown in Fig. 6.4(c). To calculate the attention, we first make the encoding of the previous outputs $\hat{S}^a_{1:t-1}$, using a RNN in Eq. (6.3), whose output—$\hat{h}^a_t$—is used to calculate the pixel-wise attention at each decoding step, for each agent; the global scene feature $\tilde{\gamma}^a_t$ (1D vector) is gathered by pooling (pixel-wise sum) the attended feature map as described in Eq. (6.4) and Fig. 6.4(d). Finally, $\tilde{\gamma}^a_t$, $\hat{h}^a_t$, and lc^a_t are concatenated to make the "global context" gc^a_t in Eq. (6.5).

$$\hat{h}^a_t = \text{RNN}_2(\hat{S}^a_{1:t-1}, \hat{h}^a_{t-1}) \tag{6.3}$$

$$\tilde{\gamma}^a_t = \textsc{Pool}(\mathbf{\Gamma} \odot \textsc{Attention}_2(\hat{h}^a_t, \mathbf{\Gamma})) \tag{6.4}$$

$$gc^a_t = \textsc{Concat}(\tilde{\gamma}^a_t, \hat{h}^a_t, lc^a_t) \tag{6.5}$$

The flow-based decoding module generates the future position $\hat{S}^a_t$. The module utilises *Normalizing Flow* [235], a generative modelling method based on a bijective and differentiable mapping. In particular, we choose an auto-regressive design [141, 237, 239]. We use fully-connected layers to project the global context gc^a_t down to a 6-dimensional vector, and we split it into a vector $\hat{\mu}_t \in \mathbb{R}^2$ and a matrix $\hat{\sigma}_t \in \mathbb{R}^{2\times 2}$. Next, we transform a standard Gaussian sample $z_t \sim \mathcal{N}(\mathbf{0}, I) \in \mathbb{R}^2$, by the bijective and differentiable mapping $g(z_t; \mu_t, \sigma_t) = \sigma_t \cdot z_t + \mu_t = \hat{S}^a_t$. The "hats" in $\hat{\mu}_t$ and $\hat{\sigma}_t$ are removed, in order to note that they went through the following details. To ensure the positive definiteness, we apply matrix exponential $\sigma_t = expm(\hat{\sigma}_t)$ using the formula in [24]. Also, to improve the the physical *admissibility* of the prediction, we apply the constraint $\mu_t = \hat{S}^a_{t-1} + \alpha(\hat{S}^a_{t-1} - \hat{S}^a_{t-2}) + \hat{\mu}_t$, where α is a model degradation coefficient. When $\alpha = 1$, the constraint is equivalent to *Verlet integration* [290], used in some previous works [237, 239],

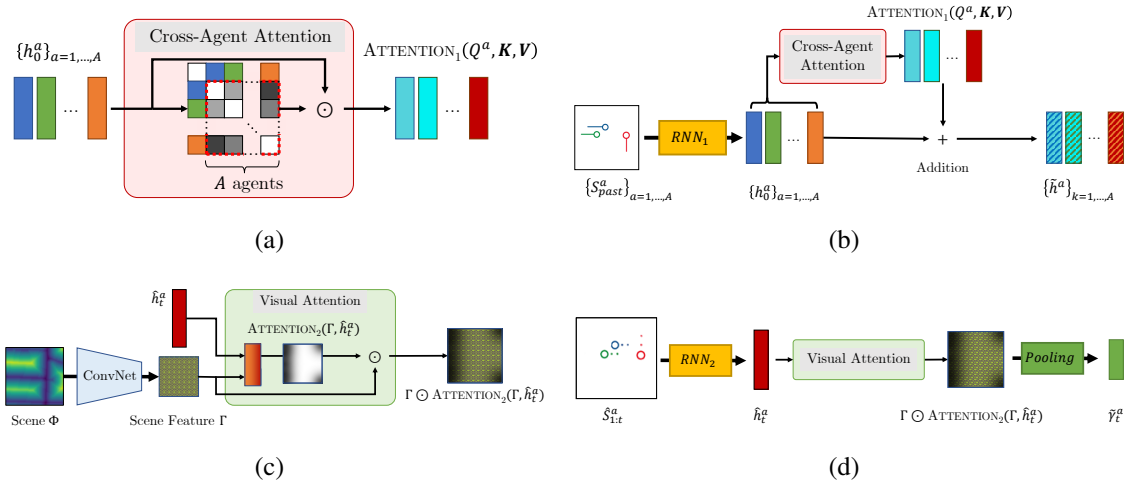


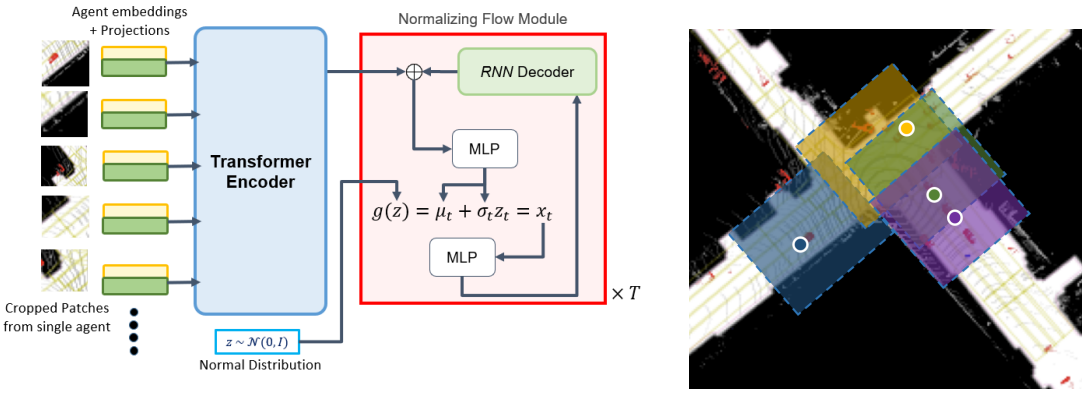
Figure 6.4: (a) Cross-agent attention. Interaction between each agent is modelled using attention, (b) Cross-agent interaction module. Agent trajectory encodings are corrected via cross-agent attention. (c) Visual attention. Agent-specific scene features are calculated using attention. (d) Agent-to-scene interaction module. Pooled vectors are retrieved from pooling layer after visual attention.

which gives the a perfect constant velocity (CV) prior to the model. However, we found empirically that, the model easily overfits to the dataset when the the perfect CV prior is used, and perturbing the CV prior model with α prevents overfitting. We use $\alpha = 0.5$ in our model. Iterating the auto-regressive decoding procedure, we get the future trajectory prediction $\hat{S}^a_{\text{pred}}$ for each agent. Note that by sampling multiple instances of z_{pred}, we can generate the multiple future $\hat{S}^{k,a}_{\text{pred}}$.

Drivable Area Map and Approximating P

In this work, we generate a binary mask feature of size $\mathbb{R}^{H \times W}$ that denotes the drivable spaces around the agents. We call the feature *drivable area map* and utilise it for three different purposes: 1) deriving the approximated true trajectory distribution $\tilde{p}$, 2) calculating the diversity and admissibility measures, and 3) building the scene context input $\mathbf{\Phi}$ for the model.

Particularly, $\tilde{p}$ is a key component for the evaluation of $H(q, p)$ in our training objective, Eq. (6.7). Since $H(q, p)$ penalises the predicted trajectories with respect to the real distribution, the approximation should not underestimate some region of the real distribution, or diversity in the prediction could be erroneously penalised. Previous works on deriving $\tilde{p}$ utilised the ground-truth (GT) trajectories to model the true distribution p [237]. However, there is often only one GT annotation available in datasets and the approximation based on the GT might severely assign low probability around some region in p. To cope with such problem in the previous methods, we propose a new method to derive $\tilde{p}$ using the drivable area. Our $\tilde{p}$ is defined based on the assumptions that every location on the drivable-area is equally probable for future trajectories to appear in and that the locations on non-drivable area are increasingly less probable, proportional to the distance from the drivable area. To derive it, we first apply the distance transform on the drivable area map, to encode the distance on each non-drivable location. Lastly, we apply softmax over the entire map to constitute it as a probability distribution. The visualisation of the $\tilde{p}$ are available in Fig. 6.8. Procedures regarding the diversity and admissibility measure will be discussed in Section 6.2.5.



(a) (b)

Figure 6.5: (a) Model architecture with Agent pose embeddings, cropped image and positional embeddings fused for input to the transformer encoder and Flow based decoding for producing T future poses. This reduced architecture can be useful for Trajectory prediction for embedded platforms in Robotic applications. (b) Depiction of Patch croppings produced from the BEV image. The different colours indicate different Agents in the scene. And the coloured area is a fixed $K \times K$ pixel size for each such crops. These patches are then again cropped into 16×16 patches and linearly projected to produce projections at the input of the transformer model.

6.2.3 Social Encoding through Local Self-attention

To encode the social factor between A agents, we combine their individual past-trajectory encodings, through consecutive additive and multiplicative fusion [173], to generate A embeddings. For each of these agent embeddings, t past time-step poses are raised to N-dimensional vectors and are combined with patch and positional embeddings, for each time-step. In this way, we generate an $N \times t$-dimensional input for the transformer model.

Let $\boldsymbol{S} \equiv \{S^1, S^2, \ldots, S^A\}$ denote the set of agent trajectories for A agents in a given scene, with S^a being the concatenation of past trajectory segment S^a_{past} and ground-truth future trajectory segment S^a_{future}. Here, a single step is indicated by $S^a_t \in \mathbb{R}^2$, for agent a and time-step t. Thus, $\boldsymbol{S}_{\text{past}}$ is the collection of past trajectory segments for all agents, and the observation set of all 3-second past trajectories is denoted by $\mathcal{O} \equiv \{\boldsymbol{S}_{\text{past}}, \boldsymbol{\Phi}, \boldsymbol{\phi}\}$, where Φ is a scene embedding [219] and ϕ is the positional embedding [288]. We want to model the posterior distribution over future trajectories, for all agents in the scene snapshot, $q(\boldsymbol{S}_{\text{pred}}|\mathcal{O})$.

Past agent trajectories are projected to a (higher) d-dimensional space, in preparation for input to the transformer, i.e., $\mathbf{e}^a_{obs} = \texttt{MLP}_{proj}(\mathrm{S}^a)$. For encoding the contextual information, we extract an m^2 pixel neighbourhood image patch, centered around each vehicle, from the *birds-eye-view* (BEV) map of the scene: $\mathbf{e}^a_{patch}$. The BEV map contains coloured objects and superimposed LiDAR points. Figure 6.5(b) illustrates the agent-wise pixel neighbourhood, which capture local contextual information.

We perform sine-distance positional encoding of the map representation, which is then added to each agent's flattened sequence of patch vectors. We then calculate a fused representation, combining the local environment information and state history, as a *Hadamard* product between each agent's past trajectory

embedding and its corresponding scene context [173]:

$$\mathbf{e}^a_{fused} = \mathbf{e}^a_{obs} \odot [\texttt{ENC}_{pos}(\mathbf{e}^a_{map}) + \texttt{FLATTEN}(\mathbf{e}^a_{patch})]$$

These fused representations are fed to a standard transformer encoder, which contains alternating layers of multi-headed self-attention and MLP blocks. The output is a set of latent codes – one for each agent: $\mathbf{c}^a_{latent} = \texttt{ENC}_{tr}(\mathbf{e}^a_{fused})$, with $\mathbf{c}^a_{latent} \in \mathbb{R}^{D\times A}$ with hyperparameter D.

The normalizing-flow-based generative decoder features an implicit auto-regressive design and performs a differentiable and bijective mapping, from the latent codes to the set of agent-wise trajectory predictions [219, 239] (see figure 6.4(a) for illustration): $g_\theta(z_t; \mu_t, \sigma_t) = \sigma_t \cdot z_t + \mu_t = S^a_{pred,t}$, with $z_t \sim \mathcal{N}(\mathbf{0}, I) \in \mathbb{R}^2$. Here, θ is the set of model parameters and $\mu_t \in \mathbb{R}^2$ and $\sigma_t \in \mathbb{R}^{2\times 2}$ are projected parameters. Iterating through time, we get the predictive trajectory S^a_{pred} for each agent. By sampling multiple instances of z_{pred} and mapping them to trajectories, we get various hypotheses of future.

6.2.4 Learning and Optimisation

Our model learns to predict the joint distribution over the future trajectories of the agents present in a given episode. In detail, we focus on predicting the conditional distribution $p(\boldsymbol{S}_{\text{pred}}|\mathcal{O})$ where the future trajectory $\boldsymbol{S}_{\text{pred}}$ depends on the set of observations of the past trajectories and the scene context $\mathcal{O} \equiv \{\boldsymbol{S}_{\text{past}}, \boldsymbol{\Phi}\}$ given an episode. As described in the previous sections, our model utilizes a bijective and differentiable mapping, parameterized by a learnable parameter θ, $S^a_{\text{pred}} = f_\theta(\boldsymbol{z} \sim q_0; \mathcal{O})$ between the future trajectory and a Gaussian prior q_0 to generate and evaluate the future trajectory. Such technique, commonly aliased 'normalizing flow', enables our model not only to generate multiple candidate samples of future, but also to evaluate the ground-truth trajectory according to the predicted distribution q_θ by using the change-of-variable formula in Eq. (6.6).

$$q_\theta(S^a_{\text{pred}}|\mathcal{O}) = q_0(f^{-1}(S^a_{\text{pred}})) \left|\det(\partial S^a_{\text{pred}}/\partial(g^{-1}(S^a_{\text{pred}}))\right|^{-1} \tag{6.6}$$

As a result, our model can simply learn to close the discrepancy between the predicting distribution q_θ and the real world distribution p. In particular, we choose to minimise the combination of forward and reverse cross-entropy $H(p, q)$ and $H(q, p)$, also known as 'symmetric cross-entropy', between the two distributions in Eq. (6.7) by optimising our model parameter θ. Minimising symmetric cross-entropy allows model to learn generating diverse and plausible trajectory, which is mainly used in [237].

$$\min_\theta H(p, q_\theta) + \beta H(q_\theta, p) \tag{6.7}$$

To realise this, we gather the ground-truth trajectories $\boldsymbol{S}$ and scene context $\boldsymbol{\Phi}$ from the dataset $\mathcal{D}$ that we assume to well reflect the real distribution p, then optimise the model parameter θ such that 1) the density of the ground-truth future trajectories on top of the predicted distribution q_θ is maximised and 2) the density of the predicted samples on top of the real distribution p is also maximised as described in Eq.(6.8).

$$\min_\theta \mathbb{E}_{\boldsymbol{S},\boldsymbol{\Phi}\sim\mathcal{D}} \left[\mathbb{E}_{S^a_{\text{pred}}\in\boldsymbol{S}} - \log q_\theta(S^a_{\text{pred}}|\mathcal{O}) + \beta\, \mathbb{E}_{\hat{S}^a_{\text{pred}}\sim q_\theta} - \log p(\hat{S}^a_{\text{pred}}|\mathcal{O}) \right] \tag{6.8}$$

Such symmetric combination of the two cross-entropy guides our model to predict q_θ that covers all plausible modes in the future trajectory while penalising the bad samples that are less likely under the real distribution p. However, one major problem inherent in this setting is that we cannot actually evaluate p

in practice. To cope with the problem, several ways of approximating p by using a separate model $\tilde{p}$ have been suggested so far [237]. In this paper, we propose a new way of modelling $\tilde{p}$ which approximates p using a discrete grid map derived from the differentiable drivable area map in our dataset which considers every drivable region around the ego-vehicle to be equally probable that the future trajectories are placed. Applying bilinear interpolation around each prediction time-step in the generative sample $\hat{S}^a_{\text{pred}}$, we get the evaluation $\tilde{p}(\hat{S}^a_{\text{pred}}|\mathcal{O})$ that is differentiable with respect to the model parameter θ. Our overall loss function is:

$$\frac{1}{\sum_{n=1}^{N} A(n)} \sum_{n=1}^{N} \sum_{a=1}^{A(n)} \Bigg[-\log q_\theta(S^a_{\text{pred},n}|\mathcal{O}_n) + \beta \frac{1}{K} \sum_{k=1}^{K} -\log p(\hat{S}^{a,k}_{\text{pred},n}|\mathcal{O}_n) \Bigg], \tag{6.9}$$

where N is the batch size, $A(n)$ is the number of total agents in nth episode, and K is the number of candidates to sample per agent. Since this objective is fully differentiable with respect to the model parameter θ, we train our model using Adam optimiser [138], a popular variant of the stochastic gradient descent algorithm. We also use adaptive learning rate scheduling and early stopping.

6.2.5 Metrics for Assessing Diversity and Admissibility

We define multiple metrics that provide a thorough interpretation about the behaviour of each model in terms of precision, diversity, and admissibility. For the *i*-th trajectory, we first evaluate a prediction in terms of Euclidean errors: *average displacement error* $\text{ADE}^{(i)} = \frac{1}{T}\sum_{t=1}^{T} \|S^i_t - \hat{S}^i_t\|_2$ and *final displacement error* $\text{FDE}^{(i)} = \|S^i_{t^i_f} - \hat{S}^i_{t^i_f}\|_2$, or ERROR to denote both. To evaluate N predictions (i.e., precision), we use the average and the minimum ERRORs: $\text{AVGERROR} = \frac{1}{N}\sum_{i=1}^{N} \text{ERROR}^{(i)}$ and $\text{MINERROR} = \min\{\text{ERROR}^{(1)}, ..., \text{ERROR}^{(N)}\}$. A large AVGERROR implies that predictions are spread out, and a small MINERROR implies at least one of predictions has high precision. From this observation, we define new evaluation metrics that capture diversity in predictions: the ratio of AVGADE to MINADE and AVGFDE to MINFDE, namely RA and RF. In particular, RF is robust to the variability of magnitude in velocity in predictions because high AVGADE and high MINADE caused by large magnitudes will be offset and only the directional variability will remain. As a result, RF provides a handy tool that can distinguish between predictions with multiple modes (diversity) and predictions with a single mode (perturbation). For deterministic models, RA and RF have a value of 1.

$$\text{Ratio of AVGFDE to MINFDE (RF)} = \frac{\text{AVGFDE}}{\text{MINFDE}} \tag{6.10}$$

$$\text{Drivable Area Occupancy (DAO)} = \frac{\text{count}(\text{traj}_{\text{pix}})}{\text{count}(\text{driv}_{\text{pix}})} \tag{6.11}$$

We also report performance on additional metrics that are designed to capture diversity and admissibility in predictions. We follow [44] in the use of *Drivable Area Count* (DAC), $\text{DAC} = \frac{n-m}{n}$, where m is the number of predictions that go out of the drivable area and n is the total number of predictions. Next, we propose a new metric, *Drivable Area Occupancy* (DAO), which measures the percentage of pixels that predicted trajectories occupy in the drivable area. Shown in Eq. (6.11), $\text{count}(\text{traj}_{\text{pix}})$ is the number of pixels occupied by predictions and $\text{count}(\text{driv}_{\text{pix}})$ is the total number of pixels of the drivable area, both within a pre-defined grid around the ego-vehicle. Due to the nature of DAO and DAC, the number of trajectory hypotheses should be set equally for fair comparison of models.

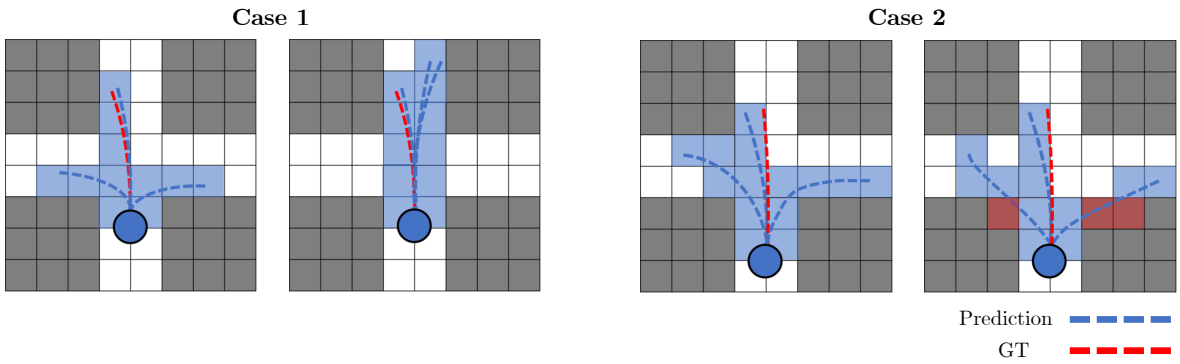


Figure 6.6: We motivate the need for multiple metrics, to assess diversity and admissibility. Case 1: DAO measures are equal, even though predictions have differing regard for the modes in the posterior distribution. Case 2: RF measures are equal, despite differing regard for the cost of leaving the drivable area. In both cases, it is important to distinguish between conditions—we do this by using DAO, RF, and DAC together.

Figure 6.7: Metric quality spectrum. Our newly proposed metrics: RF measures the spread of predictions in Euclidean distance, DAO measures diversity in predictions that are only admissible. DAC measures extreme off-road predictions that defy admissibility.

We use RF, DAO, and DAC to assess the diversity and admissibility of models. Initially, DAO may seem like a reasonable standalone measure of both diversity and admissibility, as it only cares about diversity in a reasonable region of interest. However, DAO itself cannot distinguish between *diversity* (Section 6.2.1) and arbitrary stochasticity in predictions, as illustrated by Case 1 in Fig. 6.6: although DAO measures of both predictions are equal, the causality behind each prediction is different and we must distinguish the two. RF and DAO work in a complementary way and we, therefore, use both for measuring diversity. To assure the *admissibility* of predictions, we use DAC which explicitly counts off-road predictions, as shown by Case 2 in Fig. 6.6. As a result, assessing predictions using DAO along with RF and DAC provides a holistic view of the quantity and the quality of diversity in predictions; the characteristics of each metric is summarised in Fig. 6.7.

For our experiments, we use MINADE and MINFDE to measure *precision*, and use RF, DAC, and DAO to measure both *diversity* and *admissibility*. Due to the nature of DAO, where the denominator in our case is the number of overlapping pixels in a 224×224 grid, we normalise it by multiplying by $10,000$ when reporting results. For the multi-agent experiment (experiment 4), relative improvement (RI) is calculated as we are interested in the relative improvement as the number of agents increases. If not specified, the number of hypotheses are set to 12 and MINFDE is reported for the performance.

6.2.6 Experiments

The primary goal in the following experiments is to evaluate our model, baselines, and ablations on the following criteria: (i) Leveraging mechanisms that explicitly model agent-to-agent and agent-to-scene interactions (experiment 1 and 2). (ii) Producing diverse trajectory predictions, while obeying admissibility constraints on the trajectory candidates given different approximation methods for the true trajectory distribution p (experiment 3). (iii) Remaining robust to an increasing number of agents in the scene (agent complexity; experiment 4). (iv) Learning better social context representations (experiment 5). (v) Generalising to other domains (experiment 6).

Dataset

Most current autonomous driving trajectory forecasting datasets are insufficient for evaluating predictions, due to the small size and the limited number of multimodal cues [184].

The ARGOVERSE motion forecasting dataset consists of a large volume of forecasting data with drivable area annotations, but lacks certain modalities i.e LiDAR point-clouds and map images. We have generated motion forecasting datasets from NUSCENES and ARGOVERSE tracking dataset using their original annotations through programmatic trajectory association, smoothing, and imputation. Unlike the ARGOVERSE forecasting dataset, this new dataset provides additional context information from LiDAR point-clouds and map information, for better forecasting performance. We utilise the trajectory record, vectorized geometry, and drivable area annotation as modalities for our research. In order to make the experimental setup of NUSCENES similar to ARGOVERSE, we crop each sequence to be 5 seconds long in total; 3 seconds for prediction and 2 seconds for observation, with a sampling rate of 2 Hz. Background information relating to NUSCENES, ARGOVERSE trajectory data generation are included in the supplementary material. By evaluating baselines and our models on both real world datasets, we provide complementary validation of each model's diversity, admissibility, and generalizability across domains.

Baseline Models

Deterministic baselines. We compare three deterministic models with our approach, to examine our model's ability to capture agent-to-agent interaction: *LSTM-based encoder-decoder* [271] (LSTM), *convolutional social pooling LSTM* (CSP) [73], and a deterministic version of *multi-agent tensor fusion* (MATF-D) [331]. For our deterministic model, we use an LSTM with our cross-agent attention module in the encoder, which we refer to as the *cross-agent attention model* (CAM). Because each model is predicated on an LSTM component, we set the capacity to be the same in all cases, to ensure fair comparison.

Stochastic baselines. We experiment three stochastic baselines. Our first stochastic baseline is a model based on a Variational Autoencoder structure, (DESIRE) [158], which utilises scene contexts and an iterative refinement process. The second baseline model is a Generative Adversarial Network version of multi-agent tensor fusion (MATF-GAN) [331]. Our third baseline is the Reparameterized Pushforward Policy (R2P2-MA) [239] which is a modified version of R2P2 [237] for multi-agent prediction. To validate our model's ability to extract scene information and generate diverse trajectories, multiple versions of our models are tested. While these models can be used as standalone models to predict diverse trajectories, comparison amongst these new models is equivalent to an ablation study of our final model. CAM-NF is a CAM model with a flow-based decoder. LOCAL-CAM-NF is CAM-NF with local scene features. GLOBALCAM-NF is LOCAL-CAM-NF with global scene features. Finally, ATTGLOBAL-CAMNF is

Table 6.1: Deterministic models on NUSCENES. Our proposed model outperforms the existing baselines.

Model	MINADE (↓)	MINFDE (↓)
LSTM	1.186	2.408
CSP [73]	1.390	2.676
MATF-D [331]	1.261	2.538
CAM (OURS)	**1.124**	**2.318**

GLOBAL-CAM-NF with agent-to-scene attention, which is our main proposed model.

Results

In this section, we show experimental results on numerous settings including the comparison with the baseline, and ablation studies of our model. We first show the effect of our cross-agent interaction module and agent-to-scene interaction module on the model performance, then we analyse the performance with respect to different numbers of agents, and other datasets. All experiments are measured with MINADE, MINFDE, RF, DAC, and DAO for holistic interpretation.

Effectiveness of cross-agent interaction module: We show the performance of one of our proposed models CAM, which utilises our cross-agent attention module, along with three deterministic baselines as shown in Tables 6.1. For each model we test, agent-to-agent interaction is considered in different ways. CSP models the interaction through layers of convolutional networks, and the interaction is implicitly calculated within the receptive field of convolutional layers. MATF-D is an extension of convolutional social pooling with scene information. CAM explicitly defines the interaction between each agent by using attention. The result shows that CAM outperforms other baselines in both MINADE and MINFDE, indicating that the explicit way of modelling agent-to-agent interaction performs better in terms of precision than an implicit way of modelling interaction using convolutional networks used in CSP and MATF-D. Interestingly, CAM outperforms MATF-D that utilises scene information. This infers that our cross-agent interaction module has the ability to learn the geometric structure of the roads given by the trajectories of surrounding agents.

Effectiveness of agent-to-scene interaction module: The performance of stochastic models is compared in Tables 6.2. We experiment with removing scene processing operations in the decoder to validate the importance of our proposed agent-to-scene interaction module. As mentioned previously, generating multiple modes of sample requires a strong scene processing module and a diversity-oriented decoder. Our proposed models all outperform other stochastic baseline models in terms of precision. MATF-GAN has a small RF inferring that the predictions are mostly unimodal, while other models such as VAE-based model DESIRE and flow-based models R2P2 and OURS show more spread in their predictions. We note that R2P2 was not designed for multi-agent setting which causing it to make unreasonably shaking outputs. Our model has the highest DAC and DAO, indicating that our models exhibit diverse and admissible predictions by accurately utilising scene context.

Effectiveness of new p loss. We experiment with MSE and our drivable area-based approximation of p in Table 6.3. Using our map loss in training shows superior results in most of the reported metrics. In particular, the precision and the diversity of predictions increases drastically as reflected in MINERROR and RF while DAC remains unchanged. Our map loss assures admissibility while improving precision and diversity, as drivable-area associated $\tilde{p}$ provides additional possible regions of future trajectories.

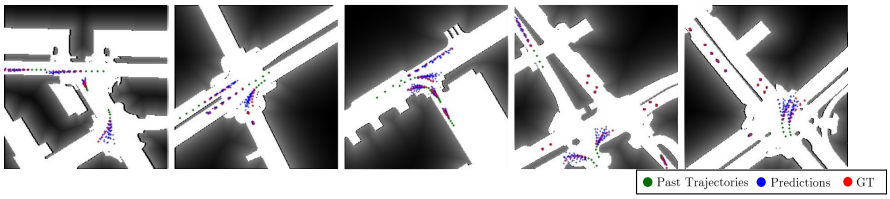


Figure 6.8: Our map loss and corresponding model predictions. Each pixel on our map loss denotes probability of future trajectories; higher probability values are represented by brighter pixels. Our approach generates diverse and admissible future trajectories. More visualisations of qualitative results are provided in the supplementary material.

Table 6.2: Stochastic models on NUSCENES. $**$: unstable outputs observed on R2P2-MA.

Model	MINADE (↓)	MINFDE (↓)	RF (↑)	DAO (↑)	DAC (↑)
DESIRE [158]	0.937	1.808	1.754	9.430	0.376
MATF-GAN [331]	1.053	2.124	1.194	5.950	0.391
R2P2-MA [237]	1.185	2.215	1.611	13.50**	0.396
CAM-NF (OURS)	0.756	1.386	2.113	11.70	**0.400**
LOCAL-CAM-NF (OURS)	0.772	1.404	2.066	11.70	**0.400**
GLOBAL-CAM-NF (OURS)	0.744	1.359	2.103	11.60	**0.400**
ATTGLOBAL-CAM-NF (OURS)	**0.638**	**1.171**	**2.558**	**12.28**	0.399

Table 6.3: Optimizing using p loss outperforms MSE loss on NUSCENES.

Model	MINADE (↓)	MINFDE (↓)	RF (↑)	DAO (↑)	DAC (↑)
ATTGLOBAL-CAM-NF(MSE)	0.763	1.390	2.009	12.09	**0.400**
ATTGLOBAL-CAM-NF	**0.638**	**1.171**	**2.558**	**12.28**	0.399

Table 6.4: Multi-agent experiments on NUSCENES (MINFDE). RI denotes ratio of MINFDE for 10 vs. 1 agent. Our approach best models multi-agent interactions.

Model	1 agent	3 agents	5 agents	10 agents	RI(1-10)
LSTM	2.736	2.477	2.442	2.268	17.1%
CSP [73]	2.871	2.679	2.671	2.569	10.5%
DESIRE [158]	2.150	1.846	1.878	1.784	17.0%
MATF GAN [331]	2.377	2.168	2.150	2.011	15.4%
R2P2-MA [237]	2.227	2.135	2.142	2.048	8.0%
ATTGLOBAL-CAM-NF (OURS)	**1.278**	**1.158**	**1.100**	**0.964**	**24.6%**

Complexity from number of agents. We experiment with varying number of surrounding agents as shown in Table 6.4. Throughout all models, the performance increases as the number of agents increases even though we observe that many agents in the surrounding do not move significantly. In terms of relative improvement RI, as calculated between 1 agent and 10 agents, our model has the most improvement, indicating that our model makes the most use of the fine-grained trajectories of surrounding agents to generate future trajectories.

Learning better social context representations. We benchmark two instances of our approach, `Trajformer-12` and `Trajformer-24`, with respectively 12 and 24 layers in the transformer encoder. We set the size of the trajectory encoder projection d to be 1024 ($\texttt{MLP}_{proj}$ is a single-layer projection), pixel neighbourhood width/height m to be 16, and the dimension of the latent code D to be 256. We choose a batch size of 128 and train with Adam optimizer. We use linear learning rate warm-up and decay. It takes 3 days to train

Table 6.5: Results of baseline models and our proposed model. LOCAL-CAM-NF is an ablation, whereas ATTGLOBAL-CAM-NF is our full proposed model. The metrics are abbreviated as follows: MINADE(**A**), MINFDE(**B**), RF(**C**), DAO(**D**), DAC(**E**). Improvements indicated by arrows. $*$: larger is better, as long as **A** and **B** are small.

Model	ARGOVERSE					NUSCENES				
	A (↓)	B (↓)	C (↑)*	D (↑)*	E (↑)*	A (↓)	B (↓)	C (↑)*	D (↑)*	E (↑)*
LSTM	1.441	2.780	1.000	1.786	0.378	1.186	2.408	1.000	1.690	0.391
CSP	1.385	2.567	1.000	1.799	0.379	1.390	2.676	1.000	1.710	0.388
MATF-D	1.344	2.484	1.000	1.768	0.379	1.261	2.538	1.000	1.690	0.384
DESIRE	0.777	1.276	**3.642**	11.80	0.301	0.937	1.808	1.754	9.430	0.376
MATF-GAN	1.214	2.316	1.099	6.075	0.376	1.053	2.124	1.194	5.950	0.391
R2P2-MA	1.270	2.190	1.589	18.10**	0.381	1.185	2.215	1.611	13.50**	0.396
CAM	1.131	2.504	1.000	1.750	**0.389**	1.124	2.318	1.000	1.670	**0.404**
CAM-NF	0.852	1.347	2.763	**17.60**	0.378	0.756	1.386	2.113	11.70	0.400
LOCAL-CAM-NF	0.807	1.250	2.858	17.00	0.381	0.772	1.404	2.066	11.70	0.400
GLOBAL-CAM-NF	0.807	1.241	3.068	16.90	0.380	0.744	1.359	2.103	11.60	0.400
ATTGLOBAL-CAM-NF	**0.731**	**1.126**	3.278	15.50	0.383	**0.638**	**1.171**	**2.558**	**12.28**	0.399

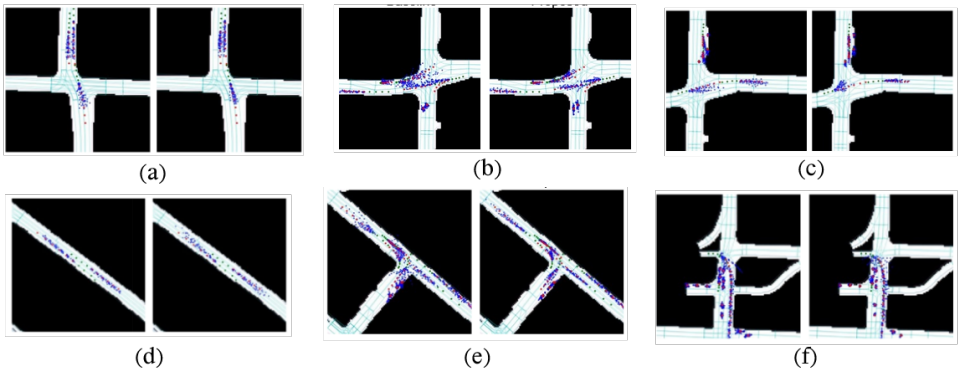


Figure 6.9: Qualitative illustration of model performance [right], compared to best-performing baseline [219][left]. Observations: (a) more precise & confident on straight in-lane trajectories; (b) more confidence in the maneuver, indicated by a cluster of trajectories; (c) more confident and diverse alternative maneuvers, with attention to standing vehicles, due to simple intersection map lane-start/end prior; (d) equivalent lane-change maneuver on empty roads, due to attention on immediate local activities; (e) more conservative turning maneuver with more agent-activity; (f) reduced confidence in strong curve lane-change maneuvers.

each model on a NVIDIA 1080 Ti GPU device, with batch data processed as in [219], from the *Tracking* split of the Argoverse dataset [44]. Quantitative results are summarised in Table 6.6, and some qualitative results are shown in 6.9. We observe a new state-of-the-art performance in our model, compared to [219], in both qualitative and quantitative results. Most of the maneuvers have been refined by the model. An interesting observation in figure 6.9b suggests that rule-based maneuvers, such as the right-of-way in intersections, are learned and followed by model (the left vehicle gains the right-of-way). Compared to DATF [219], the model is significantly lighter in time-complexity and memory-intensity, due to the social attention and scene attention blocks provided by the transformer encoder. The `Transformer-12` and `Transformer-24` model instances did not vary significantly in the qualitative and quantitative results:

Table 6.6: Comparison of improvements over baseline models on Argoverse. The metrics are abbreviated as follows: MINADE(**A**), MINFDE(**B**), RF(**C**), DAO(**D**), DAC(**E**). Improvements indicated by arrows. $*$: larger is better, as long as **A** and **B** are small.

	A (↓)	B (↓)	C (↑)*	D (↑)*	E (↑)*
LSTM	1.441	2.780	1.000	3.435	0.959
CSP [219]	1.385	2.567	1.000	3.453	0.963
MATF-D [331]	1.344	2.484	1.000	1.372	0.965
DESIRE [158]	0.896	1.453	3.188	15.17	0.457
MATF-GAN [219]	1.261	2.313	1.175	11.47	0.960
R2P2-MA [237]	1.108	1.270	2.190	37.18	0.955
DATF [219]	0.730	1.124	3.282	28.64	0.968
TRAJFORMER-12 (ours)	0.684	0.885	3.359	27.71	0.972
TRAJFORMER-24 (ours)	**0.621**	**0.719**	**3.868**	**28.21**	**0.973**

Table 6.7: Model size comparison (with optimizer state), in megabytes (MB) and number of parameters.

Model-layers	SIZE (.TAR)	#PARAMS
DATF [219]	4.7 MB	462K
TRAJFORMER-12 (ours)	2.1 MB	164K
TRAJFORMER-24 (ours)	2.9 MB	192K

the aforementioned observations remain true for both models.

Generalisability across datasets. We further compare our model with baselines extensively across two more real world datasets: NUSCENES and ARGOVERSE to test generalization to different environments. We show results in Table 6.5 where we outperform or achieve comparable results as compared to the baselines. For ARGOVERSE, we additionally outperform MFP3 [280] in MINFDE with 6 hypotheses: our full model shows a MINFDE of 0.915, while MFP3 achieves 1.399.

6.2.7 Related Work

Multimodal trajectory forecasting requires a detailed understanding of the agent's environment. Many works integrate information from multiple modalities [166, 224], such as RGB image and LiDAR point-cloud information to model the surrounding environment [158, 237] and high dimensional map data to modelling vehicle lane segmentation [18, 41, 331]. Other methods additionally fuse different combinations of map context [18, 41, 76], LiDAR [158], and RGB [184, 242] with the intention of jointly capturing all interactions between the agents and environment [4, 108, 247]. Without mechanisms to explicitly model agent-to-agent and agent-to-scene interactions, we hypothesise that these models are unable to capture complex nonlinear interactions in the high-dimensional input space. In this paper, we study and propose methods to explicitly model these interactions, escalating performance in trajectory forecasting.

Multi-agent modelling aims to learn representations that summarise the behaviour of one agent given its surrounding agents. These interactions are often modelled through either spatial-oriented methods or through neural attention-based methods. Spatial-oriented methods use pooling approaches across individual agent representations [73, 158, 331] and usually take into account inter-agent distances, through a relative coordinate system [18, 136, 218, 239]. Despite their wide usage, spatial-oriented methods are designed to concentrate only on adjacent (spatially close) agents and assume a fixed number of agents in the scene; they also limit the maximum number of agents. Attention-based methods use attention [288]

architectures to model multi-agent interaction for applications involving pedestrians [88, 247, 289], sports players [87, 269], indoor robots [228], and vehicle trajectories [184, 280]. In this paper, we use a cross-agent attention module to model the agent-to-agent interaction. Rather than using this information solely for prediction, we additionally generate attended scene context, conditioned on these cross-agent representations. We hypothesise that the attended map context will lead to improved tractability in modelling high-dimensional correlations in the scene. We support this with our empirical results in Section 6.2.6.

Diverse trajectory prediction. Many models follow a deterministic trajectory-prediction approach [73, 331] and, therefore, struggle to estimate the diversity in the future trajectories. Some works have applied generative models such as Generative Adversarial Networks (GANs) [103, 108, 247, 331] and Variational Auto Encoders (VAEs) [158] to encourage diverse predictions. However, these approaches focus more on generating and scoring multiple output candidates and focus less on analysing the diversity across distributional modes.

Trajectory forecasting. Trajectory forecasting has been studied in various domains, spanning marine vessels, aircraft, satellites, motor vehicles, and pedestrians [13, 31, 108, 239, 256]. Tasks involving motor vehicles and pedestrians are especially challenging, due to the high stochasticity that arises from attempting to model complex latent factors (e.g., human intent, "social" agent interacts, and scene context) [44]. Despite some promising empirical results, it remains difficult to evaluate both the diversity and admissibility of predictions. In this paper, we define the task of diverse and admissible trajectory forecasting and provide a new dataset generated from NUSCENES [37], a popular image tracking source. We also define new task metrics that specifically assess models on the basis of prediction diversity and admissibility, and we analyse model generalisation based on data from multiple domains.

6.3 Distribution-aware Goal Prediction in Urban Driving

Achieving generalisability to novel scenarios in urban autonomous driving remains a challenging task for artificial intelligence (AI). Recent approaches have shown promising results in end-to-end imitation learning from expert demonstrations, wherein agents learn policies that replicate the experts' actions, at each time-step, given the corresponding observations [29, 50, 64, 65, 200, 209, 215, 225]. Despite this progress, end-to-end imitative models often cannot capture the causal structures that underlie expert-environment interactions, leading models to misidentify the correct mappings from the observations [71]. Furthermore, if the coverage of expert demonstrations does not extend to *all* scenarios that the agent will encounter during test time, the agent will generate spurious actions in response to these out-of-distribution (OOD) observations [89].

In an effort to tackle *part* of this issue, recent works deviate from the end-to-end learning paradigm in their respective problem domains, opting instead to decompose learning into sub-modules, for trajectory forecasting [89, 238], indoor robot navigation [54], and learnable robot exploration [47]. Here, the intuition is that, by breaking down the inference problem into smaller units, more control over the inference step is obtained and the causal misidentification issue is somewhat avoided, by using a module that is not optimised through a data-driven process. The modularity of those approaches resonates with the method proposed in this paper, however those approaches use only the classical global-local hierarchical planning paradigm, where the responsibility of performing feature-extraction while also attempting to (implicitly) model the environment dynamics is still contained within a single unit, leading to spurious predictions in unseen environments. We proceed a step further, by defining modules in the learning-to-drive setting,

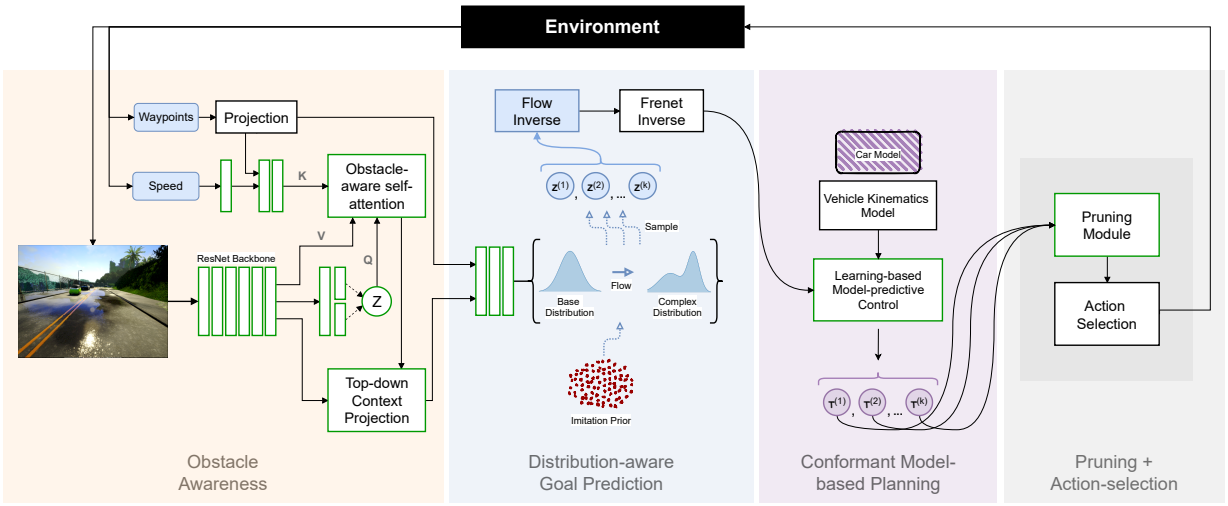


Figure 6.10: Model architecture. Our framework uses the ego-centric sequence of RGB images, world-frame waypoints, and the agent's own current speed information to learn obstacle-aware attention maps and top-down visual representations. These scene encodings inform our goal prediction module, which combines an imitation prior and a goal likelihood objective, in order to leverage expert experience for generalisability to novel scenarios. A set of candidate goal predictions are realised as trajectories, each transformed to the *Frenet* road frame coordinate system and grounded to vehicle kinematics, using a differentiable learning-based MPC controller with iLQR. Trajectories are pruned using a learnable ranking and refinement module. Boxes with green borders are learnable layers; boxes with black borders are non-learnable functions. Best viewed in colour.

such that each module's task is directly attributable to behaviour expected of an expert agent. In our decomposition, modules are given specialised roles (e.g., obstacle-awareness, explicitly modelling environmental dynamics, goal-prediction, trajectory pruning and refinement), improving both the tractability of their respective tasks and their complementarity towards the shared downstream task.

However, the challenge of generalisability still remains. How can we effectively utilise the expert's prior experience (e.g., in the form of expert demonstrations), while also achieving generalisability to novel scenarios? Some recent works from the trajectory forecasting community formulate a dual-objective optimisation, coupling an imitation objective with a likelihood density estimation term [219, 237, 239], arguing that the two ideals of using prior expert experience and generalising can be unified. A common issue with this formulation is that models are incentivised to *trade-off* the two objectives, rather than inherit their individual benefits. Samples from the likelihood density may not be sufficiently diverse, if the expert demonstrations did not provide sufficient coverage over the modes in the distribution over all possible predictions. Furthermore, predictions may not be admissible, discussed by [219], without some bias to adhere to, e.g., known physical constraints, as in *Verlet integration* [290]. In this work, we utilise expert demonstrations as pre-training for sub-modules and we use the demonstrations for density estimation, but we also ground predictions on a differentiable vehicle kinematics model and we constrain predictions to respect road admissibility through geometrical projection of goal prediction.

As a summary of our contributions, we produce a framework for generating diverse multi-mode predictions, for the learning-to-drive setting, that achieves improved generalisability through modular task structures, more informed goal likelihood density-estimation, explicit grounding on differentiable vehi-

cle kinematics for trajectory generation, and learnable trajectory-pruning through adversarial filtering and policy refinement. Our approach is also summarised in Figure 6.10. First, (i) we define a series of module primitives, based on insights about the decomposable nature of the environment. Next, (ii) we pursue model generalisability by coupling an imitation prior objective with a goal likelihood term, enabling the agent to leverage expert knowledge, while modelling more diverse modes in the underlying distribution over all trajectory futures. Next, (iii) we ground candidate trajectory predictions on conformant model-based vehicle kinematics, while learning to prune the predictions that are spurious. Finally, under CARLA simulation, (iv) we report new state-of-the-art results on the CARNOVEL benchmark.

6.3.1 Problem Formulation

We define, here, the terminology that we will use to characterise our problem. The ego-agent is a dynamic, on-road entity whose state is characterised by a 7D pose: a spatial position (consisting of x, y, and z in a Cartesian world coordinate frame), a speed v, and an orientation (consisting of *roll*, *yaw*, and *pitch*), evolving over time. For the position of the ego-agent at control time-step k, we use the notation $S_k = [x, y, v, yaw] \in \mathbb{R}^4$; for the agent's sequence of positions, from time-step k_1 to k_2, we use $S_{k_1:k_2}$. For the full sequence of the ego-agent's positions, for a single episode in the training data, we use (bold) $\boldsymbol{S}$. Setting k_0 as the present state, we define the agent's historical trajectory $t \leq k_0$ to be $\boldsymbol{S}_{\text{past}}$ and the agent's future trajectory (again, from the expert demonstrations) $t \geq k_0$ to be $\boldsymbol{S}_{\text{future}}$. At each control time-step, k, the agent is provided with contextual information from the environment, such as a frontal camera view $\boldsymbol{\Phi} \in \mathbb{R}^{H \times W \times C}$ and a sequence of waypoints $\boldsymbol{\omega}$. Combining S_{past}, $\boldsymbol{\Phi}$, and $\boldsymbol{\omega}$ we have the agent's observation, or simply $\mathcal{O} \equiv \{S_{\text{past}}, \boldsymbol{\Phi}, \boldsymbol{\omega}\}$.

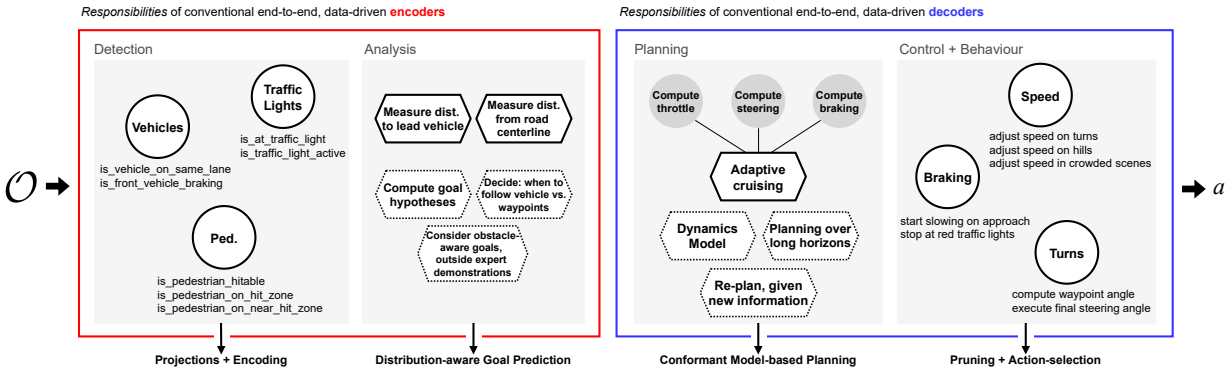
At each time-step, the agent must take an action a_k, defined as a tuple of *braking*, *throttling*, and *steering* control. Our objective is to learn a parameterised policy π_θ that maps observations to actions $a \sim \pi_\theta(\cdot|\mathcal{O})$, such that, given a sequence of observations, an agent that begins at some initial location in the environment can drive to some destination.

In this paper, we factorise the predictive distribution over actions, as a more tractable mapping: $\mathcal{P} \circ \mathcal{MPC} \circ \mathcal{GP} \circ \mathcal{OA}$. Here, $m \sim \mathcal{OA}(\cdot|\mathcal{O})$ is an obstacle-awareness module, which generates an embedding m, given an observation. $\hat{S}_{\text{goal}} \sim \mathcal{GP}(\cdot|\mathcal{O}, m)$ is a goal prediction module, whose samples are desired to be diverse in their coverage of the modes in the true, underlying goal prior $p(S_{\text{goal}}|\mathcal{O})$. Here, $\hat{S}_{\text{goal}}$ (hat) is the predicted goal from the $\mathcal{GP}$ module and S_{goal} is the true (unobserved) goal of the expert agent, which characterises its scene-conditioned navigational intent. We want $\mathcal{GP}$ to generate multiple samples, where each sample can be regarded as an independent hypothesis of what might have happened, given the same observation. $\hat{S}_{k+1:k+N}, \hat{a}_{k+1:k+N} = \mathcal{MPC}(\hat{S}_{k+N})$ is a learning-based controller, which takes K samples from the goal distribution as input and enumerates K navigation trajectory candidates. $\mathcal{P}$ is a pruning module that scores and selects the best trajectory, given an observation $\mathcal{O}$ and a collection of K trajectory candidates.

6.3.2 Modular Architectures for Multimodal Perception

Urban driving can be modelled as a composition of driving primitives, where, through decomposition of the conventional multimodal perception backbone into hierarchical units and through modular training, we enjoy lower sample-complexity and improved robustness and generalisability, compared to end-to-end policies. We propose a modular pipeline which models the multi-mode action distribution for conformant trajectory generation and planning, in urban driving settings. Our model, which we say performs

distribution-aware goal prediction (DGP), consists of four components: an *obstacle-awareness* module, a *goal-prediction* module, a *conformant model-based planning* module, and a *trajectory pruning and action-selection* module, as illustrated in Figures 6.10 (overview) and 6.11 (decomposition). Our framework uses the ego-centric sequence of RGB images, world-frame waypoints, and the agent's own current speed information to learn obstacle-aware attention maps and top-down visual representations. These scene encodings inform our goal prediction module, which combines an imitation prior and a goal likelihood objective, in order to leverage expert experience for generalisability to novel scenarios. A set of candidate goal predictions are realised as trajectories, each transformed to the *Frenet* road frame coordinate system and grounded to vehicle kinematics, using a differentiable MPC controller. The pruning module scores and filters trajectories, before feeding best trajectories for path-tracking.



(a) (b)

Figure 6.11: Issues with end-to-end imitative pipelines. The **red** (a) and **blue** (b) boxes illustrate the scope of responsibilities of conventional data-driven encoders and decoders, respectively, in the overall pursuit of replicating human driving behaviour. These include obstacle detection and scene analysis, and planning and control. Entities with dotted lines indicate behavioural components that lie outside the support of the expert demonstrations in typical learning-to-drive AI tasks, such as in CARLA simulation, such as: computing goal alternatives, in response to dynamic obstacle behaviour or re-planning over long horizons when presented with new route information. As a result, it is not possible for end-to-end imitative models to recover these skills from data, nor is it possible for end-to-end imitative models to exhibit the necessary degree of internal specialisation, without the adoption of modular training and role assignment. Our modular decomposition scheme (bottom arrows) is motivated by this taxonomy, as well as by the shortcomings of alternative decompositions.

6.3.3 Obstacle Awareness: Projections and Encoding

We condition the learning of our goal distribution on crucial scene context — from projected topdown feature representation and obstacle self-attention. This perception module's task is to transform the front-view image observations into bird's eye view (BEV) semantic object attention maps.

In this work, we leverage the orthographic feature transform (OFT) technique developed by [241]. In particular, we extract obstacle semantic information by pre-training a variational autoencoder [137] to reconstruct pixels, speed, and steering control in the next time step from current observations. It encourages the latent variables to attend to obstacle in front view (e.g., vehicles, pedestrians, traffic lights, curbs, etc.) which impact future vehicle control. The front-view feature map $f(u, v)$ is constructed by combining

the learned self-attention maps [288] with multi-scale images features of pre-trained ResNet-18 front-end. Then, voxel-based features $g(x, y, z)$ are generated by accumulating image-based features $f(u, v)$ to a uniformly spaced 3D lattice $\mathcal{G}$ fixed to the ground plane a distance y_p below the camera and has dimensions W, H, D and a voxel resolution of r using orthogonal transformation. Finally, the topdown image feature representation $h(x, z)$ is generated by collapsing the 3D voxel feature map along the vertical dimension through a learned 1D convolution. In addition to image features, we interpolate waypoint sequence and create a topdown grid representation of waypoints with one-hot encoding. The final topdown feature representation is of dimension $[W/r, D/r, C]$ where the number of channels $C = C_{\text{attn}} + C_{\text{resnet}} + 1$.

6.3.4 Distribution-aware Goal Prediction

We wish to approximate the true predictive distribution over all possible goal futures of the ego-agent, $p(S_{\text{goal}}|\mathcal{O}, m)$, given an observation $\mathcal{O}$ from the environment and an embedding vector m from the obstacle awareness module (§6.3.3). Unfortunately, the predictive *intent* of the expert agent is not observable from the training data: there do not exist ground-truth goal locations to use as labels for directly learning a scene-conditioned imitative prior over goals. Thus, we take a future state of the expert agent, at fixed time horizon T, to be the "ground-truth" ego-agent's goal $S_{\text{goal}} \in \boldsymbol{S}_{\text{future}}$, with $S_{\text{goal}} \equiv S_{k_0+N\Delta T}$, where N denotes the number of time-steps in the planning horizon.

Next, rather than learning a mapping to directly imitate these derived expert goals, we instead model an *approximation* $q_\theta(S_{\text{goal}}|\mathcal{O}, m)$ of the underlying goal distribution, by leveraging a bijective and differentiable mapping between a chosen *base distribution* q_0 and the aforementioned target approximate goal distribution q_θ. This technique is commonly referred to as a 'normalizing flow', which provides a general framework for transforming a simple probability density (base distribution) into a more expressive one, through a series of invertible mappings [139, 217, 219, 235, 274].

Formally, let f be an invertible and smooth function, with $f : \mathbb{R}^d \to \mathbb{R}^d$, $\mathbf{x} = f(\mathbf{z})$, $\mathbf{z} \sim p_{\mathbf{z}}$, $f^{-1} = g$, and thus $g \circ f(\mathbf{z}) = \mathbf{z}$, for d-dimensional random vectors $\mathbf{x}$ and $\mathbf{z}$. Further, we attribute to f the property of *diffeomorphism* [192], which ensures that $q_{\mathbf{x}}$ remains well-defined and obtainable through a change of variables, and ensures that $p_{\mathbf{z}}$ is uniformly distributed on the same domain as the data space [167] — insofar as both f and its inverse f^{-1} are differentiable and that $\mathbf{z}$ retains the same dimension as $\mathbf{x}$:

$$q_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z}) \left|\det \frac{\partial f}{\partial z}\right|^{-1} = p_{\mathbf{z}}(f^{-1}(\mathbf{x})) \left|\det \frac{\partial f^{-1}}{\partial \mathbf{x}}\right|$$

We can construct arbitrarily complex densities, by *flowing* $\mathbf{z}$ along the path created by a chain of K successive *normalizing* distributions $p_{\mathbf{z}}(\mathbf{z})$, with each successive distribution governed by a diffeomorphic transformation:

$$\mathbf{x} = \mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z_0})$$

Following this sequence of transformations, our main interfaces with the flow-based model are through either sampling or evaluating its density, where, in the former, we sample from $p_{\mathbf{z}}(\mathbf{z})$ and must compute the forward transformation f; in the latter, we must compute the inverse transformation f^{-1}, its Jacobian determinant, and the $p_{\mathbf{z}}(\mathbf{z})$ density evaluation.

We extend Eqn. (1) to obtain a conditional normalising flow formulation, in order to incorporate additional context (this additional context is sometimes referred to by some works as "side-information") and achieve

finer granularity in the density estimate of the goal distribution. While envisioned by Papamakarios et al. [216], conditioning has been hitherto unexplored for complex scenarios, as in autonomous urban driving, where the context dimension is high.

6.3.5 Trajectory Generation with Conformant Vehicle Dynamics

Being respectful of physical kinematic constraints provides a good basis for transfer to other vehicle morphologies and generalisation to unseen environments. Taking the predicted goals from the goal-prediction module, we enumerate the actions-to-goal, using model-predictive control. The MPC problem is summarised by Eqn. 6.12. The objective (Eqn. 6.12a) is to minimise the tracking error with respect to a reference trajectory, in this case the centerline of the race track at a pre-specified reference speed, with regularisation on actuations, over a planning horizon of T time steps. $\mathbf{Q}$ and $\mathbf{R}$ are both diagonal matrices corresponding to cost weights for tracking reference states and regularising actions. Simultaneously, MPC respects the system dynamics of the vehicle (Eqn. 6.12b) and allowable action range (Eqn. 6.12c):

$$\min_{\mathbf{a}_{1:T}} \quad \sum_{t=1}^{T} \left[(\mathbf{s}_t - \mathbf{s}_{ref,i})^T \mathbf{Q} (\mathbf{s}_i - \mathbf{s}_{ref,i}) + \mathbf{a}_i^T \mathbf{R} \mathbf{a}_i \right] \tag{6.12a}$$

$$\text{s.t.} \quad \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t), \quad \forall t = 1, \ldots, T \tag{6.12b}$$

$$\underline{\mathbf{a}} \leq \mathbf{a}_t \leq \bar{\mathbf{a}} \tag{6.12c}$$

Specifically, we characterise the vehicle with the kinematic bike model[1] [147], given in Eqn. 6.13, where the state is $\mathbf{s} = [x, y, v, \phi]$, and the action is $\mathbf{a} = [a, \delta]$. Here, x, y are the vehicle location in local east, north, up (ENU) coordinates, v is the vehicle speed, and ϕ is the yaw angle (measured anti-clockwise from the local east-axis). a is the acceleration, and δ is the steering angle at the front axle:

$$\dot{x} = v \cos(\phi) \tag{6.13a}$$

$$\dot{y} = v \sin(\phi) \tag{6.13b}$$

$$\dot{v} = a \tag{6.13c}$$

$$\dot{\phi} = v \tan \delta / L \tag{6.13d}$$

A key challenge is that the ground truth vehicle parameters were not known to us. Aside from L defined as the distance between the front and rear axle, the kinematic bike model expects actions, i.e. acceleration and steering, in physical units, while the environment expects commands in $[-1, 1]$. The mapping is unknown to us, and non-linear based on our observations. For instance, acceleration command = 1 results in smaller acceleration at higher speed. In the current implementation, we make a simplifying assumption that $a = k_1 \times$ acceleration command, and $\delta = k_2 \times$ steering command.

We use the iterative linear quadratic regulator (iLQR) proposed in [162], which iteratively linearizes the non-linear dynamics (Equation 6.13) along the current estimate of trajectory, solves a linear quadratic regulator problem based on the linearized dynamics, and repeats the process until convergence. Specifically, we used the implementation for iLQR from [8]. The parameters used by the MPC are summarised in Table 6.8.

[1]This set of equations is defined with respect to the back axle of the vehicle and is used for generating expert demonstrations. The kinematic bike model defined with respect to the centre of the vehicle is also included in our code base.

Table 6.8: MPC parameters

Parameter	Value
$\mathbf{Q}$	$\text{diag}([1, 1, 1, 16])$
$\mathbf{R}$	$\text{diag}([0.1, 1])$
v_{ref}	12.5 m/s
$\bar{\mathbf{a}}$	$[1, 0.2]$
$\underline{\mathbf{a}}$	$[-1, -0.2]$
L	2.7 m
k_1	10
k_2	6
T	6

6.3.6 Pruning and Action Selection

A heuristic trajectory pruning critic using Frenet transformed coordinates is employed in this work. A pruning score s_k is assigned each generated state trajectory $(x_t, y_t, v_t, \phi_t)_{t=1}^{T}$ by computing its maximum deviation from the waypoint sequence in Frenet coordinates, i.e., $s_k = \max_{t=1}^{T} |y_t^{\text{Fre}}|$. The trajectories with scores higher than a given threshold s_0 are pruned and the rest are used as candidate setpoint sequences fed to PID controller for path tracking.

6.3.7 Experiments

We assess the robustness of our approach to novel OOD driving scenarios in the CARNOVEL benchmark Filos et al. [89], comparing our work to strong baselines from both urban driving as well as trajectory forecasting. Predicated on the CARLA simulator [77], agents are first trained on the provided offline expert demonstration from `Town01` that were originally generated using a rules-based autopilot. Agents are then evaluated on various OOD navigation tasks, such as: busy-town settings, hills, and roundabouts. Performance is measured according to the following metrics: success rate (percentage of successful navigations to the destination), infractions per kilometre (ratio of moving violations to kilometre driven), and total distance travelled. In this setting, the agent is provided with an RGB image, a waypoint sequence, and vehicle speed; the agent must produce steering, throttle, and braking control, in order to navigation to a particular destination.

All experiments were conducted using CARLA simulator version 0.9.6, which is worth noting because this version introduced updates to the rendering engine and pedestrian logic, allowing for consistency across the three benchmarks but making the results of contemporary approaches on previous simulator versions no longer comparable [50]. We used a single GPU machine, with the following CPU specifications: Intel(R) Core(TM) i9-9920X CPU @ 3.50GHz; 1 CPU, 12 physical cores per CPU, total 24 logical CPU units. The machine includes two NVIDIA Titan RTX GPUs, each with 24GB GPU memory.

Baselines

We compare our model with the following baselines in the CARNOVEL benchmark:

Conditional imitation learning (CIL) [64] is an end-to-end behaviour cloning approach, which conditions its predictions on high-level commands and LiDAR information.

Learning by cheating (LBC) [50] extends CIL through cross-modal knowledge distillation, from a teacher network (trained on privileged information — e.g., environment state, overhead images) to a sensorimotor navigation agent (the ego-agent).

Deep Imitative Model (DIM) [238] is a trajectory forecasting and control method, which combines an imitative objective with goal-directed planning.

Robust Imitative Planning (RIP) [89] is the method that was proposed alongside the recent CARNOVEL benchmark. RIP is an epistemic uncertainty-aware method, targeted toward robustness to OOD driving scenarios.

6.3.8 Results

Generalisation to OOD scenes. In Table 6.9, we report the success rate of our approach, alongside strong baselines from both the learning-to-drive and trajectory forecasting communities. We show significant improvements in unseen generalisation in visuomotor control for urban driving from our approach, which jointly estimates the ego-agent's action distribution while learning to predict and score intermediate goals. Notably, we observe most significant improvements over the next-best model, Robust Imitative Planning (RIP; Filos et al. [89]; an epistemic uncertainty-aware model) when transferring models to completely unseen traffic layouts, such as Roundabouts. When the transfer represents a shift in environmental dynamics, however—as with the Hills scenarios—our approach receives moderate gains in variance.

Qualitative results on agent intention. A notable benefit of our factorising the urban driving problem—into encoding, distribution-aware goal prediction, conformant model-based planning, and pruning + action-selection—is that we obtain increased interpretability in our agent's prediction pipeline. Specifically, by way of our goal-prediction module, we obtain the ability to reveal agent intentionality, during its simulated task execution. Figure 6.12 offers qualitative results from our agent's interaction with the environment, during inference on a randomly-sampled episode. Each frame captures the agent's prediction (in red), given its own speed information, ego-vision context, and short waypoint sequences (in mustard). We observed that waypoint sequences, provided by the environment, may sometimes change, as agents approach locations where some decision must be made (e.g., turns). Coupled with our models estimation of the underlying action distribution, conditioned on offline dataset samples, our model correctly makes multiple admissible predictions of turning (a), going straight (b), changing lanes (c), or stopping/slowing (d) as it approaches the first intersection. After committing and executing the turn (e-f), the agent once again considers multiple possible futures (g), before once again following the rightward arcing waypoint sequence. On straight sections, the agent shows strong belief on forward movement, indicated by distant and straight goal predictions (h), but correctly slows (i) and stops (j) for obstacles.

6.3.9 Related Work

Learning to drive. Pomerleau [225] pioneered investigation of end-to-end imitation learning, for sensorimotor navigation in autonomous driving. Following some extensions [29, 200, 261] with applications in lane-following, highway driving, and obstacle avoidance, more recent works adapted the classic imitative modelling approach to urban driving scenarios [18, 50, 64, 65, 209, 215], with more complex road layouts and challenging dynamic obstacle interactions. Whereas the increased sample-efficiency from imitation allays much serious consideration of alternative learning paradigms, e.g., reinforcement, a common issue with imitative modelling arises from having to learn a representation from high-dimensional visual inputs, in highly-varying environments: even with sufficient data, models struggle to extract meaningful features

Table 6.9: Results of baseline models and our proposed approach on CARNOVEL [89]. We report Success Rate (↑; M × N scenes, %) on three novel (unseen scenarios).

	BUSYTOWN	**HILLS**	**ROUNDABOUTS**
CIL (Codevilla et al. [64])	05.45± 06.35	60.00± 29.34	20.00± 00.00
LBC (Chen et al. [50])	20.00± 13.48	50.00± 00.00	08.00± 10.95
DIM (Rhinehart et al. [238])	47.13± 14.54	70.00± 10.54	20.00± 09.42
RIP (Filos et al. [89])	62.72± 05.16	87.50± 13.17	42.00± 06.32
DGP (ours)	**63.60**± 00.00	**87.50**± 00.00	**70.00**± 0.00

from the input that are not confounded by high-frequency, label-independent variation (e.g., varied vehicle shapes, sensor miscalibration, different weather conditions, shadows, poor expert behaviour) [18]. In fact, access to more samples can actually yield worse performance, as low-quality data can lead the model to misidentify basic causal structures, underlying expert-environment interactions [71]. Following Codevilla et al. [64, 65], Chen et al. [50], Ohn-Bar et al. [209], Sauer et al. [252] use conditioning strategies, such as command variables, teacher networks, and mixtures of expert policies, in attempts to learn better conditional representations and thus reduce the search space for generating actions. However, a limitation of these works is that the number of modes that can be represented by these methods is limited by the number of pre-specified commands or experts—thereby limiting the model's generalisability to novel driving scenes.

Control strategies for autonomous vehicles. Aside from learning (e.g., neural) mappings from observations to actions, various works advocate for the use of feedback or model-based control: to simplify the learning process for the data-driven components of the framework, to replace the data-driven components entirely, or to ground neural predictions with explicit physical constraints. Chen et al. [50] utilise a proportional-integral-derivative (PID) controller to track the agent's target velocity, while Sauer et al. [252] use a PID controller for longitudinal tracking and a Stanley Controller (SC) [283] for lateral tracking, with respect to the road centerline. A feedback controller *myopically* and *reactively* determines its control actions based on deviations from the setpoints, whereas model-based controllers, such a model-predictive controller (MPC), can plan trajectories over long planning horizons by unrolling its model of the system dynamics. Herman et al. [117], Kabzan et al. [133] implement MPC controllers for their autonomous racing tasks, using ground truth vehicle states. However, the combination of such controllers and with high-dimensional sensory inputs remains unexplored. In this work, we integrate our perception and goal-prediction modules with an MPC, which generates trajectories conforming to vehicle kinematics.

Trajectory forecasting for autonomous driving. The notion of characterising distributions over all possible agent predictions has seen exciting growth in the domain of trajectory forecasting for autonomous driving [89, 158, 219, 237, 238, 239]. Whereas Lee et al. [158] use past trajectories and scene context as input for predicting future trajectories, and they score the 'goodness' of a trajectory as a learnable module, their method does not attempt to model the agent's predictive intent, e.g., as modes in a likelihood density. Rhinehart et al. [238] incorporated the concept of *goal-likelihood* into their model, and characterised the agent's objective via pre-specified geometric primitives: points, piece-wise linear segments, and polygons. However, their goal-likelihood is defined as simple set membership (i.e., within

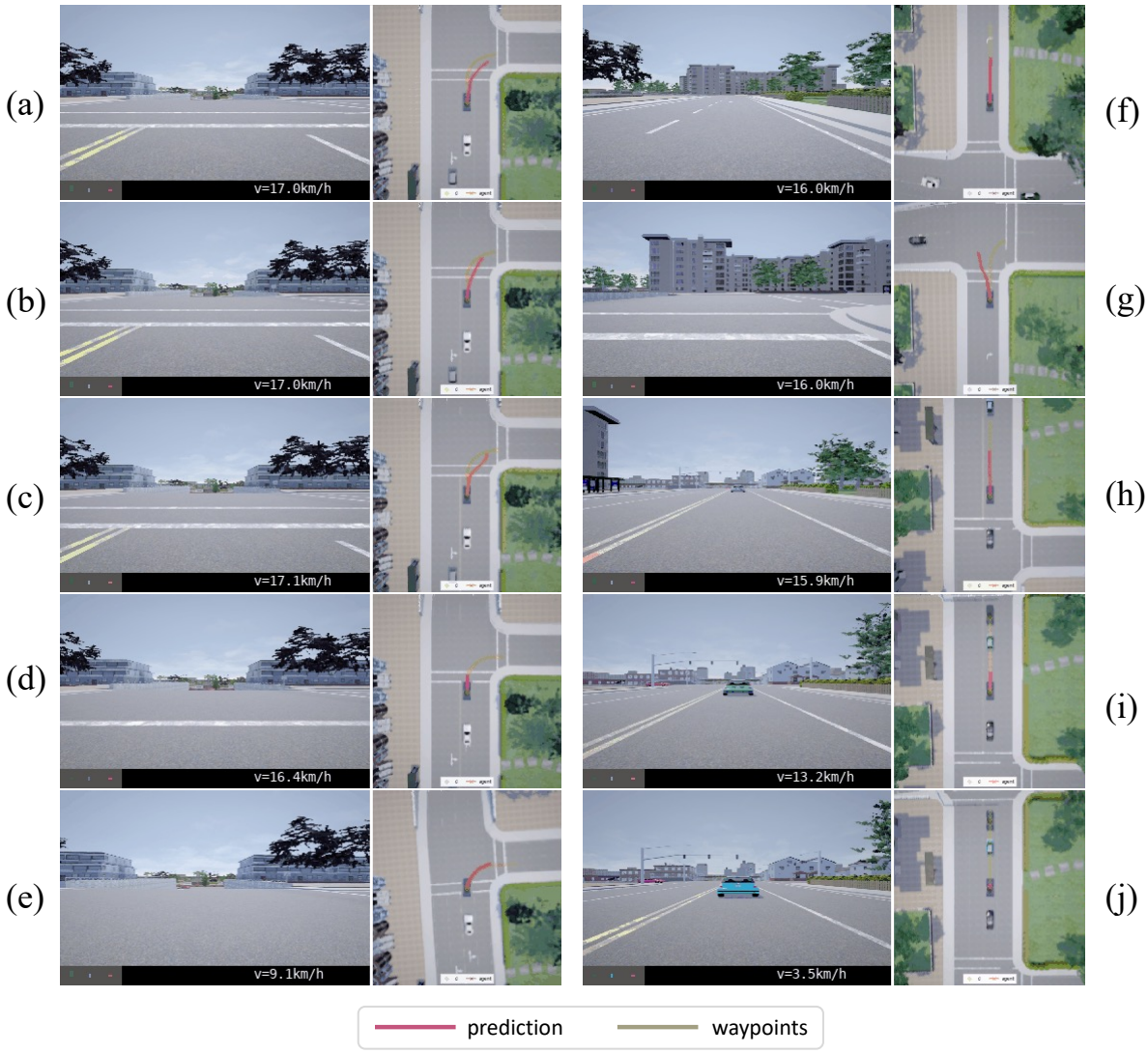


Figure 6.12: Qualitative results from simulated task execution.

the pre-specified geometry or not). Intuitively, set membership is neither a necessary or sufficient condition for good driving behaviour (e.g., banking vs following waypoints; avoiding obstacles vs staying on waypoints; staying within drivable area vs. driving safely). Filos et al. [89] aimed to improve on DIM by evaluating the trajectory on the basis of an ensemble of expert likelihood models; while that gives a more robust estimate of the 'goodness' of a trajectory, it neither considers dynamic obstacles nor more informative goal priors. Whereas multi-agent trajectory forecasting has slightly different intentions and implications than trajectory planning in learning-to-drive settings, we nonetheless draw inspiration from the trajectory forecasting literature for their distributional interpretation of the ego-agent's intent, which we combine with information about the scene context, for improved obstacle-awareness in those predicted trajectories.

6.4 Conclusion

This chapter covered the use of statistical priors, in the context of autonomous driving tasks, with a focus on how knowledge of the form of a distribution (which underlies trajectories or waypoint goals) leads to multiple benefits, including: (i) model generalisability, beyond simple interpolation between point-wise samples in a dataset; (ii) the ability to reveal agent intentionality for more interpretable predictions; and (iii) the ability to infer implicit rules about admissible behaviour in the environment. In the first section, we propose a model that addresses generalisation challenges in multi-agent trajectory forecasting tasks, through the introduction of an informative (and annotation-free) prior and rich multimodal encodings of agent-to-agent and scene-to-agent information. We offer new metrics to evaluate the diversity of trajectory predictions, while ensuring admissibility of each trajectory. Based on our new metrics as well as those used in prior work, we compare our model with strong baselines and ablations across two datasets and show a 35% performance-improvement over the state-of-the-art. In the second section, we learn the prior jointly with the task of simulated urban driving, where we introduce a distribution-aware trajectory generation mechanism that remains conformant to both road geometry and vehicle kinematics. We show how our agent uses this learned prior to generalise to completely out-of-distribution driving scenarios, such as busy towns, abnormal turns, unseen traffic patterns such as roundabouts, etc. We recognise a limitation of these works in the difficulty of the sub-task performed by modules that must learn to score the quality of multiple proposals, e.g., as in the pruning module (Section 6.3) which must learn to score the generated trajectory candidates proposed by the goal-prediction module and grounded by the conformant model- based planning module. In future work, we would consider conditioning the pruning module on additional domain knowledge, such as admissibility metrics (similar to the metrics we introduce in Section 6.2 or external functional constraints.

Chapter 7

Learning with Constraints

for safety-aware and robust prediction

7.1 Motivation

Whereas much of the existing research in autonomous driving focuses on urban or highway driving, simulated *Formula*-style track racing represents new challenges for artificial agents that must learn complex driving behaviour, safely and sample-efficiently, in rapid time-evolving scenarios. Contemporary works in simulation struggle to capture realism in the visual rendering, vehicular dynamics, and task objectives, inhibiting the transfer of learning agents to real-world contexts. Meanwhile, current algorithmic solutions are split between more classical approaches that demand privileged information, require significant parameter-tuning, and are limited in their performance capacity; versus approximate driving methods that provide no guarantees of safety and are prone to overfitting on the training scenarios. In this chapter, we study how to address both critical challenges in this task.

7.2 Safety-aware Policy Optimisation via Constraint Functions

Without safety constraints, autonomous agents are not guaranteed to adhere to safe behaviours, throughout interactions with their environment. In the context of safety-critical control applications, such as autonomous driving and human-robot interaction, recent literature has been concerned with studying how to best learn policies that are simultaneously safety-aware and performant. In the reinforcement learning (RL) literature, it is common to study constraint satisfaction under the constrained Markov decision process (CMDP) framework [6], which extends the Markov decision process (MDP) by incorporating constraints on expected cumulative costs. That is to say, aside from maximising expected cumulative rewards, the agent must ensure that the cumulative costs do not exceed pre-specified limits. The primary challenge of solving a CMDP problem is the need to evaluate whether a policy will violate the constraints [2]. Methods, such as constrained policy optimisation (CPO) [2], projection-based constrained policy optimisation (PCPO) [315] and conservative safety critic (CSC) [26], depend on collecting samples from the environment for policy evaluation. As a result, safety cannot be guaranteed, most notably during the initial learning interactions [58].

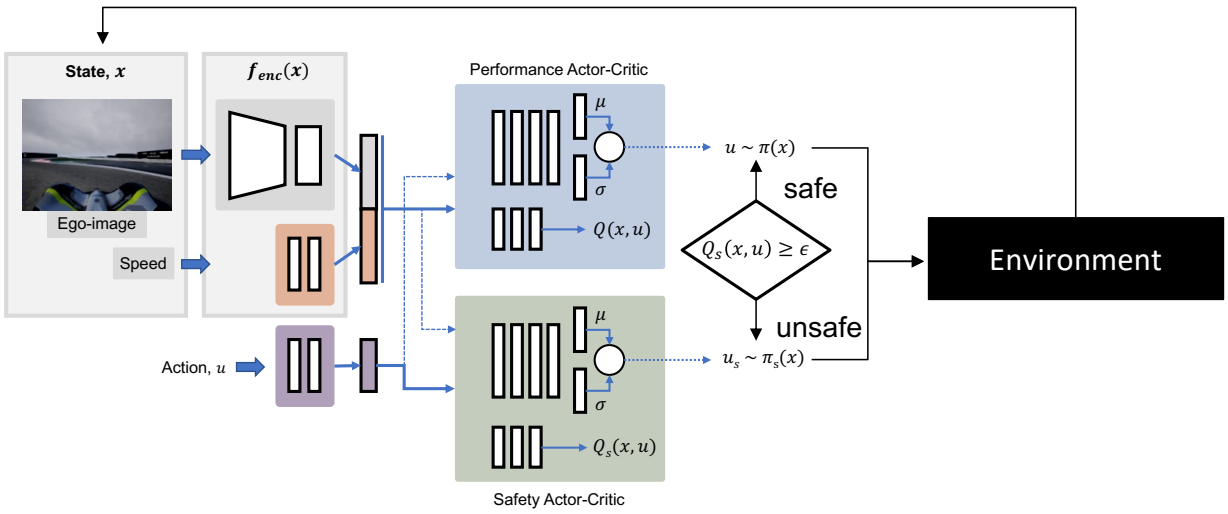


Figure 7.1: Summary of SPAR. We decompose the problem of learning under safety constraints into optimising for performance and updating safety value function. Thus, SPAR consists of two policies, both implemented with actor-critic architecture, which are in charge of safety and performance independently. First, we warm-start the safety critic, using values pre-computed with a nominal model. Then, we refine the safety critic based on observations from the environment, i.e., frontal camera view and speed, using HJ Bellman update. Simultaneously, we optimise the performance policy. At each time step, the safety critic verifies if the current state is safe, and only intervenes when necessary.

Given the practical limitations of letting agents learn safety by experiencing failures, it may be favourable to bootstrap the learning process with a model. Methods, such as Hamilton-Jacobi (HJ) reachability, compute safe sets with theoretical guarantees, using models of the system dynamics. In our autonomous racing experiments, we demonstrate that a safe set that is pre-computed with a simple kinematic vehicle model [147] can empirically keep even an agent that is generating actions at random on the drivable area, given a sufficiently large safety margin. For many engineering applications, such nominal models can often be specified by domain experts, despite complexity in the true dynamics.

In this section, we combine constrained RL with HJ Reachability theory in our problem formulation. Since safety verification under HJ Reachability theory does not depend on the performance policy, we can decompose the problem of learning under safety constraints into optimising for performance and updating safety value function. Given this intuition, we learn two independent policies (as shown in Figure 7.1), both implemented with actor-critic architecture, that manages safety and performance independently. While the performance policy focus exclusively on optimising performance, the safety critic verify if the current state is safe, and intervenes when necessary. The safety critic is updated via Hamilton-Jacobi Bellman update [91], grounded in control theory, and the safety controller is updated via the gradients from the safety critic. At the same time, the performance controller solves an unconstrained optimisation problem and thus can be trained with any appropriate RL algorithm.

As a summary of contributions, we propose the use of safety constraints for autonomous racing, inspired by the theoretical foundations of Hamilton-Jacobi (HJ) reachability analysis in optimal control. Here, we define a safety controller that intervenes whenever an agent approaches bad states, and we show that even an agent that generates actions at random is guaranteed to stay on the drivable area; we further show that an arbitrary learning policy that is coupled with this safe controller is able to learn performant driv-

ing behaviour, both safely and sample-efficiently. Finally, we demonstrate that the HJ safety value can be learned and updated directly from vision context, thereby expanding HJ reachability to applications where high-fidelity dynamics models may not be available. While not necessary for convergence, we warm-start the safety value function using values pre-computed with a nominal model. Next, the value function is updated directly on transitions of ego-agent's frontal camera view and vehicle speed. As a first experiment, we evaluate our approach on alongside strong baselines, in two environment and agent configurations, on the OpenAI Safety Gym framework; we report the minimum number of safety infractions, compared to state-of-the-art CMDP approaches. In the second experiment, we evaluate our approach on `Learn-to-Race` (`L2R`) [117], a recently-released high-fidelity autonomous racing environment, which requires the vehicle to make safety-critical decision in a fast changing environment. We obtain state-of-the-art results on `L2R` and show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

7.2.1 SPAR: Safety-aware Policy Optimisation for Autonomous Racing

Constrained MDPs. The problem of reinforcement learning (RL) with safety constraints is often formulated as a CMDP. On top of the MDP $(\mathcal{X}, \mathcal{U}, R, \mathcal{F})$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\mathcal{F} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ characterises the system dynamics, and $R : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, CMDP includes an additional set of cost functions, $C_1, \ldots, C_m$, where each $C_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ maps state-action transitions to costs characterising constraint violations.

The objective of RL is to find a policy $\pi : \mathcal{X} \to \mathcal{U}$ that maximises the expected cumulative rewards, $V_R^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k R(x_k, u_k) | x_0 = x\right]$, where $\gamma \in [0, 1)$ is a temporal discount factor. Similarly, the expected cumulative costs are defined as $V_{C_i}^\pi(x) = \mathbb{E}_{x_k, u_k \sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k C_i(x_k, u_k) | x_0 = x\right]$. CMDP requires the policy to be feasible by imposing limit for the costs, i.e. $V_{C_i}(\pi) \leq d_i, \forall i$. Putting everything together, the RL problem in a CMDP is:

$$\pi^* = \arg\max_{\pi} \; V_R^\pi(x) \quad \text{s.t.} \quad V_{C_i}^\pi(x) \leq d_i \;\; \forall i \tag{7.1}$$

HJ Reachability. To generate the safety constraint, we use HJ reachability applied to a simple, low-fidelity model of the vehicle kinematics, denoted as $\dot{x} = f(x, u)$. Here x is the state, u is the control contained within a compact set $\mathcal{U}$. The dynamics are assumed bounded and Lipschitz continuous. For discrete-time approximations the time step $\Delta t > 0$ is used.

We denote all allowable states as $\mathcal{K}$, for which there exists a terminal reward $l(x)$, such that $x \in \mathcal{K} \iff l(x) \geq 0$. An $l(x)$ that satisfy this condition is the signed distance to the boundary of $\mathcal{K}$. For instance, $\mathcal{K}$ is the drivable area and $l(x)$ is the distance to road boundary or obstacle in an autonomous driving scenario. This set $\mathcal{K}$ is the complement of the failure set that must be avoided. The goal of this HJ reachability problem is to compute a safety value function that maps a state to its safety value with respect to $l(x)$ over time. This is done by capturing the minimum reward achieved over time by the system applying an optimal control policy:

$$V_S(x, T) = \sup_{u(\cdot)} \min_{t \in [0, T]} l(\xi_{x,T}^{u,d}(t)), \tag{7.2}$$

where ξ is the state trajectory, $T < 0$ is the initial time, and 0 is the final time. To solve for this safety value function, a form of continuous dynamic programming is applied backwards in time from $t = 0$ to $t = T$ using the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI):

$$\min\left\{\frac{\partial V_S}{\partial t} + \max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle, l(x) - V(x, t)\right\} = 0, \quad V_S(x, 0) = l(x). \tag{7.3}$$

The super-zero level set of this function is called the reachable tube, and describes all states from which the system can remain outside of the failure set for the time horizon. For the infinite-time, if the limit exists, we define the converged value function as $V_S(x) = \lim_{T \to -\infty} V_S(x, T)$.

Once the safety value function is computed, the optimal safe control can be found online using the Hamiltonian: $u_S^* = \arg\max_{u \in \mathcal{U}} \langle f(x, u), \nabla V_S(x) \rangle$. This safe control is typically applied in a least-restrictive way wherein the safety controller becomes active only when the system approaches the boundary of the reachable tube, i.e. $u \sim \pi$ if $V_S(x, T) \geq 0$, u_S^* otherwise.

The newly introduced discounted safety Bellman equation [91] modifies the HJI-VI in (7.3) in a time-discounted formulation for discrete time:

$$V_S(x) = (1 - \gamma) l(x) + \gamma \min \left\{ l(x), \max_{u \in \mathcal{U}} V(x + f(x, u) \Delta t \right\}, \quad V_S(x, 0) = l(x). \tag{7.4}$$

This formulation induces a contraction mapping, which enables convergence of the value function when applied to approximate dynamic programming schemes commonly used in RL.

Safety-aware policy optimisation. We are inspired by guaranteed-safe methods, such as Hamilton-Jacobi reachability, which provides a systematic way to verify safety. Thus, we formulate our problem as a combination of constrained RL and HJ reachability theory. By leveraging this dynamics model, HJ enables evaluation of the agent's trajectory $\xi_x^u(\cdot)$ as *safe*, for all future time $t \geq 0$, insofar as $\xi_x^u(t) \in \mathcal{K}$ for constraint safe set $\mathcal{K}$. The need for an accurate model of the system dynamics can be restrictive, particularly for applications where such a model is difficult to develop or not available. We can, however, make learning more efficient by warm-starting the value estimation using a *nominal* model of the vehicle dynamics, when one is available. Because this nominal model does not come close to capturing the true underlying dynamics of the agent with respect to its environment, i.e., the residual dynamics (or disturbances) are expected to be left *unmodelled*, using the nominal dynamics to generate initial action-values does not preclude the subsequent application of Bellman-update formulæ, discussed in recent methods [26, 91]. Building upon this work, we demonstrate that it is possible to directly update the safety value function on high-dimensional multimodal sensory input—thereby expanding the scope of applications to problems previously inaccessible. We highlight the notable aspects of our framework:

i) HJ reachability provides a control-theoretic and low-latency way to verify safety. By incorporating HJ Reachability theory in the CMDP framework, we have a control-theoretic update rule to learn about safety and can verify safety by evaluating the safety value of the current state. Another positive outcome of the formulation is that the original constrained problem is decomposed into two unconstrained optimisation problems, making our formulation more amenable to gradient-based learning.

ii) Leverages domain knowledge. Aside from utilising HJ reachability theory for updating the safety critic, we also warm-start the safety value function with values pre-computed via a nominal model. We demonstrate in Section 7.2.5 that despite the naivety of the assumed model, it can empirically keep a random agent on the racetrack with a sufficiently large safety margin. This demonstrates the benefit of bootstrapping the learning process with readily-available domain knowledge.

iii) Achieves safe exploration. Our method is able to adhere to safe exploration strategies, particularly during early stages of training, when exploration is of paramount importance and when CMDP formulations tend to exhibit higher probabilities of failure.

iv) Scales to high-dimensional visual context. Compared to standard HJ Reachability methods, whose computational complexity scales exponentially with the state dimension, we updated the safety value

directly on vision embedding, with neural approximation. This is the highest-dimensional problem studied studied via HJ reachability to-date.

v) Simultaneously learns performance and safety. We formulate our problem as a combination of CMDP and HJ Reachability theory. As the result of the formulation, the problem of learning under safety constraints can be decomposed as optimising for performance and updating safety value estimation. Thus, we simultaneously, but independently, optimise a performance controller and a safety controller, both implemented with an actor-critic architecture. The safety backup controller is applied in a least restrictive way, only intervening when the RL agent is about to enter into an unsafe state and thus allowing the RL agent maximum freedom in exploring safely.

Unifying CMDPs and HJ reachability. We formulate our problem as a combination of CMDP and HJ Reachability theory. As described above, the objective of CMDP is to maximise cumulative rewards, subject to limits on cumulative costs characterising constraint violations. Without loss of generality, we can interpret the negative of a cost as a reward for safety and reverse the direction of the inequality constraint. Also recall that the super-zero set of the safety value function, i.e., $\{x|V_s(x) \geq 0\}$, designates all states that can remain within the set of allowable states, $\mathcal{K}$, over infinite time horizon. Thus, the safety value function derived from HJ reachability can be naturally embedded in the CMDP formulation:

$$\pi^* = \arg\max_{\pi} \ V_R(x), \quad \text{s.t.} \ \ V_S(x) \geq \epsilon, \tag{7.5}$$

where $\epsilon \geq 0$ is introduced as a safety margin. A key difference from the original CMDP formulation (Eqn. 7.1) is that constraint satisfaction, $V_S(x) \geq \epsilon$, no longer depends on the policy, π. Thus, we can bypass the challenges of solving CMDPs and decompose learning under safety constraints into optimising for performance and updating safety value estimation. While a number of works have similar dual-policy architecture [20, 58, 282], ours design is informed by HJ Reachability theory. Another difference is that HJ Reachability considers safety as absolute, and there is no mechanism to allow for some level of safety infraction, and thus χ_i in Eqn. 7.1 is no longer present.

Update of safety value function. We apply HJ Bellman update, in place of standard Bellman backup, to learn the safety value function. The learning rule proposed by [91] is defined on discrete action space, which we modify for continuous action space (Eqn. 7.6). While the safety actor is sub-optimal during learning, the resulting HJ Bellman target is a conservative estimation of the safety value, as $Q_S(x', u') \leq \max_{u' \in \mathcal{U}} Q_S(x', u')$, which is desirable for safety analysis. $Q_S(x, u)$ is updated model-free using state-action transitions, and only additionally requires $l(x)$. We assume $l(x)$ can be acquired from the vehicle's sensing capability [28] or estimated from perception [55].

$$\begin{aligned} Q_S(x, u) &= (1 - \gamma) l(x) + \gamma \min\{l(x), Q_S(x', u')\}, \\ u' &\sim \pi_S(x'). \end{aligned} \tag{7.6}$$

Our method, `SPAR`, consists of a performance policy and a safety policy. The safety backup controller is applied in a least restrictive way, only intervening when the RL agent is about to enter into an unsafe state, i.e., $u \sim \pi$, if $Q_S(x, u) \geq \epsilon$ and $u \sim \pi_S$ otherwise. The performance policy may be implemented with any RL algorithm. Since we expect the majority of samples to be from the performance policy, it is more appropriate to update the safety actor critic with an off-policy algorithm. In this work, we base our implementation of the safety actor critic on soft-actor critic (SAC) [110]. The safety critic is updated with Eqn. 7.6, and the safety actor π_S parameterised by θ_θ is updated via Eqn. 7.7, with learning rate η:

$$\theta_S \leftarrow \theta_S + \eta \mathbb{E}_{x,u \sim \mathcal{D}} \nabla_u Q_S(x, u) \nabla_\theta \pi_{\theta_S}(x). \tag{7.7}$$

State encoder representation. We condition the optimisation of the performance policy, as well as the safety value updates, on pre-trained embedding of vehicle's visual scene context. The perception module maps ego-images from the on-board RGB camera to feature embedding of reduced dimension. To learn this mapping, we use a standard variational autoencoding (VAE) [137] paradigm, with a convolutional encoder. We use an image reconstruction objective with binary cross-entropy loss, Adam optimizer [138], and a latent vector dimension of 32. We train the VAE encoder to reconstruct ego-images, sampled from the vehicle's front camera during random agent execution; examples are provided in Figure 7.2a. We further refine the encoder by training the VAE module to reconstruct projected road boundaries, illustrated in Figure 7.2b, with inputs in the left column and the reconstructed outputs in the right column. We first deploy the VAE on artificial projected images of road boundaries, where, compared with the left input image shown in left half in Figure 4, the right reconstructed image effectively learns the road boundaries, which demonstrates the few information loss in the latent vector. To transfer the model to the much more complex real racing environment, we crop the top part of the image to reduce distraction from irrelevant information like sky or sunlight so as to make neurons focus on learning the track shape and boundary. The final result is shown in right half of Figure 4. The 32-dimension latent vector is used as the visual feature that is passed, downstream, to the policies.

As illustrated in Figure 7.1, the vision encoder takes an image as input to produce a latent vector, which is concatenated with speed and action embedding and passed to the performance and safety actor-critics. The specific implementation of layers are summarised in Table 7.5.

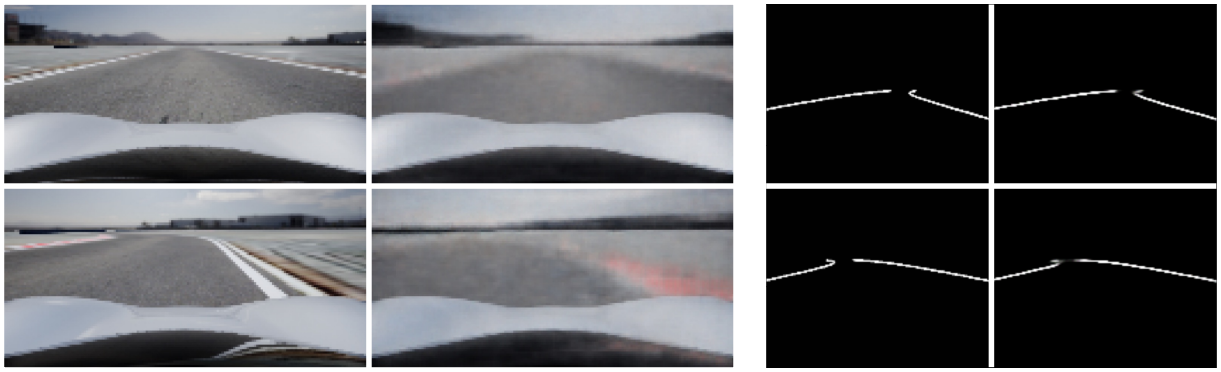
Agent training details. During training, the agent is spawned at random locations along the race track and uses a stochastic policy. During evaluation, the agent is spawned at a fixed location and uses a deterministic policy. The episode terminates when the agent successfully finishes a lap, leaves the drivable area, collides with obstacles, or does not progress for a number of steps. For each agent, we report averaged results across 5 random seeds, evaluated every 5000 steps over an episode (one lap). In total, we train each agent over 250,000 steps, and evaluate it over 50 episodes.

During its interaction with the environment, the agent receives a 192×144 ego-camera view and its speed at each time-step. The agent encodes the RGB image frame and its speed to a 40-dimensional feature representation, subsequently used as input to both actor-critic networks. We initialise the replay buffer with 2000 random transitions, following [1]. After 2000 steps, we perform a policy update at each time step. For the SafeSAC agent, we only save state-action transitions from the performance actor to the replay buffer. For the `SPAR` agent, we save all state-action transitions. Specifically, we use a squashed Gaussian policy for both performance and safety actors, following [110]:

$$u = \tanh(\mu(x) + \sigma(x) \odot \xi), \quad \xi \sim \mathcal{N}(0, \mathbf{I}) \tag{7.8}$$

7.2.2 Safe Learning Experiments on the Safety Gym Environment

We first evaluate our proposed approach, `SPAR`, in Safety Gym [233]. Specifically, we evaluate on the standard CarGoal1-v0 and PointGoal1-v0 benchmarks, where the agent navigates to a goal while avoiding hazards. We compare `SPAR` against baselines including: Constrained Policy optimisation (CPO) [2], an unconstrained RL algorithm (Proximal Policy optimisation (PPO) [253]), and its Lagrangian variant (PPO-Lagrangian). By default, distance measurements from LiDAR are available to all baselines in these benchmarks, and thus `SPAR` has direct access to $l(x)$. Episodic Performance and Cost curves are shown in Figure 7.3.



(a) Real-image VAE reconstruction

(b) Projected Road Boundary VAE reconstruction

Figure 7.2: (a) VAE image reconstruction, with real images in the left column and reconstructed images in the right column. (b) VAE reconstruction of projected road boundary images, with real images in the left column and reconstructed images in the right column.

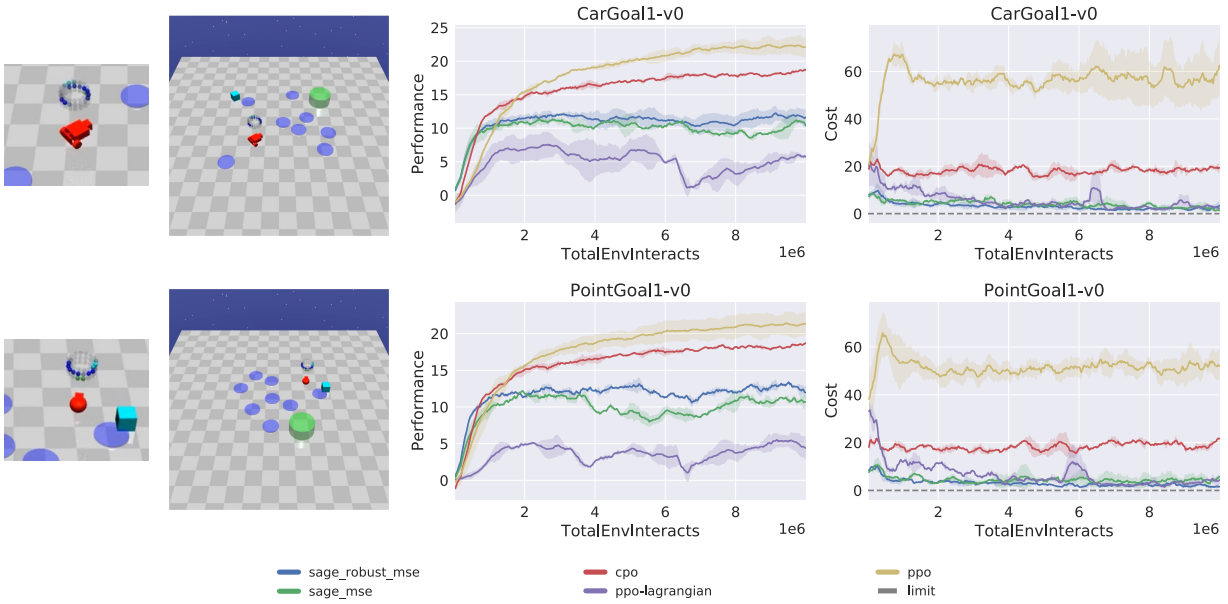


Figure 7.3: Performance of SPAR with comparison to baselines in the CarGoal1-v0 (top row) and PointGoal1-v0 (bottom row) benchmarks (averaged over 5 random seeds). In Goal tasks, agents must navigate to observed goal locations (indicated by the green regions), while avoiding obstacles (e.g., vases in cyan, and hazards in blue).

Following the default CarGoal1-v0 and PointGoal1-v0 benchmarks in Safety Gym, all agents were given LiDARs observation with respect to hazard, goal, and vase, with avoiding hazards as the safety constraints. Both environments were initialised with a total of 8 hazards and 1 vase. Agent's are endowed with accelerometer, velocimeter, gyro, and magnetometer sensors; their LiDAR configurations included 16 bins, with max distance of 3. The baselines we considered, i.e., CPO, PPO and PPO-Lagrangian follows the default implementation that comes with Safety Gym. PPO-SPAR wraps the proposed safety actor critic

around the PPO base agent. Despite PPO being an on-policy algorithm, the SPAR safety critic was implemented with off-policy updates, using prioritised memory replay based on the TD-error of predicting safety value. Since $l(x)$ is small in this environment, we scaled cost by a factor of 100. For the safety actor-critic, We used γ_S annealing from 0.85 to 1 following [91], $\tau = 0.005$, critic learning rate of 0.001, actor learning rate of 0.0003, and $\alpha = 0.2$ (regularisation on policy entropy). We used a safety margin $\epsilon = 0.25$, mainly to account for the dimension of the hazards (radius = 0.2). For each model, on each Safety Gym benchmark, results were reported as the average across 5 instances. All experiments in Safety Gym were run on an Intel(R) Core(TM) i9-9920X CPU @ 3.50GHz—with 1 CPU, 12 physical cores per CPU, and a total of 24 logical CPU units.

PPO-SPAR has significantly fewer constraint violations, compared to other baselines, and the number of violations decreases over time. While CPO and PPO-Lagrangian take into account that a certain number of violations are permissible, there is no such mechanism in SPAR, as HJ Reachability theory defines safety in an absolute sense. While the inability to allow for some level of safety infractions, unfortunately, compromises performance, SPAR learns mature obstacle-avoidance behaviours, compared to some policies, which may ignore traps in favour of fast navigation to goal locations. Violations that do occur in SPAR result from neural approximation error, and the number of violations decreases over time as the safety actor-critic gains experience, despite the randomised and constantly-changing episodic layouts. While constraint violation are non-terminal in the Safety Gym environment, the task in the next subsection *does* terminate episodes upon safety infraction.

7.2.3 `Learn-to-Race`: a Multimodal Continuous Control Environment

In this subsection, we discuss our creation and release of the Learn-to-Race framework for safe learning research in autonomous racing.

Progress in the fields of machine learning and artificial intelligence (AI) depends on challenging tasks and well-defined evaluation metrics for researchers to effectively compare and improve algorithms and architectures. Models in learning to drive settings continue to struggle with problems, such as sample-efficiency and generalisation to unseen scenarios, calling for more suitable benchmarks [50, 89, 219]. We hypothesise that high-fidelity simulation environments, with well-defined metrics and evaluation procedures, are conducive to developing more sophisticated agents, that are applicable to real-world deployments.

Competition-style racing not only has well-defined objectives, but also exhibits significant complexity. Compared to urban driving, the racing car must make safety-critical, real-time decisions based on multimodal inputs in a fast-changing environment, with consideration of competing agents in multi-agent racing. Further, a racing car operates at its physical limit and is significantly less stable compare to a street car, where small mistakes can steer the car out-of-boundary. We highlight simulated competition-style racing as an opportunity to encourage the development of learning strategies that are capable of meeting these stringent requirements. We hope this new class of algorithms can positively affect other applications that require making safety-critical, real-time decisions based on diverse information.

In this work, we address the lack of realistic simulators and tasks for studying high-speed driving and release our autonomous racing simulator, which includes numerous interfaces for both simulated and real vehicle instrumentation. Furthermore, we introduce `Learn-to-Race` (L2R), a multimodal and continuous control environment for training and evaluating autonomous racing agents. Our environment extends a racing simulator, which we use to accurately model competition-style racing cars, including their sensors, cameras, and vehicle dynamics, along with racetracks that are based off their of real-world counterparts.

As such, L2R provides agents with a variety of sensory information from multiple modalities, all of which is realistically accessible on a vehicle. While the tasks presented do not include transfer from simulation to real-world, success in a complex virtual environment like L2R is the first step towards real-world racing for learning agents. Competition-style track racing is an extreme challenge for AI, and our work opens up avenues for future research and development in problems that require making safety-critical, sub-second decisions, in highly-dynamic and unstable environments.

Concretely our contributions include: (i) the Arrival simulator, a high-fidelity competition-style autonomous racing simulator, which models simulated tracks and various vehicle sensor signals; (ii) L2R framework, a plug-and-play environment, which defines interfaces for various sensor modalities, provides an OpenAI-gym compliant training and testing environment for learning agents, and enables agents to learn to race on high-precision models of real-world tracks (e.g., the famed Thruxton Circuit and the Las Vegas Motor Speedway) and to use a suite of rich multimodal sensory information; (iii) an official L2R task and dataset with expert demonstrations, experiments, challenging metrics, and reference implementations; and (iv) release the simulator, code for the L2R framework, the dataset of expert demonstrations, and implementation of baseline agents with model checkpoints to facilitate reproducibility and to enable researchers to build on our work.

Figure 7.4: Learn-to-Race interfaces with a racing simulator, which features numerous real-world racetracks such as the Thruxton Circuit (*top-left*) and Las Vegas Motor Speedway (*top-right*). Simulated race cars (*bottom*) are empowered with learning agents, tasked with the challenge of learning to race for the fastest lap-times and best metrics.

Simulation Framework

L2R provides a series of interfaces for an agent to interact with a racing simulator, including the capabilities to send control commands and make observations of the environment and its own state via different sensors. L2R is implemented as a Gym environment [35], enabling quick prototyping of RL or other control algorithms. While we release the L2R environment and task (Section 7.2.3) alongside the Arrival Racing Simulator, we point out that other simulators may be used with our framework as well, including those provided by [273]. Figure 7.5 shows a summary of functionalities and interactions amongst the racing simulator, the L2R framework, and an agent.

Agent-Simulator Interaction. At each step t, an agent select an action, a_t, based on its current observation, s_t, using its policy, π_θ, that is $a_t \sim \pi_\theta(\cdot|s_t)$. The control action from the agent is forwarded to

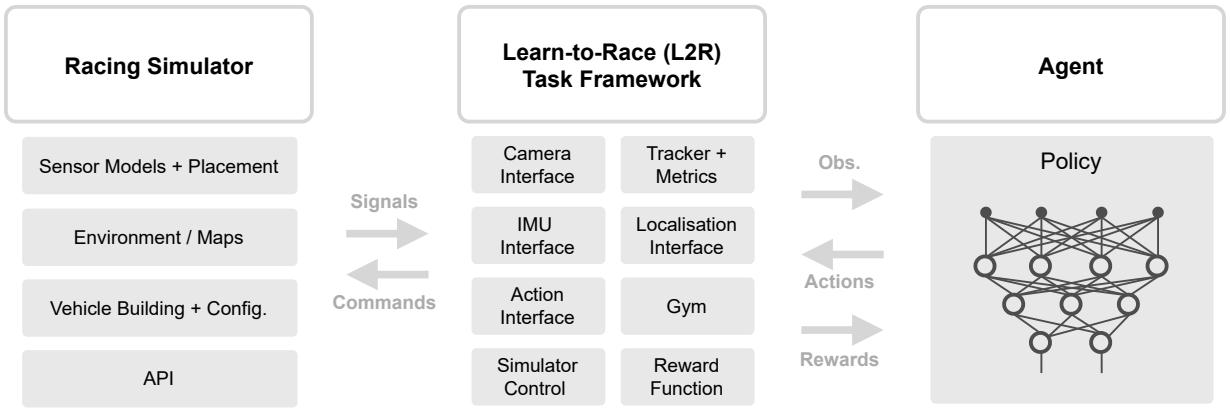


Figure 7.5: `Learn-to-Race` allows agents to interact with the racing simulator through a series of interfaces for observations, actions, and simulator control.

the simulator as a UDP message. `L2R` receives updates from the simulator, namely images from the virtual camera and/or measurements from other vehicle sensors, through TCP and UDP socket connections. Like in reality, update frequencies across the various sensor modalities are not equal, so `L2R` synchronises observations by providing agents with the most recent data from each as in Algorithm 1. The `step` method of the environment returns the new observation, s_{t+1}, along with a calculated reward to the agent, $r_t = R(s_t, a_t, s_{t+1})$, and a Boolean terminal state flag. The reward function and evaluation metrics are defined in Section 7.2.3.

Algorithm 1 Agent-Simulator Interaction

function SENSOR THREAD
    $data \leftarrow$ Initial value
    function GET DATA
        return $data$
    while $not\ terminated$ **do**
        $data \leftarrow$ **Receive Data**

function STEP(a_t)
    Send a_t as UDP message
    $s_{t+1} \leftarrow$ **Get Data** $\forall$ **Sensor Threads**
    $r_t \leftarrow R(s_t, a_t, s_{t+1})$
    $done \leftarrow$ **IsTerminal**(s_t, s_{t+1})
    return $s_{t+1},\ r_t,\ done$

Episodic Control. The control interface communicates with the simulator to automatically setup and execute simulations in an episodic manner. Our framework conveniently allows for training to be launched in one command, as all aspects of the racing simulator and learning environment are parameterized. A state is considered terminal if all laps are successfully completed, if at least 2 of the vehicle's wheels go outside of the drivable area, or if progress is minimally insufficient. The episode begins by resetting the vehicle to a standing start position, at a parameterised location, along with configured sensor interfaces and initialised reward function. Discrete steps are taken by the agent until one of the aforementioned

Table 7.1: Summary of the observation and continuous action spaces, for the `Learn-to-Race` task. When the simulator is initialised in *vision-only* mode, the observation space consists of just the images from the ego-vehicle's front-facing camera. The additional observation data, all of which is realistically accessible on a real racing car, is available in *multimodal* mode. *Whereas gear is permitted as a controllable parameter, we do not use it in our experiments.

	Signal	**Description**	**Dimension**
Action	Acceleration	Command in [-1.0, 1.0]	$\mathbb{R}^1$
	Steering	Command in [-1.0, 1.0]	$\mathbb{R}^1$
	Gear	{park, drive, neutral, reverse}	—
Observation	Image	RGB image	$\mathbb{R}^{W \times H \times 3}$
	Steering	Observed steering direction	$\mathbb{R}^1$
	Gear	{park, drive, neutral, reverse}	—
	Mode	Vehicle mode	$\mathbb{R}^1$
	Velocity	In ENU coordinate (m/s)	$\mathbb{R}^3$
	Acceleration	In ENU coordinate (m/s^2)	$\mathbb{R}^3$
	Yaw, Pitch, Roll	Orientation of the car (rad)	$\mathbb{R}^3$
	Angular Velocity	Rate of change of the orientation (rad/s)	$\mathbb{R}^3$
	Location	Location of the vehicle center in ENU (m)	$\mathbb{R}^3$
	Wheel Rotational Speed	*per wheel* (RPM)	$\mathbb{R}^4$
	Braking	Brake pressure *per wheel* (Pa)	$\mathbb{R}^4$
	Torque	*per wheel* $(N\dot{m})$	$\mathbb{R}^4$

episode termination criteria is met.

Task Definition

The `Learn-to-Race` task (`L2R`) tests an agent's ability to execute the requisite behaviours for competition-style track racing, through multimodal perceptual input. In this section, we provide a task overview and describe task properties, task dataset characteristics, and task metrics.

Task Overview. `L2R` is an OpenAI Gym [35] compliant learning environment, where researchers could flexibly select among the available sensor modalities. At the moment, it supports single-agent racing. This first version of the environment provides access to two racetracks, both modelled after their real-world counterparts. The first is the North Road Track at Las Vegas Motor Speedway, located in the United States, and the second is the Thruxton Circuit track, located in the United Kingdom. Analogous to having separate towns and maps for training and testing in other simulation environments, e.g., CARLA [77], we use Thruxton for training and the North Road track for testing. Consequently, we generate expert traces from the training track, for inclusion in our initial dataset release (see Section 7.2.3).

Many avenues for research can be explored within the `Learn-to-Race` framework, including: various learning paradigms (reinforcement learning, imitation learning, multitask learning, transfer learning, etc.), as well as both modular and end-to-end architectural strategies. Regardless of the method chosen, agents' multimodal perception capabilities—i.e., their ability to fuse and align sensory information—are of critical importance.

Learn-to-Race Dataset. We generate a rich, multimodal dataset of expert demonstrations from the training racetrack (Thruxton), in order to facilitate pre-training of agents via, e.g., imitation learning (IL). The `L2R` dataset contains multi-sensory input at a 100-millisecond resolution, in both the observation and action spaces. Depending on the selected simulator perception mode, agents have access to one (*vision-only mode*: images) or all modalities (*multimodal mode*. See Table 7.1 for a complete list of available modalities. The action space is defined by continuous values for acceleration and steering, in the ranges $[-1.0, 1.0]$, where negative acceleration values will decelerate the vehicle to a halted position. Note that *Gear* is a controllable action, but fixed to *drive* in all our experiments. Setting gear to park, neutral, or reverse does not help in the racing objectives.

The expert demonstrations were collected using a model predictive controller (MPC) (to be described in Section 7.2.3) that tracks the centerline of the race track at a pre-specified reference speed. This training dataset contains 10,600 samples of each sensory and action dimension, in this first version, which includes 9 complete laps around the track. Further description and visualisation of the dataset can be found in the supplementary material. Future version releases of `L2R` will include access to new simulated tracks (also modelled after real tracks, from around the world) as well as expert traces generated from these additional tracks—across various weather scenarios, in challenging multi-agent settings, and within dangerous obstacle-avoidance conditions.

Baseline Agents. We define a series of learning-free (e.g., `RANDOM`, `MPC`) and learning-based (e.g., reinforcement learning, imitation learning) baseline agents, to illustrate the performance of various algorithmic classes on the `Learn-to-Race` task. In addition, we provide a benchmark human performance, through a collection of expert races. The `RANDOM` agent is mainly intended as a simple demonstration of how to interface with the `L2R` environment.

The `RANDOM` agent is spawned at the start of the track, and uniformly samples actions, i.e., steering and acceleration, from the action space. The agent then proceeds to execute these random actions.

The `MPC` agent was used to generate expert demonstrations (Section 7.2.3) and is intended as a reference solution of `L2R` via classical control approaches. The MPC minimises the tracking error with respect to the centerline of the race track at a pre-specified reference speed. We use the iterative linear quadratic regulator (iLQR) proposed in [162], which iteratively linearises the non-linear dynamics along the current estimate of trajectory, solves a linear quadratic regulator problem based on the linearised dynamics, and repeats the process until convergence. Specifically, we used the implementation for iLQR from [8]. We adopt the kinematic bike model [147] to characterise the vehicle dynamics. While MPC implies optimal control performance, we want to point out the limitations of our current implementation. Firstly, the ground truth vehicle parameters were not known to us and we used estimated values. Secondly, we asked the MPC to follow the centerline of the track, which is not the trajectory expert drivers would have taken, especially when cornering. Finally, we pre-specified the MPC to drive at a conservative speed (12.5m/s), which makes the expert demonstrations easier to learn from.

For a `CIL` agent, we adopted the same neural architecture from Conditional Imitation Learning (CIL) [64], except that we do not have different commands in our case, e.g., turn left, turn right, go straight, and stop . Thus, we used a single branch for decoding actions. We assume both front view images and sensor measurements are available for the IL agent. In each sample, the input consists of a 512×384 image and 30 sensor measurements, and output is 2 actions (as listed in Table 7.1). Our CIL implementation automatically adjusts the neural architecture based on specified input-output dimensions. The imitation loss (Equation 7.9) is the mean squared error between the predicted action, $\hat{a}_t$, and the action taken by the expert, a_t: $\mathcal{L} = \sum_{i=1}^{n} ||\hat{a}_i - a_i||_2^2$

$$\mathcal{L} = \sum_{i=1}^{n} ||\hat{a}_i - a_i||_2^2 \tag{7.9}$$

Our RL-SAC agent is based on the Soft-Actor Critic (SAC) algorithm [75, 109], which is generally performant and known to be robust [83]. SAC belongs to the family of maximum entropy reinforcement learning (RL) algorithms, wherein an agent maximises expected return, subject to an entropy regularisation term (Equation 7.10), as a principled way to trade-off exploration and exploitation. Our RL-SAC agent demonstrates several of features in the environment: it operates in vision-only mode, but rather than learning directly from pixels, we pre-trained a convolutional, variational auto-encoder [137] made on sample camera images. Therefore, our agent only need to learns to decode actions from image embeddings using a multi-layer perceptron with two hidden layers of 64 hidden units each. Our agent's reward function was the environment's default with the inclusion of a bonus if the agent remained near the center of the track. The standard SAC objective is as follows:

$$\mathcal{J}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[R(s_t, a_t) - \mathcal{H}(\pi_\theta(a_t|s_t))] \tag{7.10}$$

We additionally establish a HUMAN performance baseline, by collecting simulated racing results from human expert players. The collection procedure involved a private crowd-sourcing event, which was split into two separate phases—practice/training and recording/testing. Expert players were already familiar with the simulator, task, and objective, prior to engaging in the event. In the training phase, players were instructed to engage in the race, until the variance in finished lap-times, for three consecutive runs, fell below a certain threshold. After this training phase, players were allowed to proceed to the testing phase, for which their top-3 laps were recorded. We averaged the top-3 results in the testing phase, from all experts, for each track; the training results were discarded.

Metrics for Assessing Racing Agent Performance

The primary objective of the L2R task is to minimise the time taken for an agent to successfully complete racing laps, with additional requirements on the agent's driving quality. We do not restrict the agent's learning paradigms to, e.g., IL or RL; on the contrary, we can envision a wealth of combination strategies and other methods that are applicable to the L2R task. While we do not include planning-only approaches as those that are consistent with the official L2R task, (i) we do encourage hybrid or model-based learning approaches to adopt these technologies as official task entries; furthermore, (ii) we do encourage the simulator and the L2R interface to be used to further research in these areas, more generally. Agnostic to the learning paradigm used, and inspired by concepts from high-speed driving and trajectory forecasting [219], we define the core modalities, metrics, and objectives that shall be used to train L2R agents and assess their performance. We summarise agents' action and observation spaces in Table 7.1 and the official L2R task metrics in Table 7.2.

We define the successful completion of an episode in the L2R task to be 3 completed laps, from a standing start; *Episode Completion Percentage* (ECP) measures the amount of the episode completed, and *Episode Duration* (ED) measures the minimum amount of time that the agent took to progress to its furthest extent, through the episode. We define *Average Adjusted Track Speed* (AATS) as a metric that measures the average speed of the agent, across all three laps of the episode. Metrics may also include adjustments

for environmental factors, such as wheel slippage and weather effects as the task matures. *Average Displacement Error* (ADE), a common metric in trajectory forecasting [219], measures the agent's average deviation from a reference path—in this case, the centerline of the track. *Trajectory Admissibility* (TrA) is the dimensionless metric α defined in Equation 7.11 where t_e is the duration of the episode and t_u is the cumulative time spent driving unsafely with exactly one wheel outside of the drivable area.

$$\alpha = 1 - \sqrt{\frac{t_{\mathrm{u}}}{t_e}} \tag{7.11}$$

We also utilise metrics that measure the smoothness of agent behaviour: *Trajectory Efficiency* (TrE) measures the ratio of track curvature to agent trajectory curvature, i.e., in terms of agent heading deviations; *Movement Smoothness* (MS) quantifies the smoothness of the agent's acceleration profile, adjusted for gravity, using the negated log dimensionless jerk, η_{ldj} in Equation 7.12, a valid smoothness measure as described in [15].

$$\eta_{ldj} = \ln\left(\frac{(t_2 - t_1)^3}{v_{peak}^2}\int_{t_1}^{t_2}\left|\frac{d^2v}{dt^2}\right|^2 dt\right) \tag{7.12}$$

One key property of `L2R` is that it is objective-centric. Rather than restricting agents to predefined incentive policies, input dimensions, or even input modalities, `L2R` allows and encourages flexibility so that agents can learn to race effectively. The default reward function for `L2R` is inspired by [92]. This policy provides dense rewards for progressing down the race track, consistent with the goal of minimising lap times, and negative rewards for going out-of-bounds.

Evaluation Procedures for Simulated Racing Agents

Agent assessment is conducted through a leaderboard competition, with two distinct stages: (1) pre-evaluation and (2) evaluation. Predicated on industry standards, we adopt a racing-centric pre-evaluation step, for assessing agent performance, giving agents a warm start on the test track before formal evaluation. Much like how human racing drivers are permitted to acquaint themselves with a new racing track, before competition, we run a pre-evaluation on models, with unfrozen weights, allowing for some initial (albeit constrained) exploration. In this pre-evaluation period, agents may explore the environment for a fixed time of 60 minutes, defined in the number of time-steps of discrete observation from the `L2R` framework. In the pre-evaluation, we further define a "competency check" that agents must pass, in order to successfully proceed through to the main evaluation phase. For the North Road track at Las Vegas Motor Speedway, the only competency check is that agents are able to successfully complete a lap during the pre-evaluation period with acceleration capped at 50% of maximum allowed in the action space. A successful episode is defined the completion of 3 laps from a standing start and the agent not going out of the driveable areas of the track. If the agent is unsuccessful in the pre-evaluation phase, it is disqualified and not evaluated further. As we continue to provide support for new tracks (necessitating more novel driving maneuvers), we will also continue to add and permute the driving competency checks, to maintain fairness of evaluation on those tracks.

Post a successful pre-evaluation stage, the final test stage occurs: agents are provided all the various input modalities and have to compete on the metrics defined Section 7.2.3. When the agent successfully passes through the pre-evaluation stage, the user is not provided with the results of the competency checks and instead is able to view the results of the complete evaluation directly on the leaderboard.

7.2.4 Benchmark Experiments on the `Learn-to-Race` Environment

Task Baseline Results

We evaluate each of the baseline agents—`HUMAN`, `RANDOM`, `MPC`, and `RL-SAC`—on the task of learning to race, with the objective of finishing 3 consecutive laps in minimal time. For all approaches, agents complete model training and tuning on the Thruxton track. We present the average of each metric across 3 consecutive episodes after this phase in Table 7.3. Afterwards, agents are evaluated on based on their performance on the Las Vegas track following the 1-hour pre-evaluation period described in 7.2.3. Learning-free agents, namely `RANDOM` and `MPC`, simply perform inference in the testing environment. The `RL-SAC` agent, a learning based approach, operates in vision-only mode and utilises the pre-evaluation stage to perform simple transfer learning to the new racetrack. The agent's image encoder does not have access to the test track prior to pre-evaluation and is not updated during this phase, but the model weights of the agent do update as new experience becomes available. Following the pre-evaluation phase, agents completed 3 consecutive episodes, and we present the average of each metric in Table 7.4.

Table 7.2: `Learn-to-Race` defines multiple metrics for the assessment of agent performance. These metrics measure overall success—e.g., whether and how fast the task is completed—along with more specific properties, such as trajectory admissibility and smoothness.

Metric	**Definition**
Episode Completion Percentage	Percentage of the 3-lap episode completed
Episode Duration	Duration of the episode, in seconds
Average Adjusted Track Speed	Average speed, across all three laps, adjusted for environmental conditions, in km/h
Average Displacement Error	Euclidean displacement from (unobserved) track centerline, in meters
Trajectory Admissibility	Complement of the square root of the proportion of cumulative time spent unsafe
Trajectory Efficiency	Ratio of track curvature to trajectory curvature (i.e., in agent heading)
Movement Smoothness	Log dimensionless jerk based on accelerometer data, adjusted for gravity

Table 7.3: Baseline agent results on `Learn-to-Race` task while training on Thruxton track, with respect to the task metrics in Table 7.2: Episode Completion Percentage (**ECP**), Episode Duration (**ED**), Average Adjusted Track Speed (**AATS**), Average Displacement Error (**ADE**), Trajectory Admissibility (**TrA**), Trajectory Efficiency (**TrE**), and Movement Smoothness (**MS**). Arrows (↑↓) indicate directions of better performance. Asterisks (*) in Tables 7.3 and 7.4 indicate metrics which may be misleading, for incomplete racing episodes.

Agent	**ECP** (↑)	**ED** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
`HUMAN`	100.0(±0.0)	78.6(±5.2)	79.29(±4.7)	2.4(±0.1)	0.93(±0.01)	1.00(±0.02)	11.7(±0.1)
`RANDOM`	0.5(±0.3)	14.0(±5.5)	11.9(±3.8)	1.5(±0.6)	0.81(±0.04)	0.33(±0.38)*	6.7(±1.1)
`MPC`	100.0(±0.0)	904.2(±0.7)	45.1(±0.0)	0.9(±0.1)	0.98(±0.01)	0.85(±0.03)	10.4(±0.6)
`RL-SAC`	31.1(±0.0)	251.2(±1.4)	50.5(±0.3)	0.5(±0.0)	0.97(±0.0)	0.48(±0.0)*	11.1(±0.4)

Human experts. Human experts strongly outperform both during training and testing phases suggesting a general understanding of racing as they can quickly adapt to a new track with notably different features including frequent and severe turns. The human experts fully complete 3 lap episodes at speeds near the vehicle's physical limits and estimate their lap-time performance to be within 10% of optimal. We expect strong agents to execute trajectories which are of lower curvature than the racetrack's centerline,

Table 7.4: Baseline agent results on `Learn-to-Race` task while testing on Las Vegas track.

Agent	ECP (↑)	ED (↓)	AATS (↑)	ADE (↓)	TrA (↑)	TrE (↑)	MS (↑)
HUMAN	100.0(±0.0)	176.2(±3.4)	114.2(±2.3)	1.7(±0.1)	0.88(±0.01)	1.09(±0.02)	10.1(±0.3)
RANDOM	1.0(±0.6)	21.9(±9.6)	9.2(±1.5)	1.4(±0.3)	0.74(±0.01)	0.18(±0.05)*	8.4(±1.0)
MPC	69.5(±10.7)	353.2(±54.8)	40.5(±0.1)	0.8(±0.1)	0.91(±0.02)	1.07(±0.01)*	10.4(±0.2)
RL-SAC	11.8(±0.1)	109.9(±7.5)	22.1(±1.5)	1.3(±0.1)	0.95(±0.01)	0.58(±0.01)*	9.9(±0.2)

or a **TrE** of at least 1.0, allowing the vehicle to maintain higher **AATS**. The human experts were the only agent to achieve this considering that failure to complete an episode distorts the metric. However, such trajectories are aggressive and arguably risky because they often involve cutting corners closely with the vehicle nearly outside of the driveable area which is apparent by their **ADE**, the highest of any agent, and a relatively low **TrA**. Additionally, human experts performed well relative to other agents terms of **MS**, measuring the smoothness of the acceleration profile, demonstrating a strong ability to anticipate the need for, and control of, changes in speed in a smooth manner.

Baseline agents. There are several notable conclusions that we make based on the performance of our baseline agents which we do not claim to be state-of-the-art. The first is that the task is indeed challenging, as even the `MPC` agent with an approximate car model failed to consistently complete laps on the test track. Even after over 1 million steps environment steps on the training track, the `RL-SAC` agent only completes about 90% of a lap due to the challenging speed trap near the finish line at Thruxton. However, the `RL-SAC` agent demonstrates better control than the `MPC` in training in both **ADE** and **BF**. Second, we note the lack of generalisation and poor sample efficiency of the `RL-SAC` agent whose performance dropped significantly in terms of ability to progress down the track, **ECP**, and stay near the centerline, **ADE**, despite being directly incentivised to do so. The agent learns to simply stop altogether to avoid going out-of-bounds about 1/3rd of the way around the test track. We note that imitation learning has potential for providing agents with strong priors. However, in our experiments, automatic network sizing based on input/output dimensions and step-wise supervision alone, suggested by [77], did not yield good performance. This demonstrates the challenge that `L2R` poses to this family of approaches, necessitating consideration of, e.g., hybrid IL/RL strategies.

7.2.5 Safe Learning Experiments on `Learn-to-Race` Environment

Overview

In this section, we evaluate our approach using the Arrival Autonomous Racing Simulator, through the newly-introduced and OpenAI-gym compliant `Learn-to-Race` (`L2R`) task and evaluation framework (Section 7.2.3). `L2R` provides multiple simulated racing tracks, modelled after real-world counterparts, such as Thruxton Circuit in the UK (`Track01:Thruxton`; see Figure 7.6) and the North Road Track at Las Vegas Motor Speedway in the US (`Track02:Vegas`). In each episode, a racing agent is spawned on a given track. At each time-step, the agent may have access to RGB images from any specified location, semantic segmentation, and vehicle state (e.g., pose, velocity) and uses its choice of information to determine normalised steering angle and acceleration. ll learning-based agents receive the reward specified by `L2R`, which is formulated as a weighted sum of reward for driving fast and penalty for leaving the drivable area; the main objective is to complete laps in as little time as possible, while committing as few infractions as possible. Additional metrics are defined to evaluate driving quality. Complementary to other benchmarks, the high-speed nature of `Learn-to-Race`, coupled with its realistic simulated dy-

namics, allows for a comprehensive study of Safe RL methods. In this section, we describe the evaluation of our approach, **S**afety-aware **P**olicy Optimisation for **A**utonomous **R**acing (`SPAR`), against a series of benchmarks and research goals.

(a) Aerial (b) Third-person (c) Ego-view

Figure 7.6: We use the Arrival Autonomous Racing Simulator, within the `Learn-to-Race` (`L2R`) framework (Section 7.2.3). This environment provides simulated racing tracks that are modelled after real-world counterparts, such as the famed Thruxton Circuit in the UK (`Track01:Thruxton`, (a)). Here, learning-based agents can be trained and evaluated according to challenging metrics and realistic vehicle and environmental dynamics, making `L2R` a compelling target for safe reinforcement learning. Each track features challenging components for autonomous agents, such as sharp turns (visualised in (b)), where SPAR only use ego-camera views (shown in (c)) and speed.

Task: Learning to Race with Safety Constraints

Our overall objective is to train RL agents to adhere to safety constraints, i.e., successfully complete laps without safety violations, while being as performant as possible. To characterise the empirical safety performance of our approach, we report results primarily according to the following metrics, in dedicated ablation experiments: the Average Adjusted Track Speed (*AATS*) and the Episode Completion Percentage (*ECP*) metrics (Section 7.2.3) as proxies for agent performance and empirical safety, respectively. We also provide full benchmark results on the other `L2R` metrics, defined in [117].

We use `Track01:Thruxton` in `L2R` (Fig. 7.6) for all stages of agent interaction with the environment. During training, the agent is spawned at random locations along the race track and uses a stochastic policy. During evaluation, the agent is spawned at a fixed location and uses a deterministic policy. The episode terminates when the agent successfully finishes a lap, leaves the drivable area, collides with obstacles, or does not progress for a number of steps. For each agent, we report averaged results across 5 random seeds evaluated every 5000 steps over an episode, i.e., one lap. We use SAC as the performance policy, and all agents only have access to ego-camera view (Figure 7.6c) and speed, unless otherwise specified.

Implementation Details. For all experiments, we implemented the models using the `PyTorch` 1.8.0. We optimised both the performance and safety actor-critic with Adam [138], with a learning rate of 0.003. We used $\gamma = 0.99$ for the performance critic, and annealed γ_S from 0.85 to 1 for the safety critic following [91]. We used $\tau = 0.005$ for the performance critic, and $\tau = 0.05$ for the safety critic. For both the performance and safety actor, we include the policy entropy term with $\alpha = 0.2$. We used a batch size of 256, and a replay buffer size of 250,000. For rendering the simulator and performing local agent verification and analysis, we used a single GPU machine, with the following CPU specifications: Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz; 1 CPU, 4 physical cores per CPU, total of 4 logical CPU

units. The machine includes a single GeoForce GTX TITAN X GPU, with 12.2GB GPU memory. For generating multi-instance experimental results, we used a cluster of three multi-GPU machines with the following CPU specifications: 2x Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz; 80 total CPU cores using a Cascade Lake architecture; memory of 512 GiB DDR4 3200 MHz, 16x32 GiB DIMMs. Each machine includes 8x NVIDIA GeForce RTX 2080 Ti GPUs, each with 11GB GDDR6 of GPU memory. Experiments were orchestrated on the these machines using Kubernetes, an open-source container deployment and management system. All experiments were conducted using version 0.7.0.182276 of the Arrival Racing Simulator. Our `Learn-to-Race` framework [117] are available for academic-use, here: `https://learn-to-race.org`.

Table 7.5: Network Architecture

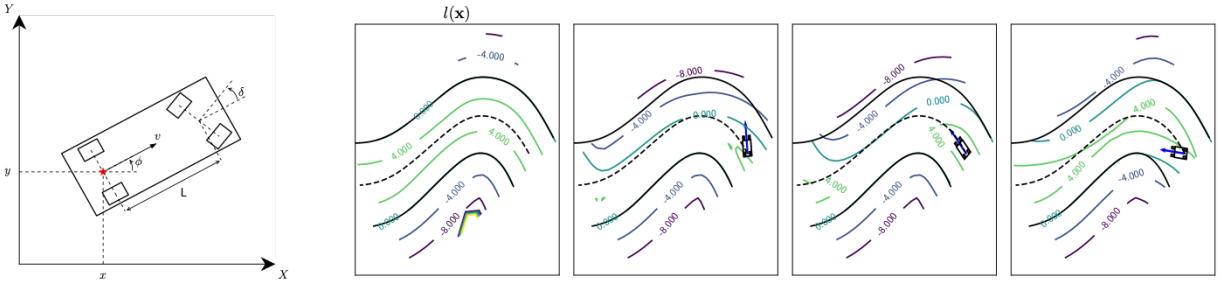
Operation	Input (dim.)	Output (dim.)	Parameters
	VISUAL ENCODER		
Conv2d	(N, chan, 42, 144), chan : 3→32	`conv1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv1`, chan : 32→64	`conv2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv2`, chan : 64→128	`conv3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Conv2d	`conv3`, chan : 128→256	`conv4`	k:=(4,4), s:=2, p:=1, activation:=ReLU
Flatten	—	—	—
	VISUAL ENCODER BOTTLENECK REPRESENTATION		
Linear (mu)	$N \times$ h_dim	$N \times 32$	—
Linear (sigma)	$N \times$ h_dim	$N \times 32$	—
	VISUAL DECODER (only for pre-training Visual Encoder)		
Unflatten	—	—	—
ConvTranspose2d	`encoder.conv4`: encoder.conv4.chan: 256 →128	`convtranspose1`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose1`, chan : 128 →64	`convtranspose2`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose2`, chan : 64 →32	`convtranspose3`	k:=(4,4), s:=2, p:=1, activation:=ReLU
ConvTranspose2d	`convtranspose3`, chan : 32 →3	`convtranspose4`	k:=(4,4), s:=2, p:=1, activation:=Sigmoid
	SAFETY ACTOR-CRITIC		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	PERFORMANCE ACTOR-CRITIC		
`actor_network`	—	—	—
`q_function1`	—	—	—
`q_function2`	—	—	—
	ACTOR NETWORK (POLICY): SQUASHEDGAUSSIANMLPACTOR		
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear (projection: mu_layer)	$N \times 32$	$N \times 3$	—
Linear (projection: log_std_layer)	$N \times 32$	$N \times 3$	—
	Q FUNCTION		
`speed_encoder`	—	—	—
`regressor`	—	—	—
	SPEED ENCODER		
Linear	$N \times 1$	$N \times 8$	activation:=ReLU
Linear	$N \times 8$	$N \times 8$	activation:=Identity
	REGRESSOR		
Linear	$N \times 42$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 64$	activation:=ReLU
Linear	$N \times 64$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 32$	activation:=ReLU
Linear	$N \times 32$	$N \times 1$	activation:=Identity

Warm-starting value estimation. While warm-start initialisation is not necessary for convergence of the safety value update, we want to demonstrate that the incorporation of readily available domain knowledge can drastically boost safety performance, especially during the initial learning phase. To do that, we first compute the safety value using a nominal model, and then pre-train the safety value function by learning the mapping from ego-view image and speed to HJ safety value.

Constructing the safe set via kinematic vehicle model. We use the kinematic vehicle model [147] to compute the safety value (see Figure 7.7), which is very simplistic compared to a realistic race car model. In comparison, [133] develop a high-fidelity vehicle model identified through extensive testing, and addi-

tionally characterise unmodeled dynamics via Gaussian process.

The dynamics and optimal safety control from solving the Hamiltonian is given in Equation 7.13, where the state is $\mathbf{x} = [x, y, v, \phi]$, and the action is $\mathbf{u} = [a, \delta]$. x, y, v, ϕ are the vehicle's location, speed, and yaw angle. a is the acceleration, and δ is the steering angle. $L = 3m$ is the car length. Setting $V_s(x, 0) = l(x)$, we calculated the backward reachable tube using the code base by [102]. For efficient computation, we divided the racetrack into overlapping segments and computed the safety value segment by segment. Fig. 7.7 illustrates resulting safety value function at slices of state space, when the vehicle enters into a sharp turn.



(a) Aerial (b) Third-person

Figure 7.7: (a) We compute the safety value function, via a kinematic vehicle model. (b) We illustrate different views of the 4D state space, given fixed velocity and three different yaw angles, indicated by the blue arrows. Here, $V_s(x, y, v, \phi)$ is computed via the nominal model, where v=12m/s.

$$\begin{cases} \dot{x} &= v\cos(\phi) \\ \dot{y} &= v\sin(\phi) \\ \dot{v} &= a \\ \dot{\phi} &= v\tan\delta/L \end{cases}, \qquad a^* \begin{cases} \underline{a} & \text{if } \nabla_v V \leq 0 \\ \overline{a} & \text{else} \end{cases}, \qquad \delta^* \begin{cases} \overline{\delta} & \text{if } \nabla_\theta V \geq 0 \\ \underline{\delta} & \text{else} \end{cases} \tag{7.13}$$

We characterise the empirical performance of this safety value function by comparing an agent that takes actions at random (`Random`) with a random agent that is coupled with our static safety backup controller (`SafeRandom`). We evaluate `SafeRandom`, through a series of safety margins, i.e., $\{3.2, 3.8, 4.2, 4.6, 5.0\}$. Empirically, $\epsilon \geq 4.2$ is sufficient to ensure the vehicle do not go out of bound at least in the speed range covered by the `SafeRandom` agent. Finally, we collected $\sim$300K state-action transitions with the `SafeRandom` agent to pre-train the safety critic in `SPAR` as a regression task.

Online safety. We pursue multiple experimental objectives, to illustrate the effect of safety constraints on learning. We use soft-actor critic (SAC) [110] as the base policy class for all learning agents.

Learning to race with safety constraints. We examine the effect of imposing safety constraints on performance and sample-efficiency, by comparing an RL agent (`SAC`) with an instance of itself that is coupled with the static safety controller (`SafeSAC`), using the pre-computed safety value function with a nominal dynamics model.

Continual safety value function updates. In this experiment, we study the effect of continual refinements of the safety value, using high-dimensional multimodal information, while optimising for driving performance. We want to show that the safety critic can learn about environment dynamics directly from vision

context and, with this better characterisation of the safety value function, the agent no longer needs to depend on a large safety margin. We compare the performance of `SafeSAC` (static safety backup) with our proposed `SPAR` agent (dynamic updates to the safety value function). For the `SafeSAC` agent, we set the safety margin ϵ to be 4.2, which is selected based on empirical safety performance of the `SafeRandom` agent to account for unmodeled dynamics. The `SPAR` agent uses a less conservative safety margin of 3.0, since it is going to refine the safety value function, over time. For the sake of comparison, we also provide results of `SafeSAC` with the same safety margin as `SPAR`.

Experimental Results

The performance comparison between different agents is summarised in Fig. 7.9 and Table 7.8.

Safety value function, pre-computed with the nominal model, provides a good initial estimate. While the kinematic vehicle model is a significant simplification of realistic dynamics, it provides a reasonable initial estimate that can ensure that a random agent can remain on the racetrack, given a sufficiently-large margin (Table 7.8).

Finding good safety margin: `SafeRandom` performance. Recall that the `SafeRandom` agent takes random actions and uses the safety value function precomputed from the nominal model. The optimal safety controller intervene whenever the safety value of the current state falls belong the safety margin. The safety margin is necessary because 1) the nominal model is a significant over-simplification of vehicle dynamics, and 2) the HJ Reachability computation does not take into consideration of the physical dimension of the vehicle. The performance of the `SafeRandom` agent at different safety margin is summarised in Figure 7.8. For safety margin $\epsilon \geq 4.2$, the `SafeRandom` agent can reliably complete laps, and thus we use $\epsilon = 4.2$ as the safety margin for the `SafeSAC` agent. On the other hand, the performance decrease drastically when the safety margin is reduced to 3.

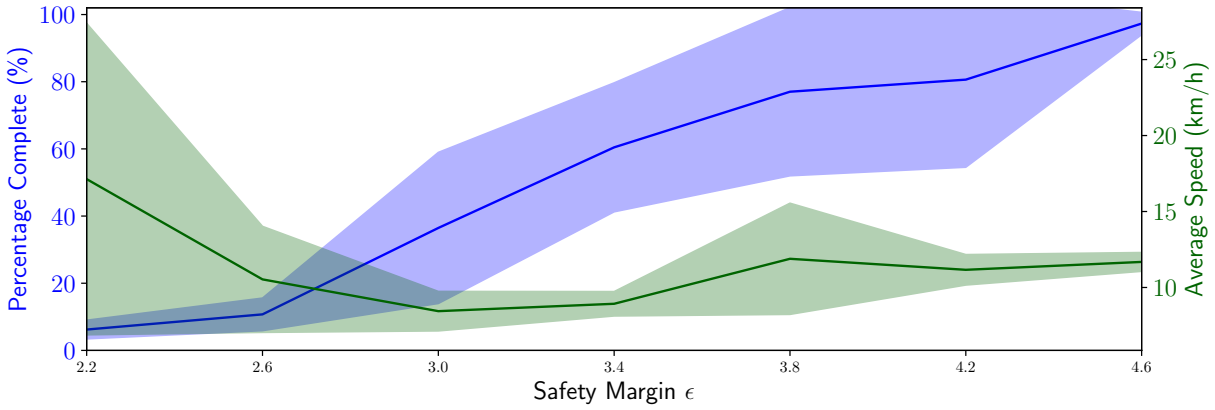


Figure 7.8: Performance of the `SafeRandom` agent at different safety margin (averaged over 10 random seeds)

Static safety backup controllers with large safety margin lead to overly conservative policies. Without the ability to perform dynamic updates, `SafeSAC` depends on a conservative safety margin in order to remain safe. Unsurprisingly, over-conservatism compromises performance. Despite its high average episode completion percentage (ECP), `SafeSAC` agents exhibit extremely low speeds. This conservatism even compromises ECP in the end, as episodes terminate early due to non-progression.

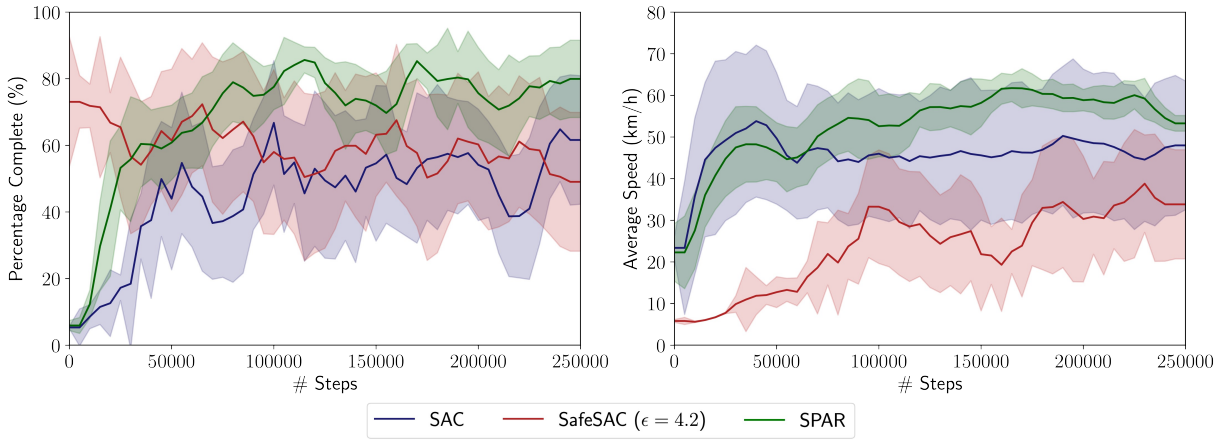


Figure 7.9: **Left:** Episode percent completion and **Right:** speed evaluated every 5000 steps over an episode (a single lap) and averaged over 5 random seeds. Results reported based on `Track01:Thruxton` in the `Learn-to-Race` environment (Section 7.2.3). For policies with static safety backup, we use safety margin of 4.2; for those with dynamic backup (i.e., `SPAR`), a safety margin of 3.0 was used.

Table 7.6: Performance of the Pre-trained Safety Critic

	F1	Precision	Recall
$V_S(\mathbf{x}) \leq 3$	0.65	0.74	0.59
$V_S(\mathbf{x}) \leq 4$	0.83	0.78	0.88
$V_S(\mathbf{x}) \leq 5$	0.96	0.96	0.96

SPAR agent learns to avoid unsafe scenarios, while driving reasonably fast. As the `SPAR` agent gains more experience, it learns to avoid unsafe scenarios, and thus the ECP ramps up very quickly, along with the speed. For example, at initialisation the agent tends to collide with a sidewall—perhaps due lower visual contrast between the track and the inadmissible region. However, it quickly learns that those states are unsafe. It is worth-noting that the speed of the `SPAR` agent start to decrease after 50K steps, which we hypothesise is due to increased awareness of unsafe scenarios and more frequent braking as a result. While `SPAR` outperforms other baselines, there is still significant performance gap with `HUMAN`, which calls for further research.

Pre-training the safety critic. We pre-trained the safety critic on 300,000+ state-action pairs collected by the `SafeRandom` agent. Recall the state consists of the front camera view and speed, and the action consists of the steering angle and acceleration. We regress the state-action pair to the safety value computed from the nominal model. After training for 10 epochs, we evaluate how well the safety critic classifies a state belonging to the sub-level set at 3, 4, and 5. We report the F1-score, precision, and recall in Table 7.6. Since the `SPAR` agent uses a safety margin of 3, the performance of classifying $V_S(\mathbf{x}) \leq 3$ is of the most importance. After pre-training, the safety critic misses 41% of the unsafe scenario, leaving much for `SPAR` to learn. The later observation, that `SPAR` learns to identify unsafe scenarios that it initially misses, indicates that `SPAR` is indeed learning the safety value function.

Interventions and violations for safe policies. At a high level, policies that frequently traverse through unsafe states require more *interventions* from the safety backup controller. Agents that perform unre-

coverable actions that lead to, e.g., leaving the drivable area or colliding with objects in the environment are said to commit *violations*. We report results on the basis of interventions per total distance travelled (in kilometers) and violations per total distance travelled, as functions of the number of agent steps, for the `SPAR` and `SafeSAC` agent classes; for both metrics, lower is better. Aggregating across 6 random seeds, `SPAR` agents receive an average of 1.37 interventions/km and commit 0.11 violations/km, while `SafeSAC` agents received 2.41 interventions/km and commit 0.22 violations/km. Figure 7.10 illustrates the number of interventions from safety backup controller, received by the `SPAR`and `SafeSAC` policies, throughout training, while Figure 7.11 illustrates the number of violations committed by the corresponding policy class. We observe that `SPAR`, with its dynamic safety updates, requires overall fewer interventions over time and commits fewer violations, compared to `SafeSAC`. Interestingly, in both cases, `SPAR` experiences a slight jump in early phases, due to the warm-start initialisation based on pre-training on context provided by a nominal dynamics model.

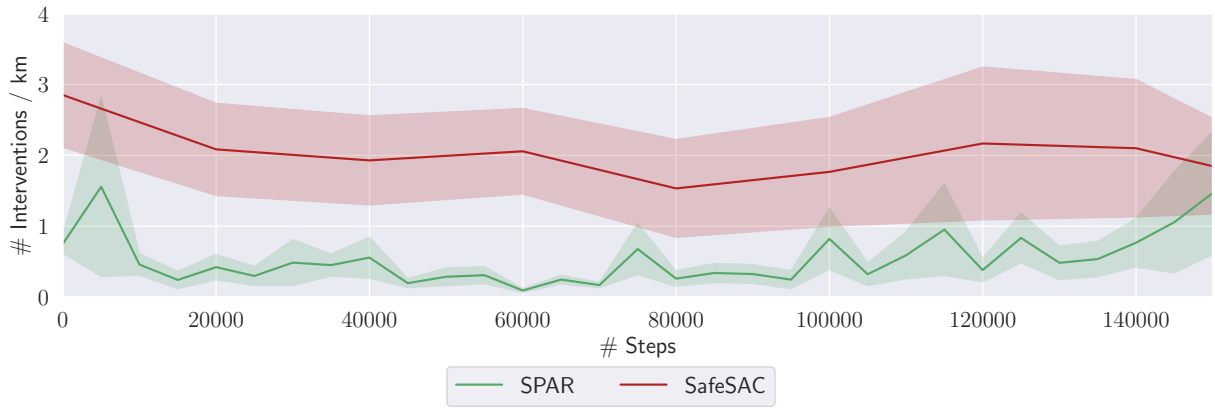


Figure 7.10: Plot of interventions/km versus steps, for policies that are coupled with safety backup controllers.

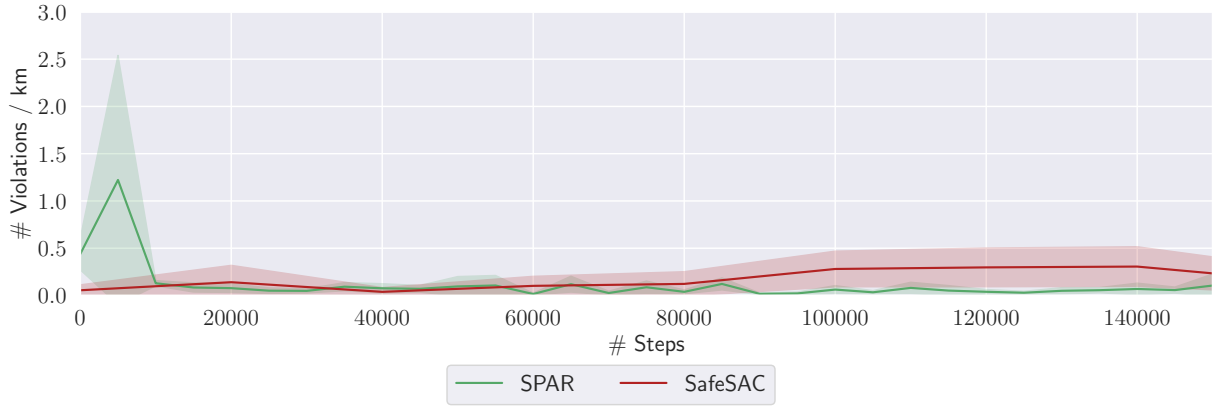


Figure 7.11: Plot of violations/km versus steps, for policies that are coupled with safety backup controllers.

SafeSAC & `SPAR` performance with same safe margin. While we choose the safety margin ϵ based on performance of the `SafeRandom` agent over a range of margins and our best engineering judgement, some may wonder if the superior performance of `SPAR` over `SafeSAC` may be attributed to the use of different safety margins. Thus, we also show here the performance of a `SafeSAC` agent with the same

safety margin as SPAR in Figure 7.12. Given the smaller safety margin, the ECP is low initially, which is inline with the observation from SafeRandom. Furthermore, the ECP barely improves over time. As the performance agent learns to drive faster, it is increasingly difficulty for the static actor-critic to catch the vehicle in marginally safe states.

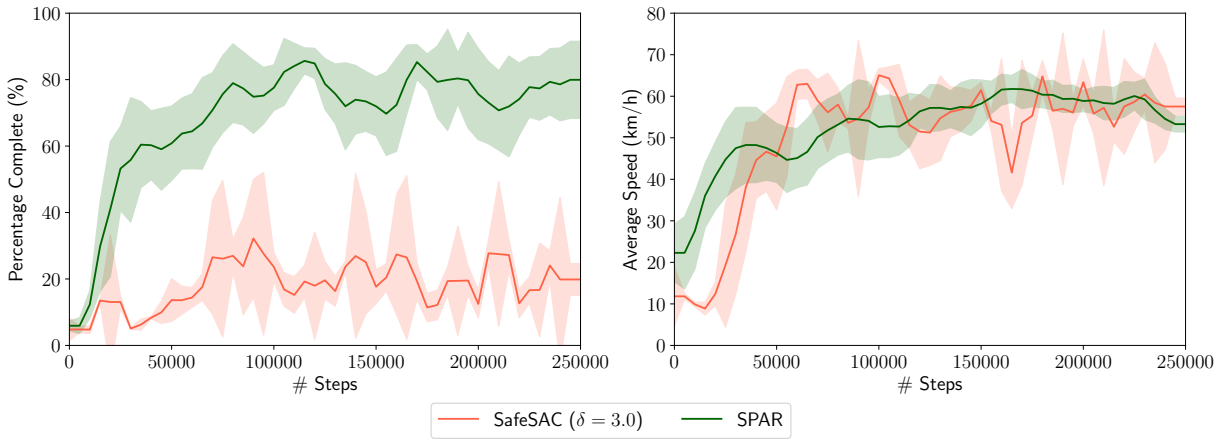


Figure 7.12: Performance of SafeSAC ($\epsilon = 3$) with comparison to SPAR

Learn-to-Race benchmark results. In tables 7.7 and 7.8, we report on all of their driving quality metrics, for the Learn-to-Race benchmark: Episode Completion Percentage (ECP), Episode Duration (ED), Average Adjusted Track Speed (AATS), Average Displacement Error (ADE), Trajectory Admissibility (TrA), Trajectory Efficiency (TrE), and Movement Smoothness (MS).

Table 7.7: Learn-to-Race task (Section 7.2.3) results on Track01 (Thruxton Circuit), for learning-*free* agents, with respect to the task metrics: Episode Completion Percentage (**ECP**), Episode Duration (**ED**), Average Adjusted Track Speed (**AATS**), Average Displacement Error (**ADE**), Trajectory Admissibility (**TrA**), Trajectory Efficiency (**TrE**), and Movement Smoothness (**MS**). Arrows (↑↓) indicate directions of better performance, across agents. **Bold** results in tables 7.7 and 7.8 are generally best, however, asterisks (*) indicate metrics which may be misleading, for incomplete racing episodes.

Agent	**ECP** (↑)	**ED*** (↓)	**AATS** (↑)	**ADE** (↓)	**TrA** (↑)	**TrE** (↑)	**MS** (↑)
HUMAN	**100.0** $\pm$ 0.0	78.6 $\pm$ 5.2	**79.29** $\pm$ 4.7	2.4 $\pm$ 0.1	0.93 $\pm$ 0.01	**1.00** $\pm$ 0.02	**11.7** $\pm$ 0.1
Random	0.50 $\pm$ 0.30	**4.67** $\pm$ 3.2	11.90 $\pm$ 3.80	1.5 $\pm$ 0.60	0.81 $\pm$ 0.04	0.33 $\pm$ 0.38*	6.7 $\pm$ 1.1
MPC	100.0 $\pm$ 0.0	301.40 $\pm$ 10.10	45.10 $\pm$ 0.0	**0.90** $\pm$ 0.10	**0.98** $\pm$ 0.01	0.85 $\pm$ 0.03	10.4 $\pm$ 0.60

We highlight the fact that such metrics as TrA, TrE, and MS are most meaningful for agents that *also* have high ECP results. Taking TrA, for example, safe policies score higher ECP values but may spend more time in inadmissible positions (as defined by the task, i.e., with at least one wheel touching the edge of the drivable area), compared to policies without a safety backup controller that may quickly terminate episodes by driving out-of-bounds (thus spending less time in the inadmissible positions). On the other hand, policies that have low completion percentages also have low ED scores, due to more frequent failures and subsequent environment resets.

We observe new state-of-the-art performance received by our approach, across the driving quality metrics, in the Learn-to-Race benchmark.

Table 7.8: `Learn-to-Race` task (Section 7.2.3) results on `Track01` (Thruxton Circuit), for learning-*based* agents.

Agent	ECP (↑)	ED* (↓)	AATS (↑)	ADE (↓)	TrA (↑)	TrE (↑)	MS (↑)
`SAC`	61.61 ± 38.57	272.75 ± 256.51	47.99 ± 30.9	1.54 ± 1.07	**0.94** ± 0.02	**0.28** ± 0.12	11.84 ± 2.12
`SafeRandom` (ours), $\delta = 3.0$	36.46 ± 23.71	654.37 ± 447.05	8.44 ± 1.37	3.93 ± 0.21	0.81 ± 0.10	0.00 ± 0.00	13.21 ± 1.88
`SafeRandom` (ours), $\delta = 4.2$	63.63 ± 39.46	761.80 ± 494.65	11.68 ± 1.07	2.74 ± 0.16	0.90 ± 0.07	0.02 ± 0.01	**13.63** ± 2.01
`SafeSAC` (ours), $\delta = 3.0$	25.70 ± 11.31	66.90 ± 23.22	49.67 ± 3.34	1.35 ± 0.05	0.86 ± 0.06	0.14 ± 0.05	8.46 ± 2.35
`SafeSAC` (ours), $\delta = 4.2$	49.05 ± 41.66	617.52 ± 842.49	33.83 ± 26.21	1.80 ± 0.63	0.91 ± 0.12	0.07 ± 0.11	10.03 ± 2.75
`SPAR` (ours)	**79.94** ± 23.20	**59.19**± 29.99	**53.28**± 3.76	**0.99**± 0.17	0.91 ± 0.03	0.22 ± 0.03	9.27 ± 1.68

7.2.6 Related Work

Autonomous racing. One approach for autonomous racing is via model predictive control [133, 170, 245], which solves an optimisation problem with a model of the system dynamics. Aside from the challenges in modelling the complex dynamics, a significant drawback of such approach is the dependence on extensive sensor installation for localisation and state estimation [38]. Another approach is to use a modular pipeline [133, 267], starting from perception on raw sensory inputs, to localisation and object-detection, and finally to planning and control. While this approach is most commonly used in practice, disadvantages of the approach include over-complexity and error propagation [94, 322]. Recently, there is a lot of interest in using RL-based approaches for autonomous racing. In [59, 92], RL agents were trained using low-dimensional features as inputs. In [55, 78], intermediate features were extracted from perception pipelines to determine control actions. In [38, 302], RL agents were trained end-to-end on visual inputs by imitating expert demonstration; in [38], a data-driven model of the environment was further utilised to train the agent by unrolling future trajectories. While RL-based approaches show considerable promise in the context of autonomous racing, their performance benefits come at the cost of unsafe exploration, especially during early phases of training. In comparison to racing, there is significantly more literature on end-to-end autonomous driving for urban scenarios [50, 64, 65, 209, 226, 326, 326, 328]. It is beyond our scope to cover this large research field, and we refer interested readers to survey papers, such as [107, 322], for more information. While we focus on high-speed racing and its unique challenges, we believe the discussion here for safety analysis on ego-vision is also relevant to urban driving.

Constrained reinforcement learning. There is growing interest in enforcing some notion of safety in RL algorithms, e.g., satisfying safety constraints, avoiding worst-case outcomes, or being robust to environmental stochasticity [98]. Some examples include methods that bound expected costs characterising violations of safety constraints under the CMDP formulation [2, 315], methods that combine control theory and RL [58, 61, 91, 294], and methods that enable the agent to explore with probabilistic safety guarantees under uncertain environment [72, 126, 153]. We focus on the notion of safety as satisfying constraints. CMDP [6] is a widely-used framework for studying RL under constraints, where the agent maximises cumulative rewards, subject to limits on cumulative costs characterising constraint violations. Solving a CMDP problem is challenging, because the policy needs to be optimised over the set of feasible states; this requires off-policy evaluation of the constraint functions, to determine whether a policy is feasible [2]. As a result, safety grows with experience, but requires diverse state-action pairs, including unsafe ones [265]. Furthermore, one needs to solve a constrained optimisation problem with a non-convex neural policy. This may be implemented with techniques inspired by convex optimisation, such as primal-dual updates [26] and projection [315], or by upper bounding the expected cost at each policy iteration [2]. Most relevant to our work is [26, 265, 282], which also uses a safety critic to verify if a state is safe.

Guaranteed safe control. Guaranteeing the safety of general continuous nonlinear systems is challeng-

ing, but there are several approaches that have been successful. These methods typically rely on knowledge of the environment dynamics. Control barrier functions (CBFs) provide a measure of safety with gradients that inform the acceptable safe actions [7]. For specific forms of dynamics, e.g., control-affine [58], and unlimited actuation bounds, this approach can be scalable to higher-dimensional systems and can be paired with an efficient online quadratic program for computing the instantaneous control [58]. Unfortunately, finding a valid control barrier function for a general system is a nontrivial task. Lyapunov-based methods [61, 62] suffer from the same limitation of requiring hand-crafted functions.

HJ reachability is a technique that uses continuous-time dynamic programming to directly compute a value function that captures the optimal safe control for a general nonlinear system [19, 90]. This method can provide hard safety guarantees for systems, subject to bounded uncertainties and disturbances. There are two major drawbacks to HJ reachability. The first is that the technique suffers from the curse of dimensionality and scales exponentially with number of states in the system. Because of this, the technique can only be used directly on systems of up to 4-5 dimensions [19]. When using specific dynamics formulations and/or restricted controllers, this upper limit can be extended [57, 148]. Second, because of this computational cost, the value function is typically computed offline based on assumed system dynamics and bounds on uncertainties. This can lead the safety analysis to be invalid or overly conservative.

There are many attempts in injecting some form of control theory into RL algorithms. In comparison to works that assume specific problem structure [58, 72] or existence of a nominal model [20, 58], our proposed approach is applicable to general nonlinear systems and does not require a model. But, we do assume access to a distance metric defined on the state space. Our primary inspiration is recent work by [91] that connects HJ reachability with RL and introduces a HJ Bellman update, which can be applied to deep Q-learning for safety analysis. This method loses hard safety guarantees due to the neural approximation, but enables scalable learning of safety value function. However, an agent trained using the method in [91] will focus exclusively on safety. Thus, we extend the method by formulating it within the CMDP framework, thereby enabling performance-driven learning.

7.3 Conclusion

This chapter studied the use of constraint functions as domain knowledge in autonomous driving settings. We observed that technologies which are to be applied to safety-critical applications must adhere to safety constraints, throughout their interactions with their environments, as any safety infraction in urban/highway driving or high-speed racing, could lead to catastrophic failures. Moreover, their training environments must capture sufficient realism (e.g., in visual rendering, vehicular dynamics, and task objectives), in order for the methods to be eligible for simulation-to-real transfer. The chapter starts by addressing the lack of realistic simulation for producing safety-aware methods: we introduce the `Learn-to-Race` (`L2R`) framework as a particularly challenging proving ground for safe learning algorithms. `L2R` is OpenAI-gym compliant training environment for simulated *Formula*-style racing, which enables agents to learn to race on high-precision models of real-world tracks (e.g., the famed Thruxton Circuit and the Las Vegas Motor Speedway) and to use a suite of rich multimodal sensory information, predicated on accurate vehicle dynamics. We additionally define the L2R AI task, by introducing two (2) objectives and seven (7) metrics that characterise and measure performance, safety, and desirable agent behaviour. We additionally provide an official `L2R` task dataset of expert demonstrations and a series of baseline experiments and reference implementations. In this task, algorithms must learn to control vehicles at their physical limits, with minimal margins for safety, while making sub-second decisions in a fast-changing environment and while remaining robust to distribution shifts and novel road features. Next, to address the challenges in safe policy optimisation, we propose the use of safety constraints for autonomous racing, inspired by the theoretical foundations of Hamilton-Jacobi (HJ) reachability analysis in optimal control. Here, we define a safety controller that intervenes whenever an agent approaches bad states, and we show that even an agent that generates actions at random is guaranteed to stay on the drivable area; we further show that an arbitrary learning policy that is coupled with this safe controller is able to learn performant driving behaviour, both safely and sample-efficiently. Finally, we demonstrate that the HJ safety value can be learned and updated directly from vision context, thereby expanding HJ reachability to applications where high-fidelity dynamics models may not be available. While not necessary for convergence, we warm-start the safety value function using values pre-computed with a nominal model. Next, the value function is updated directly on transitions of ego-agent's frontal camera view and vehicle speed. As a first experiment, we evaluate our approach on alongside strong baselines, in two environment and agent configurations, on the OpenAI Safety Gym framework; we report the minimum number of safety infractions, compared to state-of-the-art constrained Markov Decision Process approaches. We also evaluate our approach on the `L2R` benchmark task and report state-of-the-art results: we show that incorporating a dynamically updating safety critic grounded in control theory boosts performance especially during the initial learning phase.

Chapter 8

Conclusion

This thesis studied the avenues by which various types of domain knowledge can be combined with neural representations, across multiple problem domains. We saw that this fusion between domain knowledge and neural representations could be used to recover information about the underlying generating process, to design effective modelling strategies in learning problems, to ensure model transferability or generalisability, or to understand the complementarity between modalities or views. Fundamentally, this fusion gives us the best of both *knowledge-driven* methods and *data-driven* methods—e.g., interpretability and sample-efficiency from the former as well as the ability to model statistical regularity of events and accommodate some input perturbations from the latter. While there had been much focus on learning effective neural representations for specific tasks, then transferring or adapting the learned representations to other tasks, comparatively less focus had been placed on representation learning in the presence of various types of domain knowledge.

8.1 Summary of Contributions

We studied domain knowledge injection in neural systems, across neural commonsense reasoning, multimodal robot navigation, and autonomous driving; we developed methods that enable Learning with Common Sense, Learning with Primitives, Learning with Distribution-awareness, and Learning with Constraints. We developed extensible neural architectures that flexibly incorporate domain knowledge into their perceptual pipelines and/or training objectives, highlighting: (i) commonsense knowledge grounding (for extraction) and attention based combination with neural context (injection) for multiple-choice question answering in natural language processing; (ii) understanding the alignment between the downstream task and the types of commonsense knowledge that would facilitate improved optimisation and performance; (iii) leveraging domain knowledge as a statistical prior, with careful selection of modelling strategies and objectives, in the context of learning embedded skills for robot navigation, multi-agent trajectory prediction, and goal-prediction in autonomous driving; (iv) grounding neural predictions on physics-based models to ensure conformance to vehicle dynamics and driveable-area geometries; (v) leveraging domain knowledge for modularity/hierarchicality, in multimodal perception pipelines, for robot navigation and autonomous driving; (vi) performing knowledge transfer to unseen domains, e.g., in the context of zero-shot evaluation for commonsense question answering on unseen datasets and transfer of learned primitives for robot instruction-following; and (vii) combining neural models with constraints derived from optimal control, in the context of safety-aware autonomous racing. This thesis contributed a collection of tools,

methodologies, tasks, international AI challenges and leaderboards, datasets, and knowledge graphs; additionally, this work led to the successful organisation of two international workshops in safe learning for autonomous driving.

8.2 Directions for Future Work

Towards Generalised/Reusable Commonsense Representations (Sections 4.2, 4.3, 5.2)

In Chapter 4, studied symbolic commonsense knowledge and how it is organised into different dimensions (or 'domains'), on the basis of different commonsense knowledge types (e.g., declarative knowledge, procedural knowledge, relational knowledge); we also characterised the mechanisms and representations that yield generalisable neuro-symbolic architectures, and we defined strategies for pre-training models on knowledge-rich context for subsequent zero-shot evaluation. Later, in Section 5.2, we leveraged these insights in pre-training with relational commonsense knowledge, specifically, in order to generate knowledge-driven scene priors for audio-conditioned visual robot navigation. We observed a consistent leap in accuracy, across tasks, when the appropriate commonsense domain knowledge was combined with models (e.g., declarative common sense for question-answering, procedural common sense for cause-effect resolution and story understanding, relational common sense for spatial navigation, etc.). Indeed, several language modelling and embodied vision-language planning tasks actually require models to incorporate external commonsense knowledge about the world in which they operate, in order to solve those tasks and to generalise effectively. Fortunately, commonsense knowledge (as a whole) evolves slowly, compared to knowledge about specific entities and events, which changes rapidly. Thus, a natural question arises: can we develop generalised and knowledge-grounded, multimodal representations that can be leveraged across diverse tasks? This may be achieved through: (i) a series of pre-training tasks which seek to inform models about the axioms and nuances of the different commonsense knowledge types; (ii) a careful choice of inductive bias and architecture (e.g., multimodal transformers) that may represent and align the various types of commonsense knowledge with multimodal sensory information, in large classes of problems (e.g., navigation, manipulation, dialogue); (iii) generating and utilising a wider variety of commonsense knowledge resources; and (iv) designing internal modelling mechanisms that perform auxiliary objectives of classifying the type of reasoning paradigm (e.g., cause-effect, spatial reasoning) and commonsense knowledge type required (procedural, relational), given an observation.

Importance of Knowledge-Task Alignment (Sections 4.2, 4.3, 5.2)

Based on our experimental results and error analysis in Chapter 4, we see that external commonsense knowledge is only helpful when there is alignment between questions and knowledge-base types. Thus, it is crucial to identify the task type and apply the best-suited knowledge, accordingly. In terms of knowledge-integration methods, attention-based injection seems to be the better choice for pre-trained language models: even when alignment between knowledge-base and dataset is sub-optimal, the performance would not degrade. On the other hand, pre-training on (e.g., sub-optimal) knowledge would shift the language model's weight distribution toward its own domain greatly, where, if the task domain does not fit knowledge-base well, model performance is likely to drop. When the domain of the knowledge-base aligns with that of the dataset perfectly, both knowledge-integration methods bring performance boosts and a combination of them could bring further gain. We surveyed popular knowledge bases and recent knowledge-integration methods on commonsense QA tasks, and evaluation on these tasks confirm that alignment between knowledge-bases and datasets plays a crucial role in knowledge-integration. We

believe it is worth conducting a more comprehensive study of datasets and knowledge-bases and putting more effort towards defining an auxiliary learning objective that identifies the type of knowledge required, based on data characteristics. In Chapter 5, we examine knowledge-task alignment in the context of generating and using scene priors for robot navigation; for this, we use semantic information in the form of object-object, object-region, and region-region relational knowledge. Our experimental results provide additional confirmation of the importance of knowledge-task alignment, where, like the work shown in Section 4.3, we show improvements here over strong baselines in generalisation to unseen environments. For future work in this area, we plan to incorporate additional semantic knowledge about sounds, objects, and interactions in our external domain knowledge resource to further improve performance. Through this, we envision the creation of versatile scene priors, leveraged across various embodied vision-language planning (EVLP) tasks and robot morphologies.

Towards Unified Robot Learning (Sections 4.3, 5.2, 5.3)

In Chapter 4, we performed zero-shot evaluation of models that had been pre-trained with commonsense knowledge and task structural cues, in order to assess their capabilities in unseen contexts; in Chapter 5, we learned robot navigation primitives for improved transfer generalisation. In general, however, tasks are currently evaluated separately from one another, which fails to capture the underlying skills shared between tasks and the overall progress made as a field. Being able to follow instructions (e.g., as in the vision-and-language navigation [9]) and also answer questions about an environment (e.g., as in embodied question-answering [68]) are not and should not be treated as mutually-exclusive tasks. Rather than propose and evaluate tasks independently, tasks could be combined according to more unifying requirements on agent capability. Similar notions of cross-task learning gave rise to seminal modelling strategies (e.g., hierarchical task decomposition) and problem definitions (e.g., mobile manipulation), which remain quite relevant to this day, despite their classical foundations. Combinations of existing task families may, likewise, yield new insights and advances.

Towards Improved Multimodal Grounding (Sections 5.2, 5.3, 6.2)

Research in Embodied Vision-Language Planning (EVLP) tasks has experienced a quick rise in popularity, due to recent advances from the robotics, computer vision, and natural language processing communities on vision-language grounding and situated learning. Despite recent advances, EVLP tasks such as audio-visual navigation (AVN) and vision-language navigation (VLN) remain challenging, as agents must reconcile multimodal goal descriptions or satisfy complex natural language instructions, and navigate within complex photorealistic, partially-observable environments. Recent pursuit of these tasks has focused on progress-monitoring and cross-modal grounding; however, issues with instruction complexity and label-independent visual feature variation still persist, reducing agents' unseen generalisation performance. In Chapter 5, we introduced frameworks for leveraging knowledge-enhanced scene priors and for learning reusable robot primitives, and we achieved improvements in unseen generalisability in both AVN and VLN tasks. An interesting direction for future work is in exploring alternative bases for cross-modal grounding, e.g., mapping visual observations to domain-invariant state representations (navigation graphs, infused with semantic information) and mapping noisy natural language instructions to structured forms, where the goal is to encourage agents to learn stopping and recovery actions and to avoid bad states.

Towards Effective Transfer Imitation Learning (Sections 6.2, 6.3)

In Chapter 6, we introduced frameworks for trajectory forecasting and goal-prediction in urban driving, which coupled imitation learning (IL) objectives with knowledge of the underlying action prior distribution; we hypothesised that this distribution-awareness would provide agents with robustness to noise artifacts in the training data, would provide a window into agents' intentions for more interpretable predictions, would yield improved unseen generalisation, and would help bypass common issues in imitation- and transfer learning such as causal confusion and negative transfer. Indeed, we showed substantial improvements in the diversity and admissibility of our agent's predictions in multi-agent trajectory forecasting, through density estimation of the underlying action distribution, conditioned on agent-to-agent and scene-to-agent context; we showed significant improvements in unseen generalisation in visuomotor control for urban driving, by jointly estimating the ego-agent's action distribution while learning to predict and score intermediate goals. An interesting direction for future work is in the selection of more informative priors: whereas we chose an annotation-free differentiable drivable-area map as the prior in Section 6.2, we may consider selecting a prior that incorporates dense annotation of road rules, conditioned on, e.g., traffic pattern/layout. Another important direction for future work is in characterising the causal relationships between observations, actions, and rewards in a scene. This causal structure can be represented *explicitly*, by way of causal knowledge graphs (enabling explainability and counterfactual reasoning) then lexicalised as in Chapter 4, or it can be represented *implicitly*, by way of learning identifiable latent representations. Regardless of explicit or implicit representation, the goal would be to eliminate extraneous connections/edges (disturbances, confounders) in the underlying causal structure, thereby reducing causal confusion when transferring an observation model from a source domain to a target domain.

Towards Provably Conservative Safety Constraints (Sections 7.2)

In Chapter 7, we introduced a method that enables an agent to learn safety-aware and performance-oriented behaviour: instead of solving a constrained optimisation problem, we effectively decomposed the problem of learning under safety constraints into two more-tractable sub-tasks—optimising for performance and updating safety value. Alternative methods for specifying safety constraints include the use of first-order logical rules (see Section 2.1.3), which may characterise desired vehicle behaviour (e.g., adherence to the driveable area, protocols for interacting with other agents on the road, etc.), despite significant cost in annotator effort. Inspired by optimal control, we demonstrated in Section 7.2 that an approximate HJ safety value can be learned directly on visual context, thereby expanding HJ reachability to applications where dynamics models may not be available. Whereas our empirical results demonstrated that it is possible to learn a safety-aware and performant policy, through approximation of the minimum payoff distance and through subsequent dynamic updates to the safety value function, one limitation of our method is that it is by no means free from failure, due to the use of neural functional approximation. However, the method proposed in this paper represents a subtle shift away from safety constraint-satisfaction exclusively through model-free exploration, as has become popular in the recent literature. Rather than letting agents learn safe behaviours through experiencing failures, our approach provides avenues for potential online safety analysis, as in the injection of domain knowledge (when available; e.g., using a nominal dynamics model for warm-starting safety value estimation). An important direction for future work is in learning a provably-conservative estimation of the safety value function, which can retain hard safety guarantees (throughout value updates), while neural functional approximation is still utilised for problems with high-dimensional state.

Acknowledgements

The material in this this dissertation is based on work partially supported by the *Bosch Doctoral Fellowship for Artificial Intelligence Research*. The views expressed in this article do not necessarily represent those of the aforementioned entity, and no official endorsement should be inferred.

Bibliography

[1] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. *OpenAI Technical Report*, 2018. 7.2.1

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017. 2.1.3, 7.2, 7.2.2, 7.2.6

[3] Sungsoo Ahn, Shell Xu Hu, Andreas C. Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. *CoRR*, abs/1904.05835, 2019. URL `http://arxiv.org/abs/1904.05835`. 2.1.4

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 6.2.7

[5] Patricia A Alexander and Jonna M Kulikowich. Domain knowledge and analogic reasoning ability as predictors of expository text comprehension. *Journal of Reading Behavior*, 23(2):165–190, 1991. 2.1

[6] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999. 7.2, 7.2.6

[7] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019. 2.1.3, 7.2.6

[8] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*, 2018. 6.3.5, 7.2.3

[9] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. (document), 2.2.2, 5.2.5, 5.3.1, 5.3.2, 5.3.4, 5.3.4, 5.3.4, 5.4, 5.5, 5.3.5, 5.6, 8.2

[10] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *CoRR*, abs/1611.01796, 2016. URL `http://arxiv.org/abs/1611.01796`. 5.3.5

[11] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015. URL `http://arxiv.org/abs/1505.00468`. 5.3.5

[12] Amina Asif, Muhammad Dawood, and Fayyaz ul Amir Afsar Minhas. A generalized meta-loss function for regression and classification using privileged information, 2018. 2.1.4

[13] Samet Ayhan and Hanan Samet. Aircraft trajectory prediction made easy with predictive analytics. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 21–30, 2016. 6.2.7

[14] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*, 2015. 2.1.3

[15] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet. A robust and sensitive metric for quantifying movement smoothness. *IEEE Transactions on Biomedical Engineering*, 59(8):2126–2136, 2012. doi: 10.1109/TBME.2011.2179545. 7.2.3

[16] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2): 423–443, 2018. 1

[17] Pratyay Banerjee and Chitta Baral. Self-supervised knowledge triplet learning for zero-shot question answering. *ArXiv*, abs/2005.00316, 2020. 4.3, 4.10, 4.3.6

[18] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 6.2.7, 6.3.9

[19] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017. 2.1.3, 2.1.3, 7.2.6

[20] Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American Control Conference (ACC)*, pages 3488–3494. IEEE, 2021. 7.2.1, 7.2.6

[21] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. In *arXiv:2006.13171*, 2020. 5.2.5

[22] Lisa Bauer, Yicheng Wang, and Mohit Bansal. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1454. URL `https://www.aclweb.org/anthology/D18-1454`. 4.2.4, 4.2.7, 4.3.6

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 1

[24] Dennis S Bernstein and Wasin So. Some explicit formulas for the matrix exponential. *IEEE Transactions on Automatic Control*, 38(8):1228–1232, 1993. 6.2.2

[25] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*, 2019. 1

[26] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020. 7.2, 7.2.1, 7.2.6

[27] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7432–7439, 2020. 4.3, 3

[28] BMW. Automotive sensors – the sense organs of driver assistance systems, Sep 2021. URL https://www.bmw.com/en/innovation/automotive-sensors.html. 7.2.1

[29] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 6.3, 6.3.9

[30] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013. 2.2.1

[31] Piotr Borkowski. The ship movement trajectory prediction algorithm using navigational data fusion. *Sensors*, 17(6):1432, 2017. 6.2.7

[32] Antoine Bosselut and Yejin Choi. Dynamic knowledge graph construction for zero-shot commonsense question answering. *ArXiv*, abs/1911.03876, 2019. 4.10, 4.3.6

[33] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1470. 4.2.3

[34] Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E Peters, Ashish Sabharwal, and Yejin Choi. Adversarial filters of dataset biases. *arXiv preprint arXiv:2002.04108*, 2020. 4.3

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. 7.2.3, 7.2.3

[36] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018. 5.3.4

[37] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 4, 6.2.7

[38] Peide Cai, Hengli Wang, Huaiyang Huang, Yuxuan Liu, and Ming Liu. Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters*, 6 (4):7262–7269, 2021. 7.2.6

[39] Salvatore Candido, James Davidson, and Seth Hutchinson. Exploiting domain knowledge in planning for uncertain robot systems modeled as pomdps. In *2010 IEEE International Conference on Robotics and Automation*, pages 3596–3603. IEEE, 2010. 2.1

[40] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262031361. 5.2.5

[41] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 6.2.7

[42] Cristiano Castelfranchi. Modelling social action for ai agents. *Artificial intelligence*, 103(1-2): 157–182, 1998. 1

[43] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor

environments. *International Conference on 3D Vision (3DV)*, 2017. 5.2.2, 5.2.4, 5.2.5, 5.3.4

[44] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 6.2, 6.2.5, 6.2.6, 6.2.7

[45] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *CoRR*, abs/1706.07230, 2017. URL http://arxiv.org/abs/1706.07230. 5.3.5

[46] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Russ R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2c75cf2681788adaca63aa95ae028b22-Abstract.html. 5.2.5

[47] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020. 6.3

[48] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological SLAM for visual navigation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 12872–12881. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01289. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Chaplot_Neural_Topological_SLAM_for_Visual_Navigation_CVPR_2020_paper.html. 5.2.5

[49] Bingqing Chen, Jonathan Francis, Marco Pritoni, Soummya Kar, and Mario Bergés. Cohort: Coordination of heterogeneous thermostatically controlled loads for demand flexibility. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 31–40, 2020. 3

[50] Bingqing Chen, Weiran Yao, Jonathan Francis, and Mario Berges. Learning a distributed control scheme for demand flexibility in thermostatically controlled loads. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE, 2020. 3, 6.3, 6.3.7, 6.3.7, 6.3.9, 6.9, 7.2.3, 7.2.6

[51] Bingqing Chen, Jonathan Francis, Jean Oh, Eric Nyberg, and Sylvia L Herbert. Safe autonomous racing via approximate reachability on ego-vision. *arXiv preprint arXiv:2110.07699*, 2021. 2.1.3

[52] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, volume 12351 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2020. doi: 10.1007/978-3-030-58539-6_2. URL https://doi.org/10.1007/978-3-030-58539-6_2. 5.2.4, 5.2.4, 5.3, 5.2.5

[53] Changan Chen, Ziad Al-Halah, and Kristen Grauman. Semantic audio-visual navigation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021,*

virtual, June 19-25, 2021, pages 15516–15525. Computer Vision Foundation / IEEE, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Semantic_Audio-Visual_Navigation_CVPR_2021_paper.html. 5.1, 5.2.1, 5.2.2, 5.2.4, 5.2.4, 5.2.4, 5.3, 5.2.5

[54] Changan Chen, Sagnik Majumder, Ziad Al-Halah, Ruohan Gao, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Learning to set waypoints for audio-visual navigation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=cR91FAodFMe. 5.2.4, 5.2.5, 6.3

[55] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015. 7.2.1, 7.2.6

[56] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2.2.1

[57] Mo Chen, Sylvia L Herbert, Mahesh S Vashishtha, Somil Bansal, and Claire J Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018. 2.1.3, 7.2.6

[58] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019. 2.1.3, 7.2, 7.2.1, 7.2.6

[59] Eugenio Chisari, Alexander Liniger, Alisa Rupenyan, Luc Van Gool, and John Lygeros. Learning from simulation, racing in reality. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8046–8052. IEEE, 2021. 7.2.6

[60] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2.2.1

[61] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *arXiv preprint arXiv:1805.07708*, 2018. 2.1.3, 7.2.6

[62] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019. 2.1.3, 7.2.6

[63] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018. 2.1.3

[64] Felipe Codevilla, Matthias M"uller, Antonio L'opez, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2018. 6.3, 6.3.7, 6.3.9, 6.9, 7.2.3, 7.2.6

[65] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019. 6.3, 6.3.9, 7.2.6

[66] W Bruce Croft. User-specified domain knowledge for document retrieval. In *Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval*,

pages 201–206, 1986. 2.1

[67] W Bruce Croft and Roger H Thompson. I3r: A new approach to the design of document retrieval systems. *Journal of the american society for information science*, 38(6):389–404, 1987. 2.1

[68] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1–10. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00008. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Das_Embodied_Question_Answering_CVPR_2018_paper.html. 2.2.2, 5.3.5, 8.2

[69] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation, 2020. 2.2.1

[70] Ernest Davis. *Representations of commonsense knowledge*. Morgan Kaufmann, 2014. 1, 4.1

[71] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning, 2019. 6.3, 6.3.9

[72] Sarah Dean, Stephen Tu, Nikolai Matni, and Benjamin Recht. Safely learning to control the constrained linear quadratic regulator. In *2019 American Control Conference (ACC)*, pages 5582–5588. IEEE, 2019. 7.2.6

[73] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018. 6.2.6, 6.1, 6.4, 6.2.7

[74] 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 2.2.1, 2.2.1, 4.2, 4.2.1, 4.2.6, 4.3.6

[75] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017. 7.2.3

[76] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, and Jeff Schneider. Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv preprint arXiv:1808.05819*, 2018. 6.2.7

[77] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 6.3.7, 7.2.3, 7.2.4

[78] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A Theodorou, and James M Rehg. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Conference on Robot Learning*, pages 133–142. PMLR, 2017. 7.2.6

[79] Heming Du, Xin Yu, and L. Zheng. Learning object relation graph and tentative policy for visual navigation. *ArXiv*, abs/2007.11018, 2020. 5.2.5

[80] David Ellis. New horizons in information retrieval. *The Library Association, London*, 1990. 2.1

[81] David Ellis. Paradigms and proto-paradigms in information retrieval research. *Conceptions of Library and Information Science. Historical, empirical and theoretical perspectives. London*, pages

165–186, 1992. 2.1

[82] Lyn English. Children's use of domain-specific knowledge and domain-general strategies in novel problem solving. *British Journal of Educational Psychology*, 62(2):203–216, 1992. 2.1

[83] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021. 7.2.3

[84] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 538–547. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00063. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Fang_Scene_Memory_Transformer_for_Embodied_Agents_in_Long-Horizon_Tasks_CVPR_2019_paper.html. 5.2.2, 5.2.3

[85] Kuan Fang, Alexander Toshev, Fei-Fei Li, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 538–547. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00063. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Fang_Scene_Memory_Transformer_for_Embodied_Agents_in_Long-Horizon_Tasks_CVPR_2019_paper.html. 2.2.1

[86] Martha J Farah. Is an object an object an object? cognitive and neuropsychological investigations of domain specificity in visual object recognition. *Current Directions in Psychological Science*, 1 (5):164–169, 1992. 2.1

[87] Panna Felsen, Patrick Lucey, and Sujoy Ganguly. Where will they go? predicting fine-grained adversarial multi-agent motion using conditional variational autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 732–747, 2018. 6.2.7

[88] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection. *CoRR*, abs/1702.05552, 2017. URL http://arxiv.org/abs/1702.05552. 6.2.7

[89] Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning (ICML)*, 2020. (document), 6.3, 6.3.7, 6.3.7, 6.3.8, 6.9, 6.3.9, 7.2.3

[90] Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018. 2.1.3, 7.2.6

[91] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019. 2.1.3, 7.2, 7.2.1, 7.2.1, 7.2.1, 7.2.2, 7.2.5, 7.2.6

[92] F. Florian, S. Yunlong, E. Kaufmann, D. Scaramuzza, and P. Duerr. Super-human performance in gran turismo sport using deep reinforcement learning, 2020. 7.2.3, 7.2.6

[93] Jonathan Francis, Matias Quintana, Nadine Von Frankenberg, Sirajum Munir, and Mario Bergés. Occutherm: Occupant thermal comfort inference using body shape information. In *Proceedings*

of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, pages 81–90, 2019. 3

[94] Jonathan Francis, Nariaki Kitamura, Felix Labelle, Xiaopeng Lu, Ingrid Navarro, and Jean Oh. Core challenges in embodied vision-language planning. *arXiv preprint arXiv:2106.13948*, 2021. 2.2.2, 5.2.5, 5.3.2, 5.3.4, 7.2.6

[95] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *arXiv preprint arXiv:1806.02724*, 2018. (document), 5.3.2, 5.3.3, 5.3.4, 5.5, 5.3.5, 5.6

[96] Chuang Gan, Yiwei Zhang, Jiajun Wu, Boqing Gong, and Joshua B Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9701–9707. IEEE, 2020. 5.2.5

[97] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010. 2.1.3

[98] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015. 7.2.6

[99] Peter Gärdenfors. *Knowledge in flux: Modeling the dynamics of epistemic states.* The MIT press, 1988. 1

[100] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018. 5.3.2

[101] Samuel Gershman, Matt Hoffman, and David Blei. Nonparametric variational inference. *arXiv preprint arXiv:1206.4665*, 2012. 2.1.2

[102] George Giovanis, Michael Lu, and Mo Chen. Optimizing dynamic programming-based algorithms. `https://github.com/SFU-MARS/optimized_dp`, 2021. 7.2.5

[103] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 6.2.7

[104] Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proc. of *SEM)*, pages 394–398, 7-8 June 2012. 4.3.5

[105] Daniel Gordon, Dieter Fox, and Ali Farhadi. What should I do now? marrying reinforcement learning and symbolic planning. *CoRR*, abs/1901.01492, 2019. URL `http://arxiv.org/abs/1901.01492`. 5.2.5

[106] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey, 2020. 2.1.4

[107] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. 7.2.6

[108] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 6.2.7

[109] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/haarnoja18b.html. 7.2.3

[110] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018. 7.2.1, 7.2.1, 7.2.5

[111] B. Hariharan. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2.2.1, 5.3.4

[112] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb. accepted as poster. 5.3.2, 5.3.3, 5.3.5

[113] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322. 2.2.1, 5.3.4

[114] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385. 5.2.3

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385. 2.2.1

[116] 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, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90. 2.2.1

[117] James Herman, Jonathan Francis, Siddha Ganju, Bingqing Chen, Anirudh Koul, Abhinav Gupta, Alexey Skabelkin, Ivan Zhukov, Max Kumskoy, and Eric Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9793–9802, 2021. 3.3, 6.3.9, 7.2, 7.2.5

[118] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf. 4.2

[119] Birger Hjørland and Hanne Albrechtsen. Toward a new horizon in information science: Domain-analysis. *Journal of the American society for information science*, 46(6):400–425, 1995. 2.1

[120] Jerry R Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. Commonsense metaphysics and lexical semantics. *Computational linguistics*, 13(3-4):241–250, 1987. 2.1.1, 4.1

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. 2.2.1

[122] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. *Robotics: Science and Systems*, 2021. 2.1.3

[123] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016. 2.1.3

[124] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Xiaodan Liang, Lianhui Qin, Haoye Dong, and Eric P Xing. Deep generative models with learnable knowledge constraints. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 10522–10533, 2018. 2.1.3, 2.1.3

[125] G. Huang. Densely connected convolutional networks. In *CVPR*, 2017. 2.2.1, 5.3.4

[126] Subin Huh and Insoon Yang. Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach. *arXiv preprint arXiv:2002.10126*, 2020. 7.2.6

[127] Filip Ilievski, Pedro Szekely, Jingwei Cheng, Fu Zhang, and Ehsan Qasemi. Consolidating commonsense knowledge. *arXiv preprint arXiv:2006.06114*, 2020. 4.3.2, H4

[128] Muhammad Zubair Irshad, Chih-Yao Ma, and Zsolt Kira. Hierarchical cross-modal agent for robotics vision-and-language navigation. *CoRR*, abs/2104.10674, 2021. URL `https://arxiv.org/abs/2104.10674`. 5.2.5

[129] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. TGIF-QA: toward spatio-temporal reasoning in visual question answering. *CoRR*, abs/1704.04497, 2017. URL `http://arxiv.org/abs/1704.04497`. 5.3.5

[130] Zhanhong Jiang, Jonathan Francis, Anit Kumar Sahu, Sirajum Munir, Charles Shelton, Anthony Rowe, and Mario Bergés. Data-driven thermal model inference with armax, in smart environments, based on normalized mutual information. In *2018 Annual American Control Conference (ACC)*, pages 4634–4639. IEEE, 2018. 3

[131] Bangti Jin and Peter Maass. Sparsity regularization for parameter identification problems. *Inverse Problems*, 28(12):123001, 2012. 2.1.3

[132] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. 2.1.2

[133] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019. 6.3.9, 7.2.5, 7.2.6

[134] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics Autom.*, 12(4):566–580, 1996. doi: 10.1109/70.508439. URL `https://doi.org/10.1109/70.508439`. 5.2.5

[135] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *CoRR*, abs/2101.01169, 2021. URL `https://arxiv.org/abs/2101.01169`. 2.2.1

[136] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404. IEEE, 2017. 6.2.7

[137] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2.1.2, 2.1.2, 2.1.3, 6.3.3, 7.2.1, 7.2.3

[138] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 5.2.3, 6.2.4, 7.2.1, 7.2.5

[139] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 2.1.2, 6.3.4

[140] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*, 2014. 2.1.2

[141] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016. 6.2.2

[142] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl. 4.2.7, 5.2.2, 5.2.2

[143] Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. A surprisingly robust trick for the Winograd schema challenge. In *Proc. of ACL*, pages 4837–4842, July 2019. H6, 4.3.6

[144] Sven Koenig and Maxim Likhachev. Real-time adaptive a*. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 281–288. ACM, 2006. doi: 10.1145/1160633.1160682. URL https://doi.org/10.1145/1160633.1160682. 5.2.5

[145] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 2.1.2

[146] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. URL http://arxiv.org/abs/1712.05474. 2.2.2, 5.2.5

[147] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015. 6.3.5, 7.2, 7.2.3, 7.2.5

[148] Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *The International Journal of Robotics Research*, 39(12):1419–1469, 2020. 2.1.3, 7.2.6

[149] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018. 6.2

[150] Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision and language navigation in continuous environments. In *European Conference on Computer*

Vision (ECCV), 2020. 5.2.5

[151] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications, 2004. 4.3.5

[152] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017. 4.3.2, 5.2.5

[153] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 7.2.6

[154] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683, 2017. URL `http://arxiv.org/abs/1704.04683`. 4.2

[155] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016. URL `http://arxiv.org/abs/1604.00289`. 5.3.4

[156] George Lakoff, Mark Johnson, and John F Sowa. Review of philosophy in the flesh: The embodied mind and its challenge to western thought. *Computational Linguistics*, 25(4):631–634, 1999. 1

[157] Steven M. Lavalle, James J. Kuffner, and Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000. 5.2.5

[158] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 6.2.6, 6.2, 6.4, 6.6, 6.2.7, 6.3.9

[159] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. 2.1.3

[160] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. Sensebert: Driving some sense into bert. *ArXiv*, abs/1908.05646, 2019. 4.2.7, 4.3.6

[161] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 292–302, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1028. URL `https://www.aclweb.org/anthology/P19-1028`. 4.2.7

[162] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229. Citeseer, 2004. 6.3.5, 7.2.3

[163] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A. Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1494–1499, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1159. URL `https://www.aclweb.org/anthology/D19-1159`. 2.2.1

[164] Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. Harvesting and refining question-answer

pairs for unsupervised QA. In *Proc. of ACL*, pages 6719–6728, July 2020. 4.3, H2, 4.3.6

[165] Junwei Liang, Lu Jiang, Liangliang Cao, Li-Jia Li, and Alexander G. Hauptmann. Focal visual-text attention for visual question answering. *CoRR*, abs/1806.01873, 2018. 5.3.5

[166] Paul Pu Liang, Yao Chong Lim, Yao-Hung Hubert Tsai, Ruslan Salakhutdinov, and Louis-Philippe Morency. Strong and simple baselines for multimodal utterance embeddings. 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)*, 2019. 6.2.7

[167] Huadong Liao and Jiawei He. Jacobian determinant of normalizing flows. *arXiv preprint arXiv:2102.06539*, 2021. 2.1.2, 6.3.4

[168] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *ArXiv*, abs/1909.02151, 2019. 4.2.7, 4.3.6

[169] Tsung-Yi Lin. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.*, 2017. 5.3.4

[170] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015. 7.2.6

[171] Kin Sum Liu, Sirajum Munir, Jonathan Francis, Charles Shelton, and Shan Lin. Long term occupancy estimation in a commercial space: an empirical study. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 307–308. IEEE, 2017. 3

[172] Kin Sum Liu, Elvin Vindel Pinto, Sirajum Munir, Jonathan Francis, Charles Shelton, Mario Berges, and Shan Lin. Cod: A dataset of commercial building occupancy traces. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 1–2, 2017. 3

[173] Kuan Liu, Yanen Li, Ning Xu, and Prem Natarajan. Learn to combine modalities in multimodal deep learning. *arXiv preprint arXiv:1805.11730*, 2018. 6.2.3

[174] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *AAAI*, 2020. 4.3.6

[175] 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. *arXiv preprint arXiv:1907.11692*, 2019. 4.2, 4.3.6

[176] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019. 1

[177] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *International Conference on Learning Representations*, abs/1511.03643, 2016. 2.1.4

[178] Sihui Luo, Xinchao Wang, Gongfan Fang, Yao Hu, Dapeng Tao, and Mingli Song. Knowledge amalgamation from heterogeneous networks by common feature learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. 2.1.4

[179] Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowl-

edge for commonsense question answering. *ArXiv*, abs/1909.05311, 2019. 4.2.7

[180] Yunlian Lv, Ning Xie, Yimin Shi, Zijiao Wang, and Heng Tao Shen. Improving target-driven visual navigation with attention on 3d spatial relationships, 2020. 5.2.5

[181] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1901.03035. (document), 2.2.2, 5.3.2, 5.3.3, 5.3.4, 5.3.4, 5.3.5, 5.6

[182] Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. Towards generalizable neuro-symbolic systems for commonsense question answering. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 22–32, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6003. URL https://www.aclweb.org/anthology/D19-6003. 4.3, H3, 4.3.6, 5.2.5

[183] Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. Knowledge-driven self-supervision for zero-shot commonsense question answering, 2021. 4.1, 5.2.5

[184] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019. 6.2, 6.2.6, 6.2.7

[185] Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*, 2018. 4.2.7

[186] John McCarthy et al. *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA, 1960. 2.1.1, 4.3

[187] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, volume 1, page 2, 2016. 5.3.5

[188] Todor Mihaylov and Anette Frank. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1076. URL https://www.aclweb.org/anthology/P18-1076. 4.2.7

[189] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013. 2.2.1

[190] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1147. URL https://www.aclweb.org/anthology/D16-1147. 4.2.7

[191] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. 4.3.6

[192] John Milnor and David W Weaver. *Topology from the differentiable viewpoint*. Princeton university press, 1997. 2.1.2, 6.3.4

[193] Marvin Minsky. *A framework for representing knowledge*. de Gruyter, 2019. 1, 2.1.3

[194] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016. 5.3.5

[195] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/D17-1106. 5.3.5

[196] Arindam Mitra, Pratyay Banerjee, Kuntal Kumar Pal, Swaroop Mishra, and Chitta Baral. Exploring ways to incorporate additional knowledge to improve natural language commonsense question answering. *arXiv preprint arXiv:1909.08855*, 2019. 4.3, H3, 4.3.6

[197] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783. 5.3.5

[198] Mahdi Kazemi Moghaddam, Qi Wu, Ehsan Abbasnejad, and Javen Qinfeng Shi. Optimistic agent: Accurate graph-based value estimation for more successful visual navigation, 2020. 5.2.5

[199] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://www.aclweb.org/anthology/N16-1098. (document), 4.10

[200] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746. Citeseer, 2006. 6.3, 6.3.9

[201] Sirajum Munir, Ripudaman Singh Arora, Craig Hesling, Juncheng Li, Jonathan Francis, Charles Shelton, Christopher Martin, Anthony Rowe, and Mario Berges. Real-time fine grained occupancy estimation using depth sensors on arm embedded platforms. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 295–306. IEEE, 2017. 3

[202] Sirajum Munir, Le Tran, Jonathan Francis, Charles Shelton, Ripudaman Singh Arora, Craig Hesling, Matias Quintana, Anand Krishnan Prakash, Anthony Rowe, and Mario Berges. Fork: fine grained occupancy estimator using kinect on arm embedded platforms. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 1–2, 2017. 3

[203] Sirajum Munir, Jonathan Francis, Matias Quintana, Nadine von Frankenberg, and Mario Bergés. Dataset: Inferring thermal comfort using body shape information utilizing depth sensors. In *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*, pages 13–15, 2019. 3

[204] M Lynne Murphy. *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms*. Cambridge University Press, 2003. 4.3.2

[205] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 809–816. Omnipress, 2011. 2.2.1

[206] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2.2.1

[207] Yilin Niu, Fangkai Jiao, Mantong Zhou, Ting Yao, Jingfang Xu, and Minlie Huang. A self-training method for machine reading comprehension with soft evidence extraction. *arXiv preprint arXiv:2005.05189*, 2020. 4.3.6

[208] Will Norcliffe-Brown, Efstathios Vafeias, and Sarah Parisot. Learning conditioned graph structures for interpretable visual question answering. *arXiv preprint arXiv:1806.07243*, 2018. 5.3.5

[209] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta, and Andreas Geiger. Learning situational driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11296–11305, 2020. 6.3, 6.3.9, 7.2.6

[210] A Oltramari. *Hybridism in cognitive science and technology, Foundational and implementational issues*. PhD thesis, PhD thesis, Univesita degli Studi di Trento, Dottorato in Scienze della . . . , 2006. 1

[211] Alessandro Oltramari, Jonathan Francis, Cory Henson, Kaixin Ma, and Ruwan Wickramarachchi. Neuro-symbolic architectures for context understanding, 2020. 2.2.1, 5.2.5

[212] Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. SemEval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1119. URL https://www.aclweb.org/anthology/S18-1119. 4.2

[213] Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. Improving question answering with external knowledge. *CoRR*, abs/1902.00993, 2019. URL http://arxiv.org/abs/1902.00993. 4.2.7

[214] Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. Improving question answering with external knowledge. *CoRR*, cs.CL/1902.00993v1, 2019. URL https://arxiv.org/abs/1902.00993v1. 4.2.6

[215] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017. 6.3, 6.3.9

[216] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017. 6.3.4

[217] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. 2.1.2, 6.3.4

[218] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018. 6.2.7

[219] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Minseok Kang, Jonathan Francis, Ashwin Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. In *European Conference on Computer Vision*, pages 282–298. Springer, 2020. (document), 2.1.2, 6.1, 6.2.3, 6.9, 6.2.6, 6.6, 6.7, 6.3, 6.3.4, 6.3.9, 7.2.3, 7.2.3

[220] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`. 5.2.3

[221] 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)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://aclanthology.org/D14-1162`. 2.2.1, 5.2.2

[222] Matthew E. Peters, Mark Neumann, IV RobertL.Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. *ArXiv*, abs/1909.04164, 2019. 4.2.7, 4.3.6

[223] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019. 4.3, 4.3.6

[224] Hai Pham, Paul Pu Liang, Thomas Manzini, Louis-Philippe Morency, and Barnabás Póczos. Found in translation: Learning robust joint representations by cyclic translations between modalities. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, 2019. 6.2.7

[225] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY . . . , 1989. 6.3, 6.3.9

[226] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021. 7.2.6

[227] Yiding Qiu, Anwesan Pal, and Henrik I. Christensen. Learning hierarchical relationships for object-goal navigation, 2020. 5.2.5

[228] Ahmed Hussain Qureshi, Yutaka Nakamura, Yuichiro Yoshikawa, and Hiroshi Ishiguro. Show, attend and interact: Perceivable human-robot social interaction through neural attention q-network. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1639–1645. IEEE, 2017. 6.2.7

[229] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 4.2, 4.2.5, 4.3.6

[230] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://www.aclweb.org/anthology/D16-1264`. 4.2, 4.3.6

[231] Qiu Ran, Peng Li, Weiwei Hu, and Jie Zhou. Option comparison network for multiple-choice read-

ing comprehension. *CoRR*, abs/1903.03033, 2019. URL http://arxiv.org/abs/1903.03033. 4.2.1, 4.2.1

[232] Abhilasha Ravichander, Thomas Manzini, Matthias Grabmair, Graham Neubig, Jonathan Francis, Alessandro Oltramari, and Eric Nyberg. How would you say it? eliciting lexically diverse dialogue for supervised semantic parsing. *18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, 2017. 3

[233] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7:1, 2019. 7.2.2

[234] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*, 04 2020. URL http://arxiv.org/abs/2004.09813. 4.3.3

[235] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015. 2.1.2, 2.1.2, 6.2.2, 6.3.4

[236] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 2.1.2

[237] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 6.1, 1, 2, 6.2.2, 6.2.2, 6.2.4, 6.2.4, 6.2.6, 6.2, 6.4, 6.6, 6.2.7, 6.3, 6.3.9

[238] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 6.3, 6.3.7, 6.9, 6.3.9

[239] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019. 6.1, 6.2, 6.2.2, 6.2.3, 6.2.6, 6.2.7, 6.3, 6.3.9

[240] Kyle Richardson and Ashish Sabharwal. What does my qa model know? devising controlled probes using expert knowledge. *arXiv preprint arXiv:1912.13337*, 2019. 4.3, 4.3.6

[241] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018. 6.3.3

[242] Cristian Rodriguez, Basura Fernando, and Hongdong Li. Action anticipation by predicting future dynamic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 6.2.7

[243] Alvaro Collet Romea, Bo Xiong, Corina Gurau, Martial Hebert, and Siddhartha Srinivasa. Exploiting domain knowledge for object discovery. In IEEE, editor, *Proceedings of (ICRA) International Conference on Robotics and Automation*, pages 2118 – 2125, May 2013. 2.1

[244] Lorenzo Rosasco, Alessandro Verri, Matteo Santoro, Sofia Mosci, and Silvia Villa. Iterative projection methods for structured sparsity regularization. *Computer Science and Artificial Intelligence Laboratory Technical Report*, 2009. 2.1.3

[245] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, pages 5115–5120. IEEE, 2017. 7.2.6

[246] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O

Arras. Human motion trajectory prediction: A survey. *arXiv preprint arXiv:1905.06113*, 2019. 6.2

[247] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 6.2.7

[248] Homagni Saha, Fateme Fotouhif, Qisai Liu, and Soumik Sarkar. A modular vision language navigation and manipulation framework for long horizon compositional tasks in indoor environment. *CoRR*, abs/2101.07891, 2021. URL https://arxiv.org/abs/2101.07891. 5.2.5

[249] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. 4.3.3, 4.3.3, 5

[250] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035, 2019. 2.1.1, 4.2.2, 4.2.6, 4.3

[251] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proc. of EMNLP-IJCNLP*, pages 4463–4473, November 2019. 4.3, 4

[252] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Conference on Robot Learning*, pages 237–252. PMLR, 2018. 6.3.9

[253] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347. 5.2.3, 7.2.2

[254] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603, 2016. 4.2.1

[255] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *Proceedings of International Conference in Robotics and Automation (ICRA)*, 2017. URL http://arxiv.org/abs/1704.06888. 5.3.5

[256] Irwin I Shapiro. The prediction of satellite orbits. In *Dynamics of Satellites/Dynamique des Satellites*, pages 257–312. Springer, 1963. 6.2.7

[257] Peng Shi and Jimmy Lin. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019. 5.3.2

[258] Andrew Shin, Masato Ishii, and Takuya Narihira. Perspectives and prospects on transformer architecture for cross-modal tasks with language and vision, 2021. 2.2.1

[259] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01075. URL https://doi.org/10.1109/CVPR42600.2020.01075. 2.2.2

[260] Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised

commonsense question answering with self-talk. *ArXiv*, abs/2004.05483, 2020. 4.3, 4.3.4, 4.10, 4.3.6

[261] David Silver, J Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29 (12):1565–1592, 2010. 6.3.9

[262] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2.2.1

[263] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1223–1237, Berlin, Heidelberg, 2002. Springer-Verlag. ISBN 3-540-00106-9. URL http://dl.acm.org/citation.cfm?id=646748.701499. 4.2.5

[264] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, 2016. URL http://arxiv.org/abs/1612.03975. 2.1.1, 4.2.2, 4.2.6, 4.3, 5.2.2, 5.2.5

[265] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep RL with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020. 2.1.3, 7.2.6

[266] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. URL http://arxiv.org/abs/1906.05797. 5.2.4, 5.2.5

[267] Kieran Strobel, Sibo Zhu, Raphael Chang, and Skanda Koppula. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1969–1975. IEEE, 2020. 7.2.6

[268] Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. Learning stochastic inverses. *Advances in neural information processing systems*, 26:3048–3056, 2013. 2.1.2

[269] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019. 6.2.7

[270] Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. Dream: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231, 2019. ISSN 2307-387X. URL https://www.transacl.org/ojs/index.php/tacl/article/view/1534. 4.2.6, 4.2.6, 4.4

[271] Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 4, 09 2014. 6.2.6

[272] Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 1984. AAI8410337. 5.3.4

[273] Denis Sverdlov. Roborace. https://roborace.com/, 2020. Last accessed: 2021-01-30. 7.2.3

[274] Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. 2.1.2, 6.3.4

[275] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://www.aclweb.org/anthology/N19-1421. 4.2.6, 4.3, 2, 4.4

[276] Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. Teaching pre-trained models to systematically reason over implicit knowledge. *arXiv preprint arXiv:2006.06609*, 2020. 4.3.6

[277] Alexandre Tamborrino, Nicola Pellicanò, Baptiste Pannier, Pascal Voitot, and Louise Naudin. Pre-training is (almost) all you need: An application to commonsense reasoning. In *Proc. of ACL*, pages 3878–3887, July 2020. 4.3.4

[278] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018. URL http://arxiv.org/abs/1804.10332. 5.3.4

[279] Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1006. URL https://www.aclweb.org/anthology/D18-1006. 4.2.7

[280] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15398–15408, 2019. 6.2, 6.2.6, 6.2.7

[281] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000. 5.2.2

[282] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3): 4915–4922, 2021. 7.2.1, 7.2.6

[283] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. 6.3.9

[284] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020. 2.1.4

[285] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. *arXiv preprint arXiv:2103.14258*, 2021. 2.1

[286] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16(1):2023–2049, 2015. 2.1.4

[287] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009. 2.1.4

[288] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,

Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 2.2.1, 5.3.2, 1, 6.2.2, 6.2.3, 6.2.7, 6.3.3

[289] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018. 6.2.7

[290] Loup Verlet. Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967. 6.1, 6.2.2, 6.3

[291] Varun Kumar Vijay, Abhinav Ganesh, Hanlin Tang, and Arjun Bansal. Generalization to novel objects using prior relational knowledge, 2019. 5.2.5

[292] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL `http://dblp.uni-trier.de/db/journals/corr/corr1411.html#VinyalsTBE14`. 5.3.5

[293] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. 4.3.2

[294] Kim P Wabersich and Melanie N Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *arXiv preprint arXiv:1812.05506*, 2018. 7.2.6

[295] David S Wallace, Sylvia Wandell Conner West, Anne Ware, and Donald F Dansereau. The effect of knowledge maps that incorporate gestalt principles on learning. *The Journal of Experimental Education*, 67(1):5–16, 1998. 2.1

[296] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743, 2017. 2.2.1

[297] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International conference on machine learning*, pages 1083–1092. PMLR, 2015. 1

[298] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1018. URL `https://www.aclweb.org/anthology/P17-1018`. 4.2.1

[299] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. *CoRR*, abs/1803.07729, 2018. URL `http://arxiv.org/abs/1803.07729`. 5.3.4, 5.3.5

[300] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. A unified view of deep metric learning via gradient analysis. *OpenReview*, 2018. 5.3.2

[301] Theophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. *CoRR*, abs/1707.06203, 2017. URL `http://arxiv.org/abs/1707.06203`. 5.3.5

[302] Trent Weiss and Madhur Behl. Deepracing: a framework for autonomous racing. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1163–1168. IEEE, 2020.

7.2.6

[303] Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*, 2017. 4.2.7

[304] Max Wertheimer. Gestalt theory. *Kegan Paul, Trench, Trubner & Company*, 1938. 2.1

[305] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=H1gX8C4YPr`. 5.2.3, 5.2.4, 5.2.5

[306] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019. 4.3.5

[307] Michael J Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995. 1

[308] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *CoRR*, abs/1801.02209, 2018. URL `http://arxiv.org/abs/1801.02209`. 5.3.5

[309] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL `http://dx.doi.org/10.1109/TNNLS.2020.2978386`. 5.2.5

[310] Jiangnan Xia, Chen Wu, and Ming Yan. Incorporating relation knowledge into commonsense reading comprehension with multi-task learning. In *Proc. of CIKM*, CIKM '19, page 2393–2396, 2019. ISBN 9781450369763. 4.3.6

[311] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *ArXiv*, abs/1611.01604, 2016. 4.2.1

[312] Wenhan Xiong, Xiaoxiao Guo, Mo Yu, Shiyu Chang, Bowen Zhou, and William Yang Wang. Scheduled policy optimization for natural language communication with intelligent agents. *arXiv preprint arXiv:1806.06187*, 2018. 5.3.5

[313] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. 1, 6.2.2

[314] Ming Xu, Matias Quiroz, Robert Kohn, and Scott A Sisson. Variance reduction properties of the reparameterization trick. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2711–2720. PMLR, 2019. 2.1.3

[315] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020. 7.2, 7.2.6

[316] Wei Yang, X. Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *ArXiv*, abs/1810.06543, 2019. 5.2.5

[317] Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, J. Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. G-daug: Generative data augmentation for commonsense reasoning. *ArXiv*, abs/2004.11546, 2020. 4.3, 4.3.6

[318] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019. URL http://arxiv.org/abs/1906.08237. cite arxiv:1906.08237Comment: Pretrained models and code are available at https://github.com/zihangdai/xlnet. 4.2, 4.2.5

[319] Zhi-Xiu Ye, Qian Chen, Wen Wang, and Zhen-Hua Ling. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *arXiv preprint arXiv:1908.06725*, 2019. 4.3, 4.3.6

[320] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*, 2018. 2.2.1, 4.2.4

[321] Yukun Yuan, Kin Sum Liu, Sirajum Munir, Jonathan Francis, Charles Shelton, and Shan Lin. Leveraging fine-grained occupancy estimation patterns for effective hvac control. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 92–103. IEEE, 2020. 3

[322] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 7.2.6

[323] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1009. URL https://www.aclweb.org/anthology/D18-1009. 4.2

[324] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6720–6731, 2019. 4.2

[325] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1472. 4.2, 4.3.3, 4.3.5

[326] Jimuyang Zhang and Eshed Ohn-Bar. Learning by watching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12711–12721, 2021. 7.2.6

[327] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885, 2018. URL http://arxiv.org/abs/1810.12885. 4.2

[328] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15222–15232, 2021. 7.2.6

[329] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *Proc. of ACL*, pages 1441–1451, July 2019. 4.3.6

[330] Long Zhao, Xi Peng, Yuxiao Chen, Mubbasir Kapadia, and Dimitris N. Metaxas. Knowledge as priors: Cross-modal knowledge generalization for datasets without superior knowledge, 2020. 2.1.4

[331] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019. 6.2, 6.2.6, 6.1, 6.2, 6.4, 6.6, 6.2.7

[332] Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Improving question answering by commonsense-based pre-training. *CoRR*, abs/1809.03568, 2018. URL `http://arxiv.org/abs/1809.03568`. 4.2.7

[333] Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Improving question answering by commonsense-based pre-training. In Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing*, pages 16–28, Cham, 2019. Springer International Publishing. 4.3.6

[334] Xuhui Zhou, Y. Zhang, Leyang Cui, and Dandan Huang. Evaluating commonsense in pre-trained language models. In *AAAI*, 2020. 4.3.4, H1