Towards Knowledge-capable AI: Agents that See, Speak, Act and Know

Kenneth Marino

July 2021
CMU-ML-21-108

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Abhinav Gupta, Chair
Tom Mitchell
Ruslan Salakhutdinov
Rob Fergus, New York University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2021 Kenneth Marino. All rights reserved.

This research was supported by: the Intelligence Advanced Research Projects Activity contract number 2016-16032900006; the National Science Foundation award numbers DGE1252522 and IIS1320083; and the Office of Naval Research award numbers N000141010934, N000141612007 and N000141812312.

Keywords: knowledge, artificial intelligence, embodied AI, machine learning, computer vision, reinforcement learning, robotics

To Victoria.

Abstract

The field of artificial intelligence has been interested in knowledge since its early days, using carefully crafted rules and curated knowledge from humans to build effective expert systems. Since then, many fields, such as computer vision and natural language processing, have been dominated by large-scale end-to-end learning using large datasets. This has often left knowledge as an afterthought for many important problems. However, as our performance on marquee challenges and datasets such as the ImageNet Challenge [294] saturates and the field becomes more concerned with problems such as large-category recognition and problems of full embodied AI (agents that require understanding of multiple modalities), knowledge will become even more important. In this thesis, we argue that to achieve the goal of clever robots, or embodied AI, we need to deal with all three modalities of vision, language and action. We further argue that knowledge is the critical piece to in connect these modalities.

In our contributions, we show different slices of these cross-modalities to come to an understanding of how knowledge can be used to join these modalities. First, we look at how to incorporate knowledge into neural network architectures in vision problems. Next, we examine how we can combine the modalities of vision and language. We introduce a benchmark for vision and language that requires models with the capability to bring in and reason about knowledge about the world. Then we develop a method on that dataset which combines two types of knowledge: knowledge graphs and implicit knowledge from large language models. We then examine the action modality by first showing that by using the knowledge inherent in language models to solve a highly complex, semantic crafting task. Then, we apply knowledge in the robotics setting of task-oriented grasping and see how we can use knowledge to allow agents to perform tasks on never before seen object categories and new tasks. Finally, we start to move in the opposite direction and look at how knowledge can be created. We show how an action policy can be used by agents to build up their own knowledge of the world.

Acknowledgments

I first want to thank my Advisor Abhinav Gupta. Abhinav has been a dogged advocate during my career, providing advice, support and guidance throught my PhD.

Next I would like to thank my research collaborators and co-authors. While a thesis is usually thought of the culmination of the intellectual work of one person, research is a collaborative effort. In addition to my advisor Abhinav Gupta, I would like to thank my other co-authors: Ruslan Salakhudinov, Rob Fergus, Arthur Szlam, Devi Parikh, Marcus Rohrbach, Martial Hebert, Ali Farhadi, Mohammad Rastegari, Roozbeh Mottaghi, Sonia Chernova, Adithyavairavan Murali, Weiyu Liu, Jacob Walker, Xinlei Chen and Valerie Chen. And although the project we worked on never saw the light of day, I would like to thank Carl Doersch who was one of my early mentors when I first started my PhD.

I would like to thank those people who at various points in my PhD provided feedback on papers, my thesis, presentations, specific help with software or installation, etc. I will surely miss a lot of people, but I would like to thank Achal Dave, Senthil Pls, Nadine Chang, Sudeep Dasari, Shubham Tulsiani, Brian Okorn, Shikhar Bahl, Jason Zhang, Abhinav Shrivastava, Lerrel Pinto, Adithyavairavan Murali, Victoria Dean, Helen Jiang, Yufei Ye, Aishwarya Agrawal, Gunnar Sigurdsson, Victoria Donley, Xialong Wang, Eric Kolve, Dustin Schwenk, Amanpreet Singh, Abhishek Das, Ronghang Hu, Vedanuj Goswami, Allie Del Giorno, Ben Newman, Nick Rhinehart, Olga Russakovsky, Ishan Misra, as well as lab-mates at CMU and everyone else who at various points provided assistance.

I would like to thank all of the many many friends I made during my PhD and without whom I surely would not have made it through these last six years. To include names here would inevitable leave people out, so I will leave this as a generic thank you.

Finally, I would like to thank my family, especially my parents Kathleen and Charles Marino, my brother Dean and my fiancée Victoria Donley

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It has been said that which way you ought to go depends on where you want to get to [56]. So where do we want to go, and how do we get there?

For the first question, we would propose that what we want is an embodied agent that is able to perform everyday cognitive and physical tasks as a human might. We might for instance want a robot that is capable of acting as a home assistant. It would need to be able to navigate a house and understand what "living room" referred to and know the proper place for items. It would need to be able to use tools for a wide variety of tasks such chopping carrots for a stew or using a broom or vacuum to clean. We would want to be able to converse with the robot to give it instructions and to explain how we would like a particular task done, such as specifying that we like DVDs sorted by genre rather than alphabetically. We would also want it capable of understanding and performing under-specified tasks such as finding things around the house to pack for a vacation.

In other words, we want agents that are capable first of understanding the world around them, understanding how different objects and people and locations correspond semantically, and completing complex actions requiring fine motor control. We also need them to understand human language and be able to pick up on and understand not only the names of objects, but abstract concepts. From this, we propose viewing embodied intelligence as a three-legged stool (see Figure 1.1). The stool is held up by three legs. The first leg is vision. Agents need perception to make observations about their immediate environment, to navigate and to recognize various

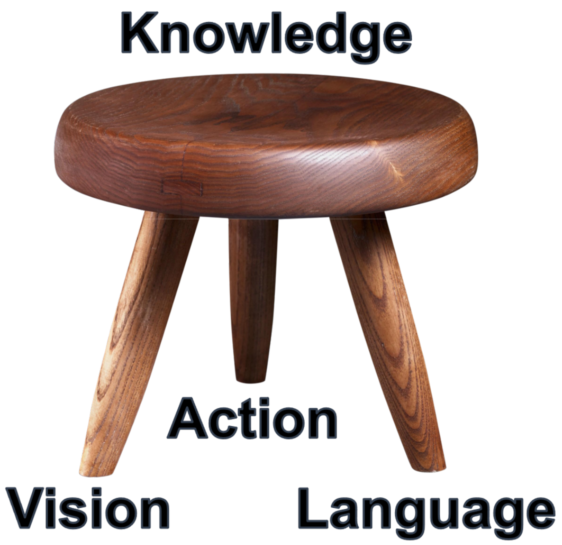


Figure 1.1: The three legged stool of embodied intelligence.

objects. The next leg is action. We want our agents to actually take action in their environments including locomotion. The third leg is language. If we want agents that can receive complex and nuanced instructions from humans or share information and be cooperative, they must understand language beyond the surface level. Finally, the stool is held together by the base: knowledge. Knowledge, in our view, is key to an agent's ability to act intelligently in the world. It is key for recognition, and for taking action and understanding language. And it is key for connecting these modalities together.

In this thesis, we will use this idea of embodied intelligence as a framework and show how our work on knowledge systems in these modalities contributes to the goal of human-like embodied intelligence.

1.1 What is Knowledge?

One preliminary question we might need to answer first, however, is: What is knowledge? As we'll see in our sections summarizing knowledge in philosophy in Chapter 2.1, this is not even a settled question outside of AI. While we probably will not be able to arrive at a final answer to this question, we can try to get at this by describing what knowledge does, what things we would informally consider to be knowledge and what we would not.

1.1.1 What Knowledge Does

Informally, we might describe knowledge as a capability. And what it does is give a human or an agent a prior or model of the world. The world is incredibly complicated. There are so many things in the world that even humans cannot have a precise model for everything we might experience in the world. So humans use knowledge, gathered from ourselves and from others to fill in missing information to act as a simplified, structured way to think about the world. Knowledge is both a prior we have to do a task (e.g. roses are red, cats have ears) and a thing we produce (e.g. summarize a historical event into an article, make a diagram describing the water cycle).

1.1.2 Types of Knowledge

Perhaps more useful to this question would be to be list the things that we or others have considered to be “knowledge” and think about what they have in common.

Knowledge graphs / knowledge bases

A knowledge graph is a particular form of knowledge structured as triplets: (head, relation, tail)[1]. The head and tail are concepts, often nouns and sometimes single words (e.g. “dog” “cat” “PhD thesis”). Relations are words or phrases, often including verbs, that describe the relationship between the head and tail concepts (e.g. is, has, part of). Because of this particular structure, we can represent this knowledge as a graph where concepts are the nodes of the graph and relations are the edges.

Textual knowledge

Text is a much broader category of knowledge. Some common forms of this might be encyclopedia articles such as from Wikipedia [1], definitions from a dictionary, or

[1]Sometimes also referred to as a knowledge base, but a knowledge graph is a particular form of a knowledge base with a graph structure

even free-form sentences about a particular object or being. It may even come as exact logical forms from which we can symbolically derive other beliefs.

Situational knowledge

Another type of knowledge (which may overlap with textual or other forms of knowledge depending on the representation) is situational or environmental knowledge. This is distinguishable from general or global knowledge in that it may relate to specific instances or be in some way embodied to a particular agent. For instance, knowing that the door near a particular exit in a particular building at a particular time gets stuck easily.

Implicit knowledge from large language models

Recently, with the advent of extremely large pre-trained language models such as BERT [90, 203] and GPT [50, 278, 279], we might also want to think about the knowledge contained within these models. These models have been trained on millions or even billions of sentences. They have essentially read thousands of books and encyclopedia articles, and often implicitly contain knowledge about many subjects. We know in fact that they contain a large amount of knowledge because many methods have in fact studied how we can extract explicit knowledge from these models [37, 39, 153, 210, 269].

1.1.3 What Knowledge is

Given these types of knowledge, we would list the following properties as important:

1. **Knowledge is semantic.** Either directly with symbols, or with words in a language which are recognized by humans as relating to discrete categories, or sub-symbolically from which symbolic meanings can be extracted [37, 39, 355]
2. **Knowledge is a priori** to the final learning task (although the knowledge itself may be learned from data)
3. **Knowledge is *not* the same as data**
4. **Knowledge relates to and contains information about the world** (either our world or its simulated environment)

While this is not meant to be an exhaustive or complete set of properties or a definition, it is at least enough for us to distinguish knowledge and knowledge-based methods in AI from other methods and allow us to discuss work in knowledge-capable AI from this point forward.

1.2 Why Use Knowledge

Now that we have an idea of what knowledge is, we can ask what benefits we gain from using knowledge. Especially in a time where it seems that methods which are able to simply scale computation and data always outpefrom carefully tuned systems or try to encorporate human knowledge [330], we want to outline some reasons why we still may want to use knowledge in our systems. Despite the centrality of data-driven, supervised learning approaches, there are still a number of applications where systems can greatly benefit from knowledge.

1. **Incorporating knowledge priors we have about the world.** This is essentially an argument of scale: if we define our set of tasks and capabilities wide enough, we cannot cannot simply learn everything, we need knowledge to give us priors which we do not need to learn. For narrow tasks, we would not see this benefit as the benefit of priors converges to zero as we increase the amount of data. But if our goal is agents that can do anything, this scaling of data for all possible tasks is not possible.
2. **Recognizing rare or novel categories by using prior knowledge we have about those categories.** This is similar to the earlier argument, but specific to categories. We know that in vision, and in other areas, concepts or objects naturally create a "long-tail" distribution [400]. So no matter how much additional data you add, you are always going to have categories with very little data. This is where knowledge can help.
3. **Planning and making decisions by using or learning specific knowledge about an environment.** This is most related to this earlier idea we had about situational knowledge. To be able to make decisions in the world, we need to know all of the relevant things in our environment.

4. **Communicating and doing recognition in settings where the learning agent must have very specific kinds of knowledge.** We know that humans have conversation based on shared knowledge about things (e.g. you can't have a conversation about a football game without shared knowledge about what football even is, its rules, knowledge about how that particular game went etc.). This insight has been shared and used in several works in conversational agents [93, 398, 398].

On a high level, we believe that we still need knowledge for the same reason that humans do. The world is so vast that it is impossible to learn everything directly. Some narrow tasks we may be able to collect larger and larger datasets, but as we focus on broader AI tasks and as focus moves away from tasks where we can exploit fixed datasets and towards tasks requiring direct experience (e.g. in robotics), knowledge will become more important. It seems inevitable that as we try to move towards better mimicry of human capabilities, that will include humans' ability to use and acquire knowledge.

1.3 Contributions

We now know that our goal is embodied intelligence. We have a working idea of what "knowledge" is. And we have a rationale as to why knowledge systems are important in the age of data-driven AI. We now discuss our contributions to the field of embodied knowledge-capable AI, each of which we will discuss in detail in later chapters[2].

1.3.1 Knowledge in Vision

The first contribution we will discuss is how we use knowledge specifically for the first leg of our stool: visual perception.

One characteristic that sets humans apart from modern learning-based computer vision algorithms is the ability to acquire knowledge about the world and use that

[2]Because of course the works in these chapters were written in collaboration with my co-authors and much of those publications are used in whole or part in this thesis, any works that want to cite this thesis should credit the corresponding papers

knowledge to reason about the visual world. Humans can learn about the characteristics of objects and the relationships that occur between them to learn a large variety of visual concepts, often with few examples. In this contribution, we investigate the use of structured prior knowledge in the form of knowledge graphs and shows that using this knowledge improves performance on image classification. We build on the work on end-to-end learning on graphs, introducing the Graph Search Neural Network as a way of efficiently incorporating large knowledge graphs into a vision classification pipeline. We show in a number of experiments that our method outperforms standard neural network baselines for multi-label classification.

We published this work in the conference paper: "The More You Know: Using Knowledge Graphs for Image Classification" *CVPR* 2017 [222]. We describe this work in Chapter 3.

1.3.2 Knowledge in Language and Vision: Benchmarks

In the second contribution we move towards joining two modalities: vision and language. In this contribution we look specifically at moving towards better tests of visual and language understanding that require reasoning about the world outside the image, requiring outside knowledge.

Visual question answering (VQA) in its ideal form lets us study reasoning in the joint space of vision and language and serves as a proxy for the AI task of scene understanding. However, most VQA benchmarks are focused on questions such as simple counting, visual attributes, and object detection that do not require reasoning or knowledge beyond what is in the image. In this contribution we address the task of knowledge-based visual question answering and provide a benchmark, called OK-VQA, where the image content is not sufficient to answer the questions, encouraging methods that rely on external knowledge resources. The dataset includes more than 14,000 questions that require external knowledge to answer. We show that the performance of the standard VQA models degrades drastically in this new setting. Our analysis shows that our knowledge-based VQA task is diverse, difficult, and large compared to previous knowledge-based VQA datasets.

We published this work in the conference paper: "OK-VQA: A visual question answering benchmark requiring external knowledge" *CVPR* 2019 [224]. We describe this work in Chapter 4.

1.3.3 Knowledge in Language and Vision: Methods

In the next contribution, now that we have a dataset to investigate knowledge-capable vision and language systems, we propose and analyze a method that combines two of our notions of knowledge: knowledge graphs and implicit knowledge from language models.

One of the most challenging question types in VQA is when answering the question requires outside knowledge not present in the image such as those in OK-VQA. To solve this, we tap into two types of knowledge representations and reasoning. First, implicit knowledge which can be learned effectively from unsupervised language pretraining and supervised training data with transformer-based models. Second, explicit, symbolic knowledge encoded in knowledge bases. Our approach combines both—exploiting the powerful implicit reasoning of transformer models for answer prediction, and integrating symbolic representations from a knowledge graph, while never losing their explicit semantics to an implicit embedding. We combine diverse sources of knowledge to cover the wide variety of knowledge needed to solve knowledge-based questions. We show our approach, *KRISP (Knowledge Reasoning with Implicit and Symbolic rePresentations)*, significantly outperforms state-of-the-art on OK-VQA. We show with extensive ablations that while our model successfully exploits implicit knowledge reasoning, the symbolic answer module which explicitly connects the knowledge graph to the answer vocabulary is critical to the performance of our method and generalizes to rare answers.

We published this work in the conference paper: "KRISP: Integrating Implicit and Symbolic Knowledge for Open-Domain Knowledge-Based VQA" *CVPR* 2021 [226]. We describe this work in Chapter 5.

1.3.4 Knowledge in Action: RL

In our next contribution, we now begin to incorporate action into our systems. We take our notion of language as knowledge and look at how we can use the implicit knowledge that comes from language to solve a complex multi-task crafting environment by training our agent to generate language.

Complex, multi-task problems have proven to be difficult to solve efficiently in a sparse-reward reinforcement learning setting. In order to be sample efficient, multi-task learning requires reuse and sharing of low-level policies. To facilitate the automatic decomposition of hierarchical tasks, we propose the use of step-by-step human demonstrations in the form of natural language instructions and action trajectories. We introduce a dataset of such demonstrations in a crafting-based grid world. Our model consists of a high-level language generator and low-level policy, conditioned on language. We find that human demonstrations help solve the most complex tasks. We also find that incorporating natural language allows the model gives prior knowledge to our agent and allows it to generalize to unseen tasks in a zero-shot setting or to learn a new task quickly from a few demonstrations.

We published this work in the conference paper: "Ask Your Humans: Using Human Instructions to Improve Generalization in Reinforcement Learning" *ICLR* 2021 [64]. We describe this work in Chapter 6.

1.3.5 Knowledge in Action: Robotics

In the next contribution, we continue exploring how we use knowledge for action, this time joining the action modality with vision in a robotics setting. In this work, we look at how we can use knowledge to allow agents to perform tasks on never-before-seen object categories and tasks.

Despite the enormous progress and generalization in robotic grasping in recent years, existing methods have yet to scale and generalize task-oriented grasping to the same extent. This is due to the lack of scale of the number of objects and tasks studied and then by the lack of knowledge and prior semantics is previous work. We address these concerns first with the TaskGrasp dataset which is more diverse both in terms of objects and tasks, and an order of magnitude larger than previous datasets. The dataset contains $250K$ task-oriented grasps for 56 tasks and 191 objects along with their RGB-D information. With a more semantically rich dataset, we have a new breadth and diversity of categories that lets us develop the GCNGrasp framework which uses the semantic knowledge of objects and tasks encoded in a knowledge graph to generalize to new object instances, classes and even new tasks. Our framework shows a significant improvement of around 12% on held-out settings compared to baseline methods which do not use semantics. We demonstrate that our dataset and model are applicable for the real world by executing task-oriented grasps on a real robot on unknown objects.

We published this work in the conference paper: “Same Object, Different Grasps: Data and Semantic Knowledge for Task-Oriented Grasping” *CoRL* 2021 [244]. We describe this work in Chapter 7.

1.3.6 Learning Knowledge from Actions

And, moving in the other direction, our final contribution looks at how we can learn what we called situation knowledge in an embodied setting. We would argue that there is a two step process to learning knowledge. The first step is to create some candidate knowledge or hypothesis. Given what knowledge we already have, and some model of what we need to know next, we generate a hypothesis that we want to test. Next, given a hypothesis, we want to verify it. So for instance, we might verify the hypothesis that the door is sticky by moving the door and observing its behavior. Once we have verified a hypothesis, we can then add it to our pool of knowledge. In this last contribution, we focus in on this second part. Given some hypothesis about the world, how do we verify it, thus turning it into knowledge?

This contribution formulates “hypothesis verification” as a reinforcement learning problem. Specifically, we aim to build an agent that, given a hypothesis about the dynamics of the world can take actions to generate observations which can help predict whether the hypothesis is true or false. Existing RL algorithms fail to solve this task, even for simple environments. In order to train the agents, we exploit the underlying structure of many hypotheses, factorizing them as {*pre-condition*, *action sequence*, *post-condition*} triplets. By leveraging this structure we show that RL agents are able to succeed at the task. Furthermore, subsequent fine-tuning of the policies allows the agent to correctly verify hypotheses not amenable to the above factorization.

We published this work on ArXiv: “Empirically Verifying Hypotheses Using Reinforcement Learning” [225]. We describe this work in Chapter 8.

Chapter 2

Background

This section will provide an overview of how knowledge has been viewed and used in the various sub-fields of AI and by others. We will also provide background details and explanations of the common datasets, knowledge sources and methods used in the field of knowledge-based AI. Finally, we give a broad overview of the use of knowledge in the fields of computer vision (CV), natural language processing (NLP), robotics, and reinforcement learning (RL).

With this chapter, we hope to do a couple things. First, to give credit to the many works throughout the years that have dealt with the question of knowledge. Second, to provide background to readers who are less familiar with knowledge-aware AI systems. Finally, we seek to put our own contributions into historical perspective. We hope that this chapter will serve not only as a useful background for reading this thesis, but also serve the community as a useful source for literature review, historical perspective and overview of common methods in this sub-field.

In Section 2.1 we begin with a very brief overview of knowledge in philosophy to try to get a grounding on the origins of the ideas we have about what knowledge is. Then in Section 2.2 we look at the origins of knowledge in the earlier years of artificial intelligence, especially looking at the symbolic approaches (sometimes nicknamed Good Old-Fashioned AI or GOFAI).

Next, in Section 2.3 we give an overview of some common methods used in knowledge systems, including word2vec, large language models, and graph neural networks used in our contributions. We then go through in Section 2.4 some common knowledge sources including knowledge graphs used by knowledge systems, including in our own work.

We then move on to problems and application areas. First in Section 2.5, we look at learning knowledge. This is relevant to our contribution in Chapter 8 on how we learn knowledge in action. We then go through the applications of knowledge in various AI subfields. First we discuss knowledge applications in natural language processing (Section 2.6) where it is perhaps most common. Next, we discuss the use of knowledge in computer vision, including in the cross-field of Language+Vision (Section 2.7). We discuss how our contributions in Chapters 3,4 &5 fit into these lines of work. Finally, we discuss knowledge in the context of the modality of action, encompassing both robotics and general RL and simulated agents (Section 2.8). Here we put our contributions in Chapters 6&7 into context of this work.

2.1 Knowledge in Philosophy

The subject of knowledge: what it is, how it's acquired and how it's used is a major topic in the field of Philosophy. Such a major topic in fact that not only is the field too big to be contained in a Philosophy PhD thesis, it is too big even for entire books to fully capture. As the author of this thesis is not a philosopher, we will not attempt anything except a very brief overview of epistemology, the philosophical study of knowledge. It will mostly be a summary of the relevant entries of the Stanford Encyclopedia of Philosophy [392], the Encyclopedia Britannia and other sources which we will note.

Most summaries of the history of epistemology start with Plato. The word epistemology in fact comes from the Greek words "episteme" and "logos" meaning knowledge and knowledge/understanding and argument/reason. Plato's epistemology looked at what it means to "know" and how acquiring knowledge is a moral virtue in and of itself [392]. Plato's understanding of knowledge contrasted sensory experience from knowledge, saying in Theaetetus that "sense experience cannot be a source of knowledge, because the objects apprehended through it are subject to change." [47]. This idea of a separation of experience and knowledge is still influential, including in this thesis. We often make a distinction between "experience" or "data," things which are directly experienced by an agent as not being knowledge (although knowledge can be informed by experience). To Plato, knowledge was transcendent to experience, requiring the application to reason to arrive at an unchanging understanding of the objects in the world. This *Platonic* idea of categories as fixed unchanging things is quite relevant to the AI community as this is the often unstated assumptions we have about, for instance, object categories such as cat or dog.

As part of the empiricist movement, Locke's epistemology was concerned with the role of experience in the formation of ideas or knowledge [47]. Locke defined knowledge as "the perception of the connexion of and agreement, or disagreement and repugnancy of any of our ideas" [204]. Locke considered knowledge to be of three types: intuitive knowledge which are driven purely the mind such as "white is not black", demonstrative knowledge, such as a conclusion driven by a premise to its conclusion by the laws of logic, and sensitive knowledge which are ideas that are caused externally rather than by the mind [47]. An important idea in Locke's epistemology is ideas of certainty. He considered intuitive knowledge to be the most certain, demonstrative knowledge to be less certain than that as it required logical steps to the conclusion and sensitive knowledge to be the least. He had an additional category which he called "probable opinion" which he did not consider knowledge at all as it could not be held with certainty [47]. Most of what we would consider knowledge, including in this thesis falls into this last category.

In his famous work Critique of Pure Reason [161] Kant argues that "space and time are merely formal features of how we perceive objects, not things in themselves that exist independently of us, or properties or relations among them" [392]. In other words, we know nothing of what Kant called *things in themselves*. Kant believed that "knowledge must rest on judgements that are a priori, for it is only as they are separate from the contingencies of experience that they could be necessary and yet also synthetic" [47]. This idea is also very relevant to our understanding of AI. For instance, Kant said that concepts are read into experience, not out of it, meaning that these are a priori concepts rather than empirical ones. In other words, knowledge is prescriptive.

Russell's epistemology gives us another useful distinction between direct and indirect knowledge of truths, and later a distinction between knowledge by Acquaintance and Knowledge by Description [392]. "To be justified, every indirect knowledge claim must be capable of being derived from more fundamental, direct or intuitive knowledge claims. The kinds of truths that are capable of being known directly include truths about immediate facts of sensation and truths of logic." [392].

More recently, philosophers have tried to understand "how our degrees of confidence are rationally constrained by our evidence" and feminist philosophers have explored how our material interests can affect our evidence and rational constraints [392]. Other philosophers such as the postmodernists (characterized by their skepticism of modernism) reject the idea of being able having objective knowledge. To the postmodernist, what is considered to be knowledge is influenced by or even wholly determined by the society which they are formed in [47]. This idea is useful to keep in mind as our datasets and knowledge sources are human-made and thus encode certain cultural or political assumptions about our world and if we are not careful this can have unintended consequences.

2.2 Knowledge in Good Old-Fashioned AI

Knowledge, what it is, how it's represented and how it is used is one of the foundations problems in the field of artificial intelligence. In the inaugural presidential address to the American Association for Artificial Intelligence (AAAI), Allen Newell (one of the founders of the field as well as the School of Computer Science at Carnegie Mellon) said this about the question of knowledge and representation: "[This] is a little like a physicist wishing to address the question of radiation and matter." [251]

Knowledge [40] is a central theme of traditional AI. Commonsense reasoning [82, 83, 200] approaches, e.g. CYC [190], codify everyday knowledge into a schema that permits inference and question answering. However, the underlying operations are logic-based and occur purely within the structured representation, having no mechanism for interaction with an external world. Expert systems [122] instead focus on narrow domains of knowledge, but are similarly self-contained. Logic-based planning methods [73, 109] generate abstract plans that could be regarded as action sequences for an agent.

Some of the earliest work in artificial intelligence was concerned with the manipulation of knowledge. For example, in "Baseball: An Automatic Question Answerer" [128], Bert Green and his collaborators devised a system called Baseball. Baseball was a program designed to answer question asked by humans posed in "ordinary English" related to the data it had stored. The program was written in IPL-V [252], an information processing language using lists and list hierarchies to store information. The program would take in a question and parse the question to generate specification list, or a simplified canonical expression for what the question was asking. Using this, the program would then use a series of logical rules to pull relevant entries and find the answer which answers the spec list. This early period in AI and knowledge was characterized by these so-called “AI-languages” and the direct manipulation of symbols like entries in a database.

In his AAAI address in 1982, The Knowledge Level [251], Newell recounts what he calls the great theorem-proving controversy. Alan Robinson developed the first-order logic program Resolution [289] which kicked off a flood of papers exploring resolution-based theorem-proving. In his telling, “[w]ithin about five years, however, it became clear that this basic engine was not going to be powerful enough to prove theorems that are hard on a human scale, or to move beyond logic to mathematics, or to serve other sorts of problem solving, such as robot planning.” [251]. Newell further summarizes his opinion of the state of knowledge representation in AI by recounting the SIGART “Special issue of knowledge representation” [41]. The issue gathered together survey results from dozens of researchers working on knowledge representation. Newell concluded from the diverse results that “[t]here was no consensus on any question of substance” and highlighting one response: “Standard practice in the representation of knowledge is the scandal of AI.” Despite these pessimistic notes, however, Newell was still optimistic about the role of knowledge in AI, believing it to be essential. He proposed in that address the “Knowledge Level” [251] which is a level of abstraction for agents above the program or symbol level “which is characterized by knowledge as the medium and the principle of rationality as the law of behavior.” In other words, the knowledge level was not merely a claim that agents *have* knowledge, but a claim that there exists this higher level of computation or reasoning that reasons over *knowledge*. The knowledge itself (the body of knowledge) is distinct here from the agent’s knowledge level.

While none of these ideas or approaches about knowledge or symbolic AI really disappeared (especially in robotics where planning is still widely used), we believe that it is fair to say that this became a less overwhelming focus of the community. In the decades since Newell’s address, much of the focus of research in the community moved towards statistical techniques. The field of machine learning has become so dominant now in the way that we think about AI that in common usage the two terms became interchangeable[1].

The work in this thesis is largely concerned with how knowledge systems can be incorporated as a part of machine learning systems rather than as an attempt to directly continue the work in symbolic AI. However, there is still much that can be learned and appreciated from the early work in knowledge AI. For instance, the work on building up a large body of knowledge such as with CYC [190] has carried through to the present era. More recent knowledge graphs such as ConceptNet [200, 321] owe their origins to this line of work. And the contributions of this thesis certainly owe a lot to this line of work in knowledge graphs, although our interest in them has mostly been as the consumer of this knowledge (Chapters 3,5,7). The problem formulation of question answering has its origins from at least [128]. From its translation from Question Answering in the NLP community to visual question answering (VQA) in the vision community, it lost this idea of retrieving from a base of knowledge. But in our work in Chapter 4 (and in other contemporary works) owes greatly to the original understanding of the problem. Indeed, even Newell's ideas about knowledge influence the ways that we now think about what knowledge in AI even is.

2.3 Common Methods in Knowledge Systems

In this section, we discuss some of the most common methods and forms that knowledge-aware systems take, with a particular emphasis on the methods that we employ with out contributions. In particular, we discuss word embeddings, large language models and graph neural networks. In the next section (Section 2.4) we discuss the sources of knowledge themselves (although with word vectors for instance, this distinction can be blurry).

2.3.1 Word Embeddings

Maybe one of the earliest forms that prior knowledge takes is word embeddings. As we discussed in Chapter 1.1.2, text and pre-trained models trained on text is itself a

[1]This is a descriptive statement rather than a normative one.

form of knowledge. In early systems in computer vision and robotics in particular, word embedding were a common way to summarize or get prior knowledge about a particular word or category.

At the most basic level, a word embedding is simply a mapping of a single word[2]. We could write this as $f : WORD \rightarrow emb \in \mathcal{R}^d$ where d is the fixed dimensionality of our embedding. While the idea representing words as continuous vectors is very old [22, 101, 293], where this gets truly interesting from a knowledge perspective is when we train these embedding on datasets of billions of words. The best known early example of this was Mikolov et al. [232], often referred to as Word2Vec[3]. Other popular embeddings include GloVE [266], ConceptNet NumberBatch Speer et al. [321] and FastText [30] (which includes N-grams).

The basic training procedure to collect word embeddings is pretty simple. You start with a large corpus of text such as Wikipedia [1], Common Crawl, or even Twitter. You then create a neural network architecture and train on that corpus by trying to learn to predict the context of a particular word. In the Continuous Bag-of-Words (CBOW) approach, the model learns a projection for all words (this will later become the word2vec embedding), projects the four words before and the four words after (the context words) and tries to predict the target word between. The second architecture, the Continuous Skip-gram Model does the opposite, predicting the context words given the input word. See Figure 2.1 taken from Mikolov et al. [232]. In either model, you end up with a learned projection for each word in the vocabulary that we can now use as the vector representation of that word.

Why do people train these models? The simple answer that relates to this thesis is that it provides a very convenient representation for words that also gives *knowledge* about that word or category. Because the CBOW or Skip-gram model is trained to try to learn the context of a particular word, this representation tends to group similar concepts together. For instance, synonyms or very similar words such as dog and canine are grouped together *because these words often appear in a very similar context*. Another result that we have found with word embeddings is that they tend to actually encode useful knowledge. For example, in [234] and [233] they find that word vectors

[2]Only one word per vector is not strictly necessary, see Speer et al. [321]. And this idea of word vectors can be extended to multiple words using n-grams [45]

[3]In fact, this term is often used interchangeably with the overall concept of *any* word embeddings

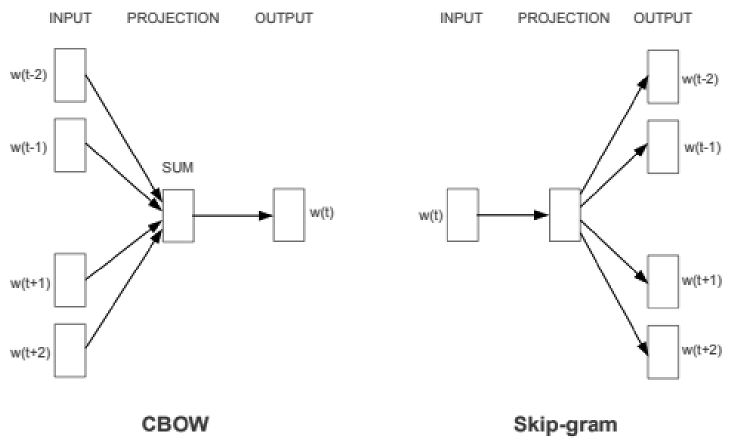


Figure 2.1: Figure showing the architecture and training of the Continuous Bag-of-Words (CBOW) and Continuous Skip-gram Models. Figure taken from Figure 1 of Mikolov et al. [232].

tend to implicitly encode relationships between words. For instance, if you look at countries and their capital city, you find that the vector difference for different pairs tends to be very similar. So the vector when you subtract “Greece” from “Athens” is similar to the vector “Oslo – Norway” and “Harare – Zimbabwe.” And in practice, word embeddings have been useful in many different application areas as one form of knowledge. Word embeddings are quite convenient because the user of the embedding does not need to train the large models themselves and because the simple nature of vectors makes them incredibly versatile. We discuss many applications of word embeddings in Section 2.7 and Section 2.8. We also make use of word vectors as a form of knowledge in several of our contributions: in Chapter 5 and Chapter 6 we use them as the initialization for our graph node embeddings and in Chapter 7 we use them as our representation of the task, environment and instruction.

2.3.2 Large Language Models

The next logical step is to extend the embedded knowledge from word embeddings to entire sentences or even paragraphs. This is where large pretrained language models come in. One of the early extensions of the idea of word embeddings to sentences was Skip-Thought Vectors [172]. Similar to the Skip-gram model for word embeddings, this model would train on a large text corpus. The model takes in a fixed-length sentence as input and tries to predict the previous and next sentences as context. This method has two major drawbacks. The first is that it only works for sentences of a fixed size. The second is that because sentences are sequential, one word after another, training this model at scale on large corpora is difficult.

The transformer architecture [347] solved both of these problems, and as a result is the basis for the modern pretrained language models trained on large corpora. As with previous works, these transformer-based models are trained on an unsupervised objective on a large corpus. For BERT (Bidirectional Encoder Representations from Transformers) [90, 203], the proxy task is again predicting the context of words in a sentence, this time by making use of a special mask token, asking the model to predict the masked out words of a sentence. GPT (Generative Pre-Training) [50, 278, 279] on the other hand uses sentence generation as the proxy task. Given the first part of a sentence or paragraph, it predict the most likely completion of that text.

As with word embedding models, many works have looked at these models as learning and storing vast amounts of knowledge learned from the Web. For instance, many works have found that knowledge graph triplets can be explicitly extracted using these models [37, 39, 153, 210, 269], implying that they essentially contain knowledge bases [153].

We include works in the proceeding sections that use the knowledge from these models. Section 2.5 discusses works which automatically learn knowledge bases from these models. Section 2.6&2.7 show the use of pretrained language models for downstream tasks in NLP and computer vision. In Chapter 5, we use knowledge from one of these pretrained language models (BERT [90]) and combine this with knowledge graphs to achieve better performance on our knowledge-based visual question answering dataset OK-VQA (Chapter 4).

2.3.3 Graph Neural Networks

Another common method in knowledge-based machine learning systems is to use a knowledge graph or graphs (see Section 2.4) as a source of knowledge and incorporate in an end-to-end manner using graph neural networks). In this section, we will give an overview of graph neural networks to better understand the methods in our thesis.

Much of the work on graph neural networks starts with Graph Neural Networks [300]. The idea was fairly straightforward, consisting of a propagation model to compute node representations on the graph and an output model to make predictions on nodes. For each node v we have a state $x_v \in \mathcal{R}^d$. And we let the graph neural network update itself based on the value of its neighboring nodes and the edge types between them.

$$x_v = f_w(l_v, l_{co[v]}, x_{ne[v]}, l_{ne[v]}) \tag{2.1}$$

$$o_n = g_w(x_v, l_v) \tag{2.2}$$

where f_w is a parametric function called the local transition function that expresses the dependence between node v on its neighbors and g_w is the local output function. l_n, $l_{co[v]}$, $x_{ne[v]}$ and $l_{ne[v]}$ are the label of v, the labels of its edges, the states and labels of the nodes in the neighborhood of v respectively. Because these equations are recurrent (the value of x_v depends on the value of neighboring states $x_{v'}$ which in turn rely on x_v) these networks were restricted so that the propagation function is a contraction map around a unique fixed point and trained with the Almeida-Pineda algorithm [6, 272].

One of the later extensions of Graph Neural Networks was Graph Gated Neural Networks (GGNN) [196]. Like GNN, GGNN is an end-to-end neural network model operating on graph structured data and which can output a per-node classification. As with Graph Neural Networks the idea of GGNN is that given a graph with N nodes, we want to produce some output which can either be an output for every graph node $o_1, o_2, ...o_N$ or a global output o_G. Similarly to GNN, the hidden state of each node x_v is updated by the value of its neighbors based on the type and direction of edge joining them. Unlike GNNs however, this model unrolls the recurrence to a fixed number of steps and does propagation through time to update the value of the nodes, similar to how recurrent network models such as LSTMs [141] are trained. Like with GNNs, we have a hidden state x_v, but it is also indexed by time: $x_v^{(t)}$ at every time step t. We start at $t = 0$ with initial hidden states i_v that depends on the problem. For instance, for learning graph reachability, this might be a two bit vector that indicates whether a node is the source or destination node.

Next, we use the structure of our graph, encoded in a matrix A which serves to retrieve the hidden states of adjacent nodes based on the edge types between them. The hidden states are then updated by a gated update module similar to an LSTM. The basic recurrence for this **propagation network** is

$$x_v^{(1)} = [i_v, 0]^T \tag{2.3}$$

$$a_v^{(t)} = A_v^T [x_1^{(t-1)} ... x_N^{(t-1)}]^T + b \tag{2.4}$$

$$z_v^t = \sigma(W^z a_v^{(t)} + U^z x_v^{(t-1)}) \tag{2.5}$$

$$r_v^t = \sigma(W^r a_v^{(t)} + U^r x_v^{(t-1)}) \tag{2.6}$$

$$\widetilde{x_v^t} = tanh(W a_v^{(t)} + U(r_v^t \odot x_v^{(t-1)})) \tag{2.7}$$

$$x_v^{(t)} = (1 - z_v^t) \odot x_v^{(t-1)} + z_v^t \odot \widetilde{x_v^t} \tag{2.8}$$

A_v is the adjacency matrix of the graph for node v, and the W's and U's are learned parameters. Eq 2.3 is the initialization of the hidden state with i_v and 0 in the empty dimensions. Eq 2.4 shows the propagation updates from adjacent nodes. Eq 2.5-2.8 combine the information from adjacent nodes and current hidden state of the nodes to compute the next hidden state.

After T time steps, we have our final hidden states. The node level outputs can then just be computed as:

$$o_v = g(x_v^{(T)}, i_v) \tag{2.9}$$

where g is a fully connected network: the **output network**.

Another line of work reframes graphs as a special case of a convolutional input. Pixels on a grid can be thought of as a graph structure where each node is connected to its neighboring pixels, relying on either some global graph structure or doing some sort of pre-processing on graph edges [51, 100, 138, 254]. The most widely-used of these works is the Graph Convolutional Network (GCN) model from Kipf and Welling [170]. The first input of a GCN is an undirected graph $\mathcal{G} = (V, E)$. The graph is represented as a binary adjacency matrix A, which is normalized to obtain $\hat{A}$ following [170] [4]. As with earlier graph networks, each node v has an embedding $x_v \in \mathcal{R}^{d_{input}}$[5] The input

[4]This follows from a first order approximation of localized spectral filters on graphs, see [170]

[5]For GCN, the dimension can be different for the input d_{input} at each convolutional layer, and at the output.

to each node can be rewritten as a single input matrix $\mathcal{X} \in \mathcal{R}^{|V| \times d_{input}}$. The output of the GCN is another d_{output}-dimensional embeddings for each node $\mathcal{Z} \in \mathcal{R}^{|V| \times d_{output}}$. The node embeddings are propagated to their neighbours using message passing in each convolutional layer:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \tag{2.10}$$

where σ is the ReLU activation function, $W^{(l)}$ is the learned weight parameter for each layer l, $H^{(0)} = \mathcal{X}$ and $H^{(L)} = \mathcal{Z}$ where L is the number of layers.

Since these initial work, many variants of graph neural networks have been proposed and studied. First, there are many varieties of graph convolutional networks. The Relational Graph Convolutional Network (RGCN) [301]. This model (unlike GCN which takes undirected graphs as input) supports having different calculations between nodes for different edge types and edge directions. Other variants include Edge-Conditioned Convolution on Graphs [313], Crystal Graph Convolutional Neural Networks [369], Edge-Conditioned Convolution [313], Topology Adaptive Graph Convolutional Networks [97], Signed Graph Convolutional Network [88] and many others [19, 63, 85]

Another line of work in graph networks has looked at how powerful the representations are for different varieties of graph network are [375]. Many works such as the Graph Attention Network [348] have looked at adding attention mechanisms into the propagation equations of graph networks [165, 340]. Others have investigated various kinds of graph pooling [25, 52, 91, 92, 187, 214, 284, 313, 388] and neighborhood aggregation [74, 107]. Memory-based Graph networks (MemGNN) [163] integrate an external memory with graph networks. Other works have looked at Graph Encoders [259, 297] including Variational Graph Auto-Encoders [169]. Others works have introduced yet more varieties of graph neural networks such as SAGEConv [134], Residual Gated Graph ConvNets [46] and many others [26, 48, 214, 240, 291, 363, 374][6].

There are a number of works looking at graph embeddings [129, 136, 349, 380], and still more works relating deep neural networks with graph structures [7, 154, 267, 302, 341]. Another related neural network structure are Deep Sets [391] and

[6]Thanks to Fey and Lenssen [108] for compiling and implementing an extensive list of Graph Neural Networks, which we used to find many of these citations.

Set2Set [351] which, like graph networks, do not have a set ordering of elements. Unlike graph neural networks, the inputs are un-ordered elements, which can be seen as a special case of a graph where all of the nodes are un-connected. This makes the networks operating on graphs in many ways similar to networks operating on graphs. The Transformer architecture [347], while primarily used for sequential data, also naturally deals with sets by simply removing the positional encoding input of the input encoder.

There is also a parallel line of work that uses graph kernels for SVM classifiers rather than neural networks. The first of these works that referred themselves as "graph kernels" were Gärtner et al. [121] and Kashima et al. [162][7] Since then many varieties of graph kernels have been proposed including shortest-path kernels [35], graphlet kernels [307], Weisfeiler-Lehman graph kernels [308], deep graph kernels [271] and graph invariant kernels [258]. Vishwanathan et al. [352] gives an overview of many of these kinds of kernels and creates a framework for comparing them to each other. Some works, such as [98] have even looked at fusing graph neural networks with graph kernels.

For a more in-depth review of graph networks, graph kernels, deep sets and other related geometric deep learning topics, see Bronstein et al. [49].

We make use of graph neural networks in several of our contributions. In Chapter 3 we build off of GGNNs [196]; in Chapter 7 we use GCN [170]; and in Chapter 5 we use the related RGCN [301].

2.4 Types and Sources of Knowledge

In this section, we talk about the sources of knowledge that are commonly used in the community, including in our contributions. This includes knowledge graphs (Section 2.4.1) and textual sources (Section 2.4.2) either directly or as a source of training large language models (see Section 2.3.2).[8] We will mention knowledge

[7]Somewhat similar approaches such as diffusion kernels [176] predate this, but were not specifically referred to as graph kernels.

[8]One thing you may notice is that nearly all of the knowledge sources listed here are in English. The overwhelming number of works in NLP are done in the English language, and this bias is reflected here in these datasets. Surely we miss something by only looking at English language sources. See Ruder [292] for further discussion.

sources such as NELL [55] which are collected automatrically from the web or by other means, but will discuss that like of work in Section 2.5.

2.4.1 Knowledge Bases/Graphs

A knowledge graph is a particular representation of knowledge commonly used in knowledge AI systems. It is structured as a graph $\mathcal{G} = (V, E)$ where each node v corresponds to a concept represented as a word or series of words (e.g. “dog”, ‘cat”, “Labrador retriever”) and each edge corresponds to a relation between those concepts (e.g. “is a”, “has”, “is part of”). An equivalent way to describe a knowledge graph is a knowledge base containing a list of tuples of the form $(head\_concept, relation, tail\_concept)$.

CYC

Perhaps the earliest large-scale knowledge bases was CYC [190]. CYC aimed to be a universal schema to span all all concepts understood by humans. It contains 10^5 concepts and 10^6 handcrafted axioms. Concepts (called CycL terms or constants) include individuals such as #$BillGates or #$England, collections such as #$Tree-ThePlant (all trees), and functions which take input terms and output a new one, and truth functions which return true or false. So this database would contain simple relations such as Bill Clinton is the President of the U.S. (#$isa #$BillClinton #$United-StatesPresident) and even more complicated expressions such as (#$relationAllExists #$biologicalMother #$ChordataPhylum #$FemaleAnimal) which translates to “for every instance of the phylum Chordata, there is a female animal which is its mother.”[9] CYC falls pretty clearly into the Good Old-Fashioned AI (See Section 2.2) way of thinking about knowledge. The goal was to encode every concept and axiom known to humans to allow an expert system to make logical inferences about every concept in existence. The classical example would be that given that Socrates is a man, and all men are mortal, the system would be able to conclude that Socrates is mortal.

[9]These examples come from the Wikipedia [1] article on CYC

WordNet

Shortly after, WordNet [235] sought to capture a database of all English nouns, verbs, adjectives and adverts. Unlike CYC, the goal was not to try to compile all human knowledge, but to create a complete ontology of English words. Words are grouped into synonyms (synsets) to represent each distinct concept. These synsets are joined by semantic relations. Most notable of these was the hyperonymy, instance or “is a” relationship between nouns, which gave users of WordNet a graph of concepts linked by their instance relationship to each other. We might for instance have pug, which is a dog, which is a canine. This resource was incredibly influential, both as a source of knowledge (e.g. in Zhu et al. [401] and in our own contributions in Chapters 3&7) and as a way to have a list of discrete concepts, taking away synonyms[10]. This second idea in particular was critical in ImageNet [86] by allowing the dataset authors to collect a set of $1,000$ image categories without significant overlap between categories.

ConceptNet

Building itself out from the WordNet ontology, ConceptNet [200], sought, like CYC [190], to collect commonsense knowledge about the world. The project was crowd-sourced and contains over 100,000 facts. Unlike CYC, however, it is organized as a knowledge graph, collected by translating English-language facts gathered from crowd workers into a triplet structure. Typical examples from the dataset include things such as (dog, has_property, friendly) or (cat, is_capable_of, hunt_mice). We make use of ConceptNet in our contribution in Chapter 5.

DBPedia

One of the largest available knowledge graph sources is DBPedia [18], which contains millions of knowledge triplets. DBPedia is a crowd-sourced project which structures content created from the various Wikimedia projects, including Wikipedia (see Section 2.4.2). Being sourced from Wikipedia articles and info boxes, DBPedia contains knowledge about virtually every film, book, song and notable human in history. Unlike previous knowledge graphs such as ConceptNet or Freebase, its knowledge is

[10]Although WordNet, like every ontology, does not do this perfectly

heavily biased towards proper nouns rather than everyday or commonsense knowledge. We use part of DBPedia for our contribution in Chapter 5.

Visual Genome

Visual Genome [179] is a dataset for studying visual relationships. Drawn from Flickr images, it contains annotations of over 33k object categories, 68k attribute categories and 42k relationship categories which were all later canonicalized to WordNet synsets. For each image in the dataset, there is a corresponding ground truth "scene graph" which contains labels for objects in these scenes, relational edges between those objects, and attributes for each object. In many works, a knowledge graph is built from these scene graphs by taking the most frequently occurring object-object and object-attribute relationships in the dataset. The Visual Genome graphs are useful for our research problems because they contain scene-level relationships between objects, e.g. person wears pants or fire hydrant is red and thus allow the graph network to reason about what is typically in a scene. This knowledge source tends to give spatial relationships e.g. (boat, is_on, water) and common pairwise affordances e.g. (person, sits_on, coach) as well as attributes. This knowledge-graph version of Visual Genome was possibly first used by our work in Chapter 3, and we make use of it again in Chapter 5.

hasPartKB

Another knowledge graph specifically collected for part relationships is hasPartKB [24]. It contains just one relation: "has part" for common objects such as (dog, has_part, whiskers) as well as scientific ones (molecules, has_part, atoms). We use hasPartKB as one of our knowledge sources in Chapter 5.

Other Knowledge Bases

Freebase [31] was another collaborative knowledge base attempting to harvest all kinds of structured data. It contains more than 125 million tuples. Freebase was eventually shut down and merged into Wikidata. Similarly, ATOMIC [298] looks at inferential knowledge including if-then relations (for example "if X pays Y a compliment, then

Y will likely return the compliment”). Later [147] introduced the bigger ATOMIC 2020 which focuses on knowledge not contained in pretrained language models.

Some knowledge bases were collected specifically with robotics in mind, such as KnowRobo [339] and Robobrain [299], ORO [189], OUR-K [197], and RoboCSE [79]. These knowledge bases aimed to provide robots with extensive knowledge about objects, spaces, tasks, actions, and agents they were likely to encounter. Other knowledge bases were collected specifically for visual reasoning tasks [65, 94, 295, 401, 402, 404].

Several works have looked at automatically gathering knowledge graphs from the web such as NELL [55] and the related NEIL [65], while others have looked at extracting them from large language models [37, 39, 153, 210, 269]. We give a more thorough survey of this line of work in Section 2.5.

2.4.2 Textual Knowledge Sources

Textual sources and especially knowledge-like text is very important in NLP. Many of our contributions use these textual sources in some way. Some such as the baselines in Chapter 4 use them directly, while other use them through pre-trained word embeddings (Chapters 5,6&7) or pre-trained language models (Chapters 4&5).

Wikipedia

Perhaps the most widely used textual knowledge source in AI is Wikipedia [1]. Like an encyclopedia, Wikipedia is divided into articles for each topic which can include historical figure, celebrities, novels, movies and music albums. The English-language Wikipedia alone contains over 6 million articles and 2.5 billion words. It was created to be a free online resource created and maintained by volunteers and contributors. According to Wikipedia, it is the most-read reference work in history. Because it contains information-dense text on so many topics, it is a very common use of both textual knowledge and text for training word embeddings large-language models such as GloVE [266], BERT [90] and GPT-3 [50]. Wikipedia is the source text for a number of NLP question answering datasets such as SQuAD [281] and VQA datasets such as WIT [323]. It has also been used as the source text for retrieval systems trying to apply knowledge on another task, including by us in our contribution in

Chapter 4.

Common Crawl

Another common source for large-scale text data is Common Crawl. As the name suggests, Common Crawl is data that has been continually scraped (using a web crawler) from the web since 2008. It contains peta-bytes of information including raw webpages, media and text. Unlike Wikipedia which is specifically curated as a useful source of knowledge, Common Crawl is meant to be just all of the data from the Web. As such it is much less structured and often contains duplicate or non-useful text. Despite this, due to the sheer amount of text it contains, it is a very popular dataset for training large language models, including GloVE [266] and GPT-3 [50].

BookCorpus

The BookCorpus dataset[11] was compiled from 11,038 free, unpublished books found by the authors on the internet. It contains over 74 million sentences and almost 1 billion words [12] and is split into 16 genres such as Science Fiction, Romance and Fantasy. The corpus was used to train GPT [278] and BERT [90], but now appears to be no longer accessible. Subsequently, GPT-3 [50] uses datasets which it calls Books1 and Books2[13].

Other crawls and datasets

Many other large-scale raw text datasets have been described and used in papers. One common type of text dataset is news. Common Crawl collected a crawl dataset specifically for news. Gigaword 5 [262] was an archive of newswire text gathered by the Linguistic Data Consortium (LDC). APNEWS was a dataset of AP news articles which was often used in the NLP literature including in Lau et al. [185]. And the Google News dataset, which was used to train Word2vec [234] was a proprietary

[11]Later works that use this including GPT Radford et al. [278] and BERT [90] refer to this as the Book**s**Corpus, which is presumably a typo.

[12]However, [90] describes it as 800 million words while in the original paper [405] it is said to have 984 million words.

[13]Which from the name would seem to be similar, but little information is given about them other than that they contain 12 and 55 billion tokens respectively

Google dataset containing one billion words. Twitter, the famous[14] social media platform which shows tweets publicly has been used as another source of raw text, including in GloVE [266]. ELMo [268] uses the 1B Word Benchmark [60]. GPT-2 [279] created a dataset of 8 million documents it calls WebText by crawling webpages linked on the social media site Reddit. Unlike common crawl, it only crawls webpages whose links on the social media cite are highly "upvoted" indicating that users found these pages useful or interesting.

2.5 Automated Knowledge Learning

While much of the focus in the literature (and indeed this thesis) is on knowledge as a source or input to models, it can also be an output. In our final contribution (Chapter 8) we tackle this problem in the context of learning knowledge about our immediate environment with our action policy, but this idea of learning knowledge automatically has many precedents.

2.5.1 Learning Knowledge from Text on the Web Text

One influential work in this area was the Never-Ending Language Learner (NELL) [55]. The essential idea was to build an intelligent computer agent that does two things. First, it extracts text from reading the web to populate a structured knowledge graph. Second, it learns to perform this first task better than it could the day before. NELL learns two types of knowledge: knowledge about which noun phrases map to specified semantic categories (e.g. cities, companies), and knowledge about pairs of noun phrases which specify semantic relations. It learns these types of knowledge in a number of ways. It learns to extract these from free-form text patterns (e.g. it learns that "Boston is a city" can be structured as the entity "Boston" having an ISA relationship with "city"), from sem-structured data such as tables and lists, and it learns probabilistic horn clause rules that let it infer relations based on other relations it knows (e.g. $\langle$ dog, is a, mammal$\rangle$ and $\langle$ mammal, is a, animal$\rangle \implies \langle$ dog, is a, animal$\rangle$). NELL has been continuously running since 2010 and has to date

[14]infamous?

logged almost 3 million high confidence beliefs with over 2,000 nodes/concepts and over 4,000 edges/relationships.

Other works have similarly learned knowledge from text data such as books [4] or web data, either by semi-supervised learning [54], by analyzing the frequency of a relational predicate in a text corpus [96, 383].

2.5.2 Learning Knowledge from Images

One of NELL's primary influences was the similarly named Never-Ending Image Learner (NEIL) [65]. Like NELL, NEIL ran unsupervised continuously for several years to extract, in this case, *visual* knowledge from the web. NEIL scrapes images from the internet and learns new object categories and relations between the object categories based on new images and its previously learned detectors. For instance, it might discover a new category "Corolla" and based on its similarity to its already trained "car" detector learn that (Corolla is a kind of car). In addition to taxonomy relationships, NEIL can learn part of relationships (eye is a part of baby), similarity relationships (swan looks similar to goose), object attribute relationships (pizza has round shape), scene-object relationships (bus found in bus depot) and scene attribute relationships (ocean is blue). It has learned an ontology of over 8000 visual concepts along with over 20,000 commonsense relationships.

In a similar vein, other systems learn knowledge about visual concepts. VisKE [295] learns to verify a relation predicate by jointly reasoning over text and images while other works learn to automatically discover new visual categories by using paired visual and textual data sources [328] or by web search of text [94]. Other works have used fully annotated object detection datasets combined with text to extract commonsense facts [384]. Other work has been done in object discovery [66], or learning structure or parts [255, 256], and other works have also explored this general idea of collecting knowledge for visual question answering [402, 404].

Unlike the approaches described here and in Section 2.5.1, our method in Chapter 8 does not use web data (either text or images) to predict edges in a knowledge base, but rather has an agent that needs to act in the world and observe the results of those actions. Like these works on never-ending learning [55, 65, 238] however, there is a similarity in that knowledge creation is seen as a two step process of hypothesizing facts and then verifying them. In our case, however, these hypotheses are generated by the environment rather than gathered from the web.

2.5.3 Learning Knowledge from Large Language Models

Most recently, the extreme scale of training data used for large language models (see Section 2.3.2&2.4.2) has inspired works to extract explicit knowledge from these models.

There were several concurrent works that made the observation that these models implicitly contained a lot of this relational knowledge. In [269], the authors find that BERT contains relational knowledge and is competitive with oracle knowledge on several NLP tasks, and shows that BERT can perform well on open-domain question answering. It develops a benchmark called LAMA to test completion of knowledge based sentence (e.g. Obama is a __ by profession). [153] builds on this approach by automatically discovering prompts that better induce the correct answer. Similarly [372] shows that pretrained models such as BERT can be used for zero-shot fact completion. Thus, this line of work aims to find ways of better extracting knowledge these large language models implicitly contain. In COMET [37], this observation was used combined with the idea that these models could be train to explicitly generate knowledge graphs. Given a seed of training knowledge, for instance on ConceptNet [200], the base model would finetune a pretrained language model to generate sentences of the form head_concept, relation, tail_concept. This was found to be able to generate novel, high-quality knowledge. [39] similarly generates new knowledge graph relations by first going to the raw text to find sentences that often express relations, extract the relation according to a template rule and then using the pretrained language model to predict whether that pair is likely correct. This recalls an early approach from [77], but this approach makes use of the explicit knowledge from the language model. Wang et al. [353] takes a similar approach as [39] using a text corpus in addition to a large language model to learn knowledge graphs. Wang et al. [355] shows that language models can be fine-tuned for multiple-relations extraction which is similar to knowledge base completion in that the task asks to find the semantic relation between two input entities given a source text. And although it does not generate knowledge graphs, [280] showed that language models could be used to generate commonsense explanations and boosted state-of-the-art by 10% on a commonsense question answering task CommonsenseQA [333].

As with the previous sections, this line of work is related to our contribution in Chapter 8 in that they are data-driven approaches to extracting knowledge, but extracted from a source of data rather than by interacting with an environment.

2.5.4 Learning Knowledge with RL and Robotics

Most similar to our ideas in Chapter 8 are lines of work that have looked at agents which take action which are used to infer or create knowledge.

One line of work including [209, 395] have attempted to combine RL with Knowledge representation and reasoning, for tasks such as navigation and dialogue. These take the world dynamics learned by RL and make them usable in declarative form within the knowledge base, which is then used to improve the underlying RL policy. Unlike in these works, however, these do not take the form of formal statements about the world and the goal is to improve the downstream policy rather than to extract knowledge per-se.

Another line of work relating to formalizations of (statistical) causality [265] has developed to provide a framework for an agent to draw conclusions about its world, and verify hypothesis as in Chapter 8. This is the approach taken in [81], where RL is used to train an agent that operates directly on a causal Bayesian network (CBN) in order to predict the results of interventions on the values on its nodes. In contrast, our approach was to avoid this formalization, with the hope of training agents who test hypotheses without building explicit CBNs. Unlike [81], our agents intervene on the actual world (where interventions make take many actions), rather than the abstract CBN.

The most similar work to ours is [87], which uses reinforcement learning (or imitation learning) directly on the world, and the agent gets reward for answering questions that require experimentation. However, the form of these questions (and thus the form of the knowledge they can acquire) in each world is the same, involving asking to describe an attribute or count of objects in a partially observable environment. In addition, the space of actions is such that random experimental policies could still find correct answers. In contrast, in our contribution this space of possible questions for a world is combinatorial and both random experimentation and vanilla RL are insufficient.

In the area of robotics, work has been demonstrated for automating the scientific process. In [127], robotic exploration of chemical reactivity was demonstrated using ML techniques. [167] developed a robot scientist that explored genomics hypotheses about yeast and experimentally tested them using laboratory automation. In biochemistry [345] used Bayesian methods for optimal experiment design. More generally, the Automated Statistician project [324] uses a Bayesian approach to reason about different hypotheses for explaining the data, with the aim of creating interpretable knowledge. These works can all be seen as automating the acquisition of knowledge by an agent, similar to our contribution.

2.6 Applications of Knowledge in NLP

The area of knowledge [15] is perhaps best studied in the field of natural language processing from knowledge base completion [32] to knowledge-based question answering [281] to open-domain question answering [61]. Because we are primarily interested in applying or learning knowledge in embodied systems, our contributions generally fall outside of the field of NLP, or are at least exist in the various subfields of language + vision/RL/robotics/etc. However, the line of text or language-only research has been enormously influential to our thinking in the area and sometimes directly on our methods. We will therefore give a broad overview of knowledge in NLP and where necessary relate it back to our contributions.

The generation of knowledge from textual sources is of course incredibly relevant to our work in Chapter 8. We discuss this line of work separately in its own section (Section 2.5), including works in multi-modal knowledge learning.

Another particular work of interest to knowledge-based systems is knowledge base completion. Some of the earliest work in this field comes from [104] and [32]. The typical setup for this task is that, given an existing knowledge graph, infer new facts

[15]The line of work on using and acquiring knowledge when it relates to what humans would consider “common sense” is often referred to commonsense or commonsense reasoning. We often use the terms interchangeably, but in the literature, the distinction will often be made to contrast this line of work with those which retrieve facts (e.g. “what is the capitol of Qatar”).

from an incomplete knowledge base. This task has been looked at many times using a variety of methods including using entity and relation embedding [376], tensor factorization or tensor networks [20, 317] and a variety of other methods [216, 288, 361]. This line of work again relates to our contribution in Chapter 8 except that here the agent needs to act in a world and observe the results of those actions rather than using an incomplete knowledge base or text corpus.

Another relevant area of NLP research is question answering, especially question answering from knowledge sources [23, 34, 382] including in open-domain question answering [61, 329, 358, 378, 379]. In the open-domain question answering setting, perhaps best exemplified by [61], the system is asked to answer a general knowledge question such as "Who won the 2018 super bowl" and the system has to reference an external knowledge source such as Wikipedia and answer the question. In the closed setting such as SQuAD [281, 282] the passage containing the answer was given, but in the open-domain setting, the exact passage will not be given. In the most general case, there might not even be any guarantee that there is a closed source text containing the answer. This is of special interest to commonsense knowledge as in CommonsenseQA [333].

This line of work on knowledge-requiring question answering datasets is quite similar to our work in knowledge-requiring *visual* question answering in Chapter 4. The major difference between these two lines of work is that in visual question answering, the question comes paired with an image as visual context. So the problem becomes not just one of question answering from knowledge sources, but of perception and reasoning about knowledge in the context of a visual scene. The methods developed for knowledge-based question answering are relevant to our work in Chapter 5, especially methods which also use knowledge graphs [320, 371, 387], other external knowledge [260] and especially works with mixed symbolic and implicit methods for question answering [164, 212].

Other knowledge intensive NLP tasks aside from question answering have also been introduced. One interesting line of work looks at using knowledge to augment dialog systems to produce more knowledgeable, engaging dialog agents [93, 398]. There has also been work in knowledge graph alignment [276], logical reasoning and theorem proving on knowledge graphs [236, 277], and generating structured graph explanations for commonsense reasoning [296]. Most recently KILT [270] introduced a benchmark for knowledge intensive language tasks including not only open domain QA but also dialog, fact checking, slot filling and entity linkin.

Another very relevant line of work to embodied knowledge is so-called physical commonsense. Recent works [27, 112] have investigated whether pre-trained language can learn what they called physical commonsense including the properties of objects (boats need fuel), affordances (boats can be driven) and inferences (if it can be driven it requires fuel). The conclusion of [112] is that language models can in fact learn some of this commonsense, but in general they only can learn things that have been explicitly written down. This validates our motivation in Chapter 8: certain kinds of knowledge about the world have to be learned directly through interaction.

2.7 Applications of Knowledge in CV

In this section, we discuss the related work in the field of computer vision (including the intersection of language and vision) using knowledge-based methods, from early work in attribute and ontology graphs to more recent work in knowledge-aware VQA systems. In this section, we will also give context and comparison to our work in knowledge and vision in Chapters 3,4&5.

2.7.1 Recognition

The use of knowledge and knowledge graphs for visual reasoning in general has been quite well studied. Possibly the earliest work we could call "knowledge" was the work relating to attribute approaches [106] to vision such as [183] which uses a fixed set of binary attributes to do zero-shot prediction, [311] which uses attributes shared across categories to prevent semantic drift in semi-supervised learning and [99] which automatically discovers attributes and uses them for fine-grained classification. We can think of this use of attributes for these applications as using a knowledge graph where the knowledge is simply a graph of visual concepts connected to their attributes (e.g. our knowledge is that elephants are grey, cats are furry, etc.). Attribute knowledge is in fact among the knowledge used in our work in Chapters 3&5 through the Visual Genome graph (see Section 2.4.1) so in some sense this line of work is a subset of ours.

Another line of early work on knowledge for vision was in using pretrained word vectors from large-scale language training. As we saw in Section 2.3.1, word vectors trained on billions of words or sentences can give a lot of prior knowledge about words and categories which can be used in visual recognition, or other tasks. One important early work in this was DeViSE [114] which demonstrated improved performance on ImageNet [86] and zero-shot learning. Some works, such as [178] even looked to create more visually aligned word embeddings by making use of both visual and textual data. Word embeddings have been shown in a wide variety of visual tasks including image retrieval [171], image captioning [171, 350] and visual relationship detection [205]. This technique is in fact used so frequently, especially in vision and language tasks (including our work in Chapter 5), that to try to cite all of the works which use it would be impossible.

Another line of work in vision has attempted to use a class hierarchy such as WordNet [235] (Section 2.4.1) as a source of external knowledge to aid in image recogntion [285, 290, 401]. More generally, knowledge graphs have also been used in visual classification [67] as well as zero-shot classification [360] and image retrieval [155]. In particular, our work in Chapter 3 was one of the earliest work which made use of the then new techniques in graph neural networks (Section 2.3.3) combining these with the existing interest in the CV community with knowledge. Prior to our contribution, several approaches to reasoning on graphs have been studied. For example, [401] collects a knowledge base and then queries this knowledge base to do first-order probabilistic reasoning to predict affordances. [220] builds a graph of exemplars for different categories and uses the spatial relationships to perform contextual reasoning. Approaches such as [184] use random walks on the graphs to learn patterns of edges while performing the walk and predict new edges in the knowledge graph. There has also been some work using a knowledge base for image retrieval [155] or answering visual queries [402]. Our main contribution was to shift the focus away from building specialized knowledge bases and towards existing knowledge bases, and to include end-to-end reasoning using graph networks rather than on query operations or other non-differentiable operations. After the publication of our work in Chapter 3, a number of works have also used knowledge graphs using graph neural networks or similar setups including for detection [67] and zero-shot classification [360].

2.7.2 VQA Methods

The work on knowledge-aware systems is perhaps best studied in computer vision in the language and vision community. Because success on VQA, especially those datasets specifically designed for the use of outside knowledge (Section 2.7.3), relies on systems having both visual recognition and the ability to reason about a question and a scene, knowledge-based methods have become popular.

Several approaches to using knowledge have been proposed. We can roughly break this into three categories: explicit knowledge graphs/bases, retrieval on source text or with large-scale language pre-training such as with BERT [90]. Several works have been studied which directly try to use knowledge graphs or knowledge bases [248, 249, 356, 357]. Some of these, however only handle the knowledge that is represented by subject-relation-object or visual concept-relation-attribute triplets, and rely on direct supervision to do the retrieval of the relevant knowledge. More recent methods such as [194] encorporate knowledge graphs without this supervision.

Most common are methods which rely on much of the implicit knowledge from large language models (Section 2.3.2) to improve performance by using a transformer architecture for both language and visual features [5, 68, 192, 195, 326, 334, 399]. In several works they not only pretrain on language tasks, but on language and vision tasks such as masking on image captions [316], or by training on several language and vision tasks simultaneously [70, 207, 208].

In contrast to these works, our contribution in Chapter 5 takes advantage of both the implicit and symbolic knowledge methods. We retain symbols until the end without the need of knowledge/fact annotations and integrate it with implicit knowledge and powerful reasoning abilities of multi-modal transformers. Concurrent with the publication of that work, others have also sought to combine these forms of knowledge. In ConceptBERT [120], a knowledge graph embedding is combined with the embedded output of a pretrained VilBERT [207] to produce a summary hidden state which is then used for classification. Similarly, in [309], the matching knowledge in a knowledge base is embedded using a vision and language transformer which are then fed into a pretrained transformer (LXMERT [334]). Unlike both of these approaches, we do not fuse the hidden representation of symbols (the knowledge graph node hidden states) until answer prediction. We discuss in detail Chapter 5 the positive effects of this choice.

2.7.3 VQA Tasks

Many works in the vision and language space have specifically considered knowledge as an inherent capability of these models. The task of visual question answering [16, 218] is a popular vision and language task where the VQA system is given a natural language question Q and an image I and must correctly answer ($a \in \mathcal{A}$) what the question asks in the context of the image. Several datasets for VQA have been proposed [16, 117, 156, 179, 218, 286, 335, 390, 403]. However, in many cases, these questions do not require much knowledge of the world, often just asking about the attributes of specific objects, counting objects in a scene, or about basic visual recognition. To better benchmark systems which incorporate visual and language reasoning as well as the capability to draw upon outside knowledge, several datasets and benchmarks have been proposed, including ours in Chapter 4.

Early work in benchmarking knowledge-based VQA began with KB-VQA [356] and FVQA [357]. While this work explicitly tackles this problem of knowledge-aware VQA benchmarking, these benchmarks annotate questions in a way that makes them "closed" rather than "open" knowledge systems. FVQA [357] is annotated by selecting a fact (a knowledge triplet such as "dog is mammal") from a fixed knowledge base and then writing a question around this fact. While this construction does force questions to require specific knowledge, because it is generated from a known fixed knowledge base, it generates questions which are quite easy to solve when the right knowledge is retrieved. It also does not test the ability of a system to draw upon outside knowledge without a fixed source of knowledge and without explicit annotation of the "correct" knowledge. These datasets also only test knowledge which can be retrieved as one triplet from a knowledge base while in datasets with more natural questions such as the ones from our contribution in Chapter 4 do not necessarily require triplet knowledge as the questions were written without an explicit correct piece of knowledge. Similarly, KVQA [304] is generated on top of Wikipedia articles with images and ask questions relating to that image and article. Because the images in Wikipedia articles tend to be depictions of famous people and important world events, this dataset also suffers from the questions mainly being about recognizing specific named entities (e.g.

Syama Prasad Mookerjee) in the image and retrieving facts about those same people.

A later line of work called VisualCOMET [261] tests similar capabilities as our dataset in Chapter 4. In [261] they introduce a dataset which requires systems to predict given an image: events that happened before the image, events that might happen after, and the intentions of the people. The Ads Dataset [146] is also a dataset requiring a lot of background knowledge, specifically about brands and even commonsense knowledge like "sunglasses symbolically indicate that something is cool" with knowledge-based approaches such as [385] being proposed to fill in the missing knowledge. Unlike our work, these works are not framed as visual question answering: these predictions take the form of requiring the generation of a commonsense graph for an image or as sentence outputs.

2.8 Applications of Knowledge in Interaction

Finally, as our motivation is to understand knowledge in *e*mbodied agents, we turn to the application of knowledge systems in robotics and other agents.[16] Here we will discuss work using textual knowledge and supervision in RL relating to our contribution in Chapter 6 where we use the implicit knowledge from language to solve zero and few-shot tasks. We will also discuss the work of direct or symbolic knowledge in robotics system relating to our work in Chapter 7 which uses background knowledge about categories and tasks to do semantic grasping on novel objects and tasks.

2.8.1 Language and Interaction

First, we look at the intersection of language and robotics and embodied agents. As we discussed in Section 1.1, knowledge can take the form of text or implicit knowledge from the structure of language or learned models such as word embeddings. Thus, works that look at this intersection by using language are also in some sense using knowledge and this work is relevant to our thesis. This implicit knowledge from language (and from word embeddings) also forms the basis of our work in Chapter 6.

[16]For lack of a better term, we will refer to the latter as "Interaction" or "RL"

The sub-field of language and vision navigation has investigated the use of natural language and action, training agents to navigate to a particular location in an environment (real or simulated) given templated or natural language [11, 29, 59, 62, 229, 336, 389] or to navigate to a particular location to answer a question [80]. In these works, the agent reads some text (both at train and test time) and follows these instructions towards some goal. In contrast, our work in Chapter 6 is not a translation task of semantics to action. Our agent uses language as supervision, but this is only provided as training data, and at test time, no instructions our given. Our agent must generate instructions from a high-level goal and then use those instructions to reach the goal.

Other related works such as [253] uses human-generated language to find objects in a simulated environment, [44, 396] read a document (i.e. a players manual) to play a variety of games, and [213] trains agents to follow both image and language-based goals. Others have utilized natural language for other tasks, including [362], [72], and [14] but not focused on the multi-task learning setting.

Many works, as ours in Chapter 6 are interested in the use of language as descriptions of tasks and sub-tasks, often with the hope of achieving some form of compositionality or generalization. Early work often relied on what [13] calls "sketches." A sketch specifies the necessary sub-tasks for a final task and is manually constructed for every task. The agent then relies on reward signals from the sketches in order to learn these predefined sub-tasks. [312] introduced what they call a Stochastic Temporal Grammar for multi-task RL in Minecraft. Similarly, BabyAI [69] creates a synthetic language inside a grid-based environment. [71] extends Hindsight Experience Replay (HER) to language goals in the BabyAI platform to solve a single instruction generated from a hand-crafted language. [152] also uses procedurally generated language on top of MuJoCo [342] and the CLEVR [156] engine to learn a hierarchical representation for multi-task RL. [257] also tackles zero-shot generalizations, but like the others considers only procedurally-generated instructions, learning to use analogies to learn correspondences between similar sub-tasks. All of these works rely on procedurally defined instructions. However, because these are pre-defined, it is less realistic for settings where we do not have a predefined ontology of tasks or simulator to generate the sketches, such as our work in Chapter 6 where we only have access to a limited number of human-generated instructions.

Most similar to our contribution is [142], which also investigates using a limited number of human-generated instructions in RL environments. This paper also uses natural language instructions in hierarchical decision making to play a real-time strategy game involving moving troop units across long time scales. The paper differs from ours in methods (using only imitation learning which fails in our setting) and in task definition. While their method has a wide variability in starting locations and placement of objects, they do not investigate whether language can be used in zero-shot or few-shot settings or in cross-task generalization as we do in Chapter 6.

A number of other language and action environments have been proposed which could benefit from both the implicit knowledge of large language models and symbolic knowledge. In [359], they create a language learning setting involving placing differently colored voxel blocks to make different objects. The computer initially knows nothing about language and learns it directly through interactions where a human player gives commands about what it should do (e.g. put down brown blocks on orange) and gives feedback. In a similar vein, agents in [139] are dropped into a simulated 3D world and must learn new tasks one-shot through human instructions and a access to language clues about objects. Similar to our environment in Chapter 6, ALFRED [310] requires the completion of high-level goals specified in language in a 3D simulated house and gives access to low-level demonstrations and language instructions for training. In situated (non-simulated) robotics, there have been a number of works which looked at teaching robots new tasks or goals through human demonstrations and language instructions [28, 57, 306].

2.8.2 Knowledge in Robotics and RL

Finally, we look at the use of explicit knowledge in robotics and other embodied agents, and place our work in Chapter 7 into this context. For a more complete survey on knowledge and semantic reasoning in robotics, see Liu et al. [201].

Text Adventure games such as Zork Nemesis and TextWorld [75] have recently emerged as an interesting test-bed for RL agents which require commonsense and language understanding [246]. These games were originally developed when rendering graphics in video games was prohibitively expensive and all perception and action happened through plain text. The player or agent receives descriptions of a scene through text (e.g. there is a door to your left) and takes actions by typing in the prompt window to take actions in the simulated world. Because the space of observations is textual, it can benefit enormously from semantic and knowledge-based techniques. In particular Ammanabrolu and Riedl [8] showed that representing the game state of text adventures as a knowledge graph improves learning and Ammanabrolu and Riedl [9] showed that adding prior knowledge from existing knowledge graphs would allow for better transfer to new games and genres of games (e.g. horror). Voice assistants, text or speech AI systems which can speak with humans and do useful tasks are very similar to this line of research as both the observation and action spaces are textual. Similarly, works such as [250] show that using external knowledge sources can improve these agents.

In the field of robotics, semantic knowledge has been used to help robots in adapting to diverse and changing environments and to provide priors and abstractions that help them generalize to new situations. As discussed in 2.4.1, knowledge bases have been specifically collected for robotics tasks including KnowRobo [339] and Robobrain [299], ORO [189], OUR-K [197], and RoboCSE [79]. Knowledge has been used for a variety of robotics tasks such as affordance learning [239, 346] (what actions can be done with different objects), active object search [394] (searching for a particular object in a room), plan repair [38] (autonomously correcting plans if a plan is invalid), semantic localization [116, 274] (room categories such as “kitchen”), and visual semantic navigation [377] (navigating to an object specified by language).

One line of work considers how to improve planning from background knowledge such as text or instructional videos. Kaiser et al. [159] looks at using recipes to learn background knowledge relevant for robotics-related planning problems. In [43], a model learns to predict precondition relations from text to learn the high level plan for crafting tasks (similar to our tasks in Chapter 6). Similarly, [58], takes instructional videos and learns a structured and plannable state and action spaces directly from those videos.

Most relevant to our contribution in Chapter 7 on semantic grasping is the line of work using for grasping. In Antanas et al. [15] and [17], grounded geometric information about objects paired with semantic information about the category of different parts of the object were used to improve semantic grasping. However, this requires fine-grained segmentation of objects and only reasons implicitly based on a limited number of segmentation labels. In contrast, our work in Chapter 7 does not rely on this kind of annotation and can reason based purely on the semantic category at the object level.

Chapter 3

Knowledge in Vision

3.1 Introduction

In this chapter, we build towards our goal of knowledge-capable embodied systems by first looking at how knowledge systems can be incorporated into deep learning-based visual perception systems. Our world contains millions of visual concepts understood by humans. These often are ambiguous (tomatoes can be red or green), overlap (vehicles includes both cars and planes) and have dozens or hundreds of subcategories (thousands of specific kinds of insects). While some visual concepts are common such as person or car, most categories have many fewer examples, forming a long-tail distribution [400]. And yet, even when only shown a few or even one example, humans have the remarkable ability to recognize these categories with high accuracy. In contrast, while modern learning-based approaches can recognize some categories with high accuracy, it usually requires thousands of labeled examples for each of these categories. Given how large, complex and dynamic the space of visual concepts is, this approach of building large datasets for every concept is unscalable. Therefore, we need to ask what humans have that current approaches do not.

One possible answer to this is structured knowledge and reasoning. Humans are not merely appearance-based classifiers; we gain knowledge of the world from experience and language. We use this knowledge in our everyday lives to recognize objects. For instance, we might have read in a book about the "elephant shrew" (maybe even seen an example) and will have gained knowledge that is useful for

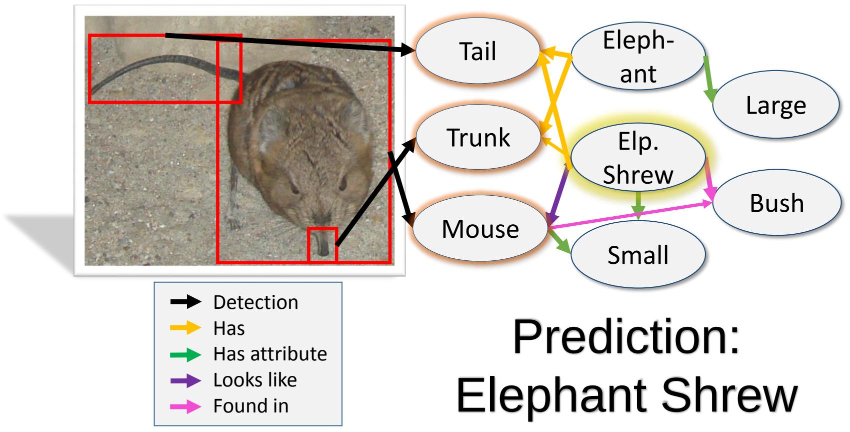


Figure 3.1: Example of how semantic knowledge about the world aids classification. Here we see an elephant shrew. Humans are able to make the correct classification based on what we know about the elephant shrew and other similar animals.

recognizing one. Figure 3.1 illustrates how we might use our knowledge about the world in this problem. We might know that an elephant shrew looks like a mouse, has a trunk and a tail, is native to Africa, and is often found in bushes. With this information, we could probably identify the elephant shrew if we saw one in the wild. We do this by first recognizing (we see a small mouse-like object with a trunk in a bush), recalling knowledge (we think of animals we have heard of and their parts, habitat, and characteristics) and then reasoning (it is an elephant shrew because it has a trunk and a tail, and looks like a mouse while mice and elephants do not have all these characteristics). With this information, even if we have only seen one or two pictures of this animal, we would be able to classify it.

There has been a lot of work in end-to-end learning on graphs or neural network trained on graphs [51, 100, 125, 138, 227, 231, 254, 300]. Most of these approaches either extract features from the graph or they learn a propagation model that transfers evidence between nodes conditional on the type of edge. An example of this is the Gated Graph Neural Network [196] which takes an arbitrary graph as input. Given some initialization specific to the task, it learns how to propagate information and predict the output for every node in the graph. This approach has been shown to solve basic logical tasks as well as program verification.

This work improves on this model and adapts end-to-end graph neural networks to multi-label image classification. We introduce the Graph Search Neural Network (GSNN) which uses features from the image to efficiently annotate the graph, select a relevant subset of the input graph and predict outputs on nodes representing visual concepts. These output states are then used to classify the objects in the image. GSNN learns a propagation model which reasons about different types of relationships and concepts to produce outputs on the nodes which are then used for image classification. Our architecture mitigates the computational issues with the Gated Graph Neural Networks for large graphs which allows our model to be efficiently trained for image tasks using large knowledge graphs. We show how our model is effective at reasoning about concepts to improve image classification tasks. Importantly, our GSNN model is also able to provide explanations on classifications by following how the information is propagated in the graph.

3.2 Related Work

Knowledge Bases in CV: Learning knowledge graphs [55, 65, 295] and using graphs for visual reasoning [220, 400] has long been of interest to the vision community. For reasoning on graphs, several approaches have been studied. For example, [401] collects a knowledge base and then queries this knowledge base to do first-order probabilistic reasoning to predict affordances. [220] builds a graph of exemplars for different categories and uses the spatial relationships to perform contextual reasoning. Approaches such as [184] use random walks on the graphs to learn patterns of edges while performing the walk and predict new edges in the knowledge graph. There had also been some work using a knowledge base for image retrieval [155] or answering visual queries [402], but these works were focused on building and then querying knowledge bases rather than using existing knowledge bases as side information for some vision task. However, none of these approaches had been learned in an end-to-end manner and the propagation model on the graph was mostly hand-crafted.

Graph Neural Networks: Learning from knowledge graphs using neural networks and other end-to-end learning systems to perform reasoning had recently become an active area of research. Several works treat graphs as a special case of a convolutional input where, instead of pixel inputs connected to pixels in a grid, we define the inputs as connected by an input graph, relying on either some global graph structure or doing some sort of pre-processing on graph edges [51, 100, 138, 254]. Li and Zemel present Graph Gated Neural Networks (GGNN) [196] which uses neural networks on graph structured data. This work (an extension of Graph Neural Networks [300]) serves as the foundation for our Graph Search Neural Network (GSNN). Several papers have found success using variants of Graph Neural Networks applied to various simple domains such as quantitative structure-property relationship (QSPR) analysis in chemistry [231] and subgraph matching and other graph problems on toy datasets [125]. GGNN is a fully end-to-end network that takes as input a directed graph and outputs either a classification over the entire graph or an output for each node. For instance, for the problem of graph reachability, GGNN is given a graph, a start node and end node, and the GGNN will have to output whether the end node is reachable from the start node. They show results for logical tasks on graphs and more complex tasks such as program verification.

Attributes: This work is also related to attribute approaches [106] to vision such as [183] which uses a fixed set of binary attributes to do zero-shot prediction, [311] which uses attributes shared across categories to prevent semantic drift in semi-supervised learning and [99] which automatically discovers attributes and uses them for fine-grained classification. Our work also uses attribute relationships that appear in our knowledge graphs, but also uses relationships between objects and reasons directly on graphs rather than using object-attribute pairs directly.

3.3 Methodology

In this section, we begin with an explanation of the Graph Gated Neural Network (Section 3.3.1) which serves as the basis of our network[1]. We then describe our network, the Graph Search Neural Network (Section 3.3.2) and describe our complete image processing pipeline and baselines (Section 3.3.3).

[1]We more thoroughly discuss prior graph neural network models in Chapter 2.3.3 and explain the recurrence equations there. But for clarity, we repeat parts of that section here.

3.3.1 Graph Gated Neural Network

The idea of GGNN is that given a graph with N nodes, we want to produce some output which can either be an output for every graph node $o_1, o_2, ...o_N$ or a global output o_G. This is done by learning a propagation model similar to an LSTM. For each node in the graph v, we have a hidden state representation $h_v^{(t)}$ at every time step t. We start at $t = 0$ with initial hidden states x_v that depends on the problem. For instance, for learning graph reachability, this might be a two bit vector that indicates whether a node is the source or destination node. In case of visual knowledge graph reasoning, x_v can be a one bit activation representing the confidence of a category being present based on an object detector or classifier.

Next, we use the structure of our graph, encoded in a matrix A which serves to retrieve the hidden states of adjacent nodes based on the edge types between them. The hidden states are then updated by a gated update module similar to an LSTM. The basic recurrence for this **propagation network** is:

$$h_v^{(1)} = [x_v^T, 0]^T \tag{3.1}$$

$$a_v^{(t)} = A_v^T [h_1^{(t-1)} ... h_N^{(t-1)}]^T + b \tag{3.2}$$

$$z_v^t = \sigma(W^z a_v^{(t)} + U^z h_v^{(t-1)}) \tag{3.3}$$

$$r_v^t = \sigma(W^r a_v^{(t)} + U^r h_v^{(t-1)}) \tag{3.4}$$

$$\widetilde{h_v^t} = tanh(W a_v^{(t)} + U(r_v^t \odot h_v^{(t-1)})) \tag{3.5}$$

$$h_v^{(t)} = (1 - z_v^t) \odot h_v^{(t-1)} + z_v^t \odot \widetilde{h_v^t} \tag{3.6}$$

where $h_v^{(t)}$ is the hidden state for node v at time step t, x_v is the problem specific annotation, A_v is the adjacency matrix of the graph for node v, and W and U are learned parameters. Eq 3.1 is the initialization of the hidden state with x_v and empty dimensions. Eq 3.2 shows the propagation updates from adjacent nodes. Eq (3-6) combine the information from adjacent nodes and current hidden state of the nodes to compute the next hidden state.

After T time steps, we have our final hidden states. The node level outputs can then just be computed as

$$o_v = g(h_v^{(T)}, x_v) \tag{3.7}$$

where g is a fully connected network, the **output network**, and x_v is the original annotation for the node.

3.3.2 Graph Search Neural Network

The biggest problem in adapting GGNN for image tasks is computational scalability. NEIL [65] for example has over 2000 concepts, and NELL [55] has over 2M confident beliefs. Even after pruning to our task, these graphs would still be huge. Forward propagation on the standard GGNN is $O(N^2)$ to the number of nodes N and backward propagation is $O(N^T)$ where T is the number of propagation steps. We perform simple experiments on GGNNs on synthetic graphs and find that after more than about 500 nodes, a forward and backward pass takes over 1 second on a single instance, even when making generous parameter assumptions. On 2,000 nodes, it takes well over a minute for a single image. Using GGNN out of the box is infeasible.

Our solution to this problem is the Graph Search Neural Network (GSNN). As the name might imply, the idea is that rather than performing our recurrent update over all of the nodes of the graph at once, we start with some initial nodes based on our input and only choose to expand nodes which are useful for the final output. Thus, we only compute the update steps over a subset of the graph. So how do we select which subset of nodes to initialize the graph with? During training and testing, we determine initial nodes in the graph based on likelihood of the concept being present as determined by an object detector or classifier. For our experiments, we use Faster R-CNN [287] for each of the 80 COCO categories. For scores over some chosen threshold, we choose the corresponding nodes in the graph as our initial set of active nodes.

Once we have initial nodes, we also add the nodes adjacent to the initial nodes to the active set. Given our initial nodes, we want to first propagate the beliefs about our initial nodes to all of the adjacent nodes. After the first time step, however, we need a way of deciding which nodes to expand next. We therefore learn a per-node scoring function that estimates how “important” that node is. After each propagation step, for every node in our current graph, we predict an importance score

$$i_v^{(t)} = g_i(h_v, x_v) \tag{3.8}$$

where g_i is a learned network, the **importance network**.

Once we have values of i_v, we take the top P scoring nodes that have never been expanded and add them to our expanded set, and add all nodes adjacent to those nodes to our active set. Figure 3.2 illustrates this expansion. At $t = 1$ only the detected nodes are expanded. At $t = 2$ we expand chosen nodes based on importance values and add their neighbors to the graph. At the final time step T we compute the per-node-output and re-order and zero-pad the outputs into the final classification net.

To train the importance net, we assign target importance value to each node in the graph for a given image. Nodes corresponding to ground-truth concepts in an image are assigned an importance value of 1. The neighbors of these nodes are assigned a value of γ. Nodes which are two-hop away have value γ^2 and so on. The idea is that nodes closest to the final output are the most important to expand.

We now have an end-to-end network which takes as input a set of initial nodes and annotations and outputs a per-node output for each of the active nodes in the graph. It consists of three sets of networks: the propagation net, the importance net, and the output net. The final loss from the image problem can be backpropagated from the final output of the pipeline back through the output net and the importance loss is backpropagated through each of the importance outputs. See Figure 3.3 to see the GSNN architecture. First x_{init}, the detection confidences initialize $h_{init}^{(1)}$, the hidden states of the initially detected nodes. We then initialize $h_{adj1}^{(1)}$, the hidden states of the adjacent nodes, with 0. We then update the hidden states using the propagation net. The values of $h^{(2)}$ are then used to predict the importance scores $i^{(1)}$, which are used to pick the next nodes to add $adj2$. These nodes are then initialized with $h_{adj2}^{(2)} = 0$ and the hidden states are updated again through the propagation net. After T steps, we then take all of the accumulated hidden states h^T to predict the GSNN outputs for all the active nodes. During backpropagation, the binary cross entropy (BCE) loss is fed backward through the output layer, and the importance losses are fed through the importance networks to update the network parameters.

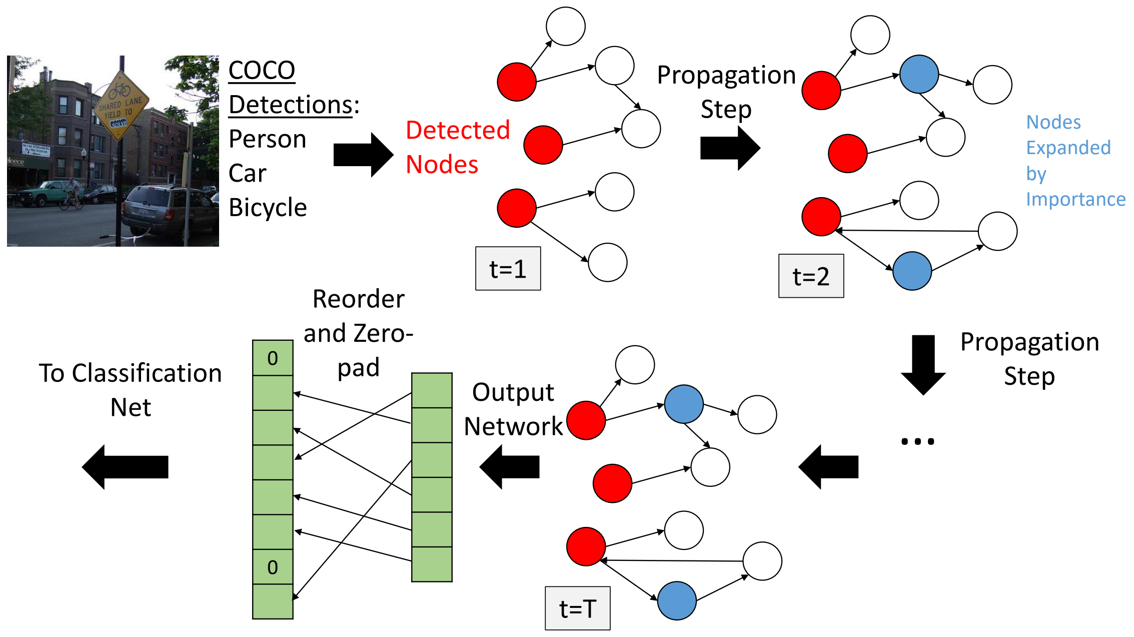


Figure 3.2: Graph Search Neural Network expansion. Starts with detected nodes and expands neighbors. Adds nodes adjacent to expand nodes predicted by importance net.

One final detail is the addition of a “node bias” into GSNN. In GGNN, the per-node output function $g(h_v^{(T)}, x_v)$ takes in the hidden state and initial annotation of the node v to compute its output. In a certain sense it is agnostic to the meaning of the node. That is, at train or test time, GSNN takes in a graph it has perhaps never seen before, and some initial annotations x_v for each node. It then uses the structure of the graph to propagate those annotations through the network and then compute an output. The nodes of the graph could have represented anything from human relationships to a computer program. However, in our graph network, the fact that a particular node represents “horse” or “cat” will probably be relevant, and we can also constrain ourselves to a static graph over image concepts. Hence we introduce node bias terms that, for every node in our graph, has some learned values. Our output equations are now $g(h_v^{(T)}, x_v, n_v)$ where n_v is a bias term that is tied to a particular node v in the overall graph. This value is stored in a table and its value are updated by backpropagation.

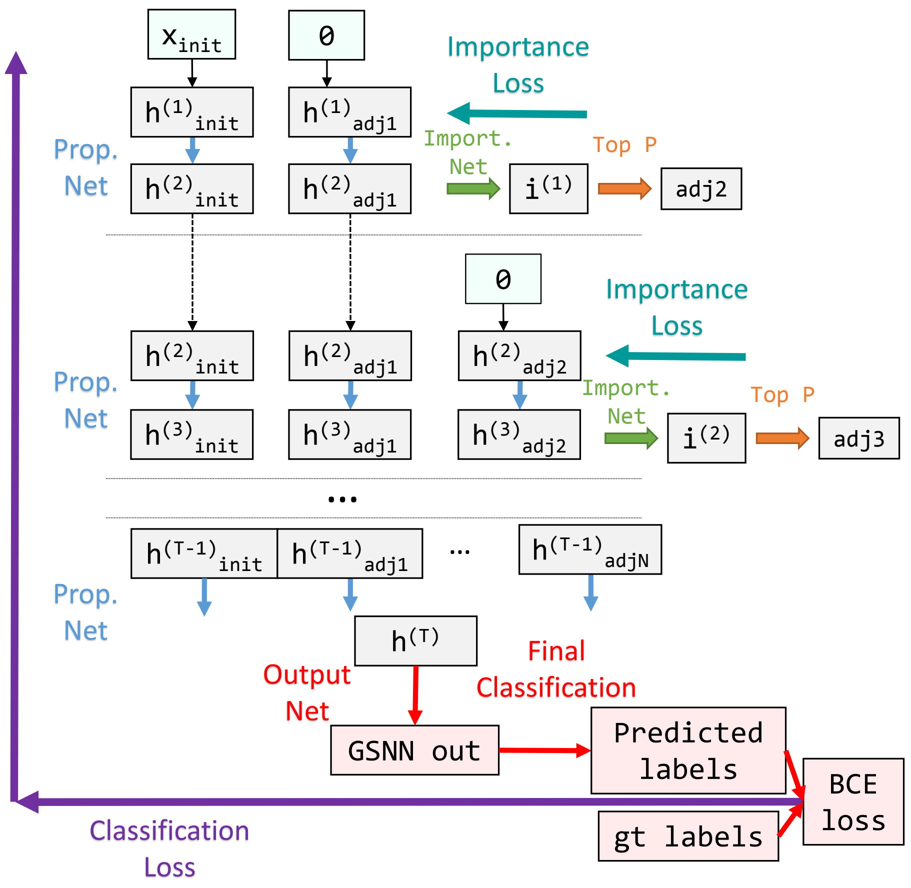


Figure 3.3: Graph Search Neural Network diagram. Shows initialization of hidden states, addition of new nodes as graph is expanded and the flow of losses through the output, propagation and importance nets.

3.3.3 Image Pipeline and Baselines

Another problem we face adapting graph networks for vision problems is how to incorporate the graph network into an image pipeline. For classification, this is fairly straightforward. We take the output of the graph network, reorder it so that nodes always appear in the same order into the final network, and zero pad any nodes that were not expanded. Therefore, if we have a graph with 316 node outputs, and each node predicts a 5-dim hidden variable, we create a 1580-dim feature vector from the graph. We also concatenate this feature vector with fc7 layer (4096-dim) of a fine-tuned VGG-16 network [314] and top-score for each COCO category predicted by Faster R-CNN (80-dim). This 5756-dim feature vector is then fed into 1-layer final classification network trained with dropout.

For baselines, we compare to: (1) VGG Baseline - feed just fc7 into final classification net; (2) Detection Baseline - feed fc7 and top COCO scores into final classification net.

3.3.4 Building the Knowledge Graph

We use Visual Genome [179] as a source for our knowledge graph. Using only the train split, we build a knowledge graph connecting the concepts using the most common object-attribute and object-object relationships in the dataset. Specifically, we counted how often an object-object relationship or object-attribute pair occurred in the training set, and pruned any edges that had fewer than 200 instances. This leaves us with a graph over all of the images with each edge being a common relationship. The idea is that we would get very common relationships (such as grass is green or person wears clothes) but not relationships that are rare and only occur in single images (such as person rides zebra).

The Visual Genome graphs are useful for our problem because they contain scene-level relationships between objects, e.g. person wears pants or fire hydrant is red and thus allow the graph network to reason about what is in a scene. However, it does not contain useful semantic relationships. For instance, it might be helpful to know that dog is an animal if our visual system sees a dog and one of our labels is animal. To address this, we also create a version of graph by fusing the Visual Genome Graphs with WordNet [235]. Using the subset of WordNet from [133], we first collect new nodes in WordNet not in our output label by including those which directly connect to our output labels and thus likely to be relevant and add them to a combined graph. We then take all of the WordNet edges between these nodes and add them to our combined graph.

3.4 Experiments

3.4.1 Datasets

For our experiments, we wanted to test on a dataset that represents the complex, noisy visual world with its many different kinds of objects, where labels are potentially ambiguous and overlapping, and categories fall into a long-tail distribution [400]. Humans do well in this setting, but vision algorithms still struggle with it. To this end, we chose the Visual Genome dataset [179][2].

Visual Genome contains over 100,000 natural images from the Internet. Each image is labeled with objects, attributes and relationships between objects entered by human annotators. Annotators could enter any object in the image rather than from a predefined list, so as a result there are thousands of object labels with some being more common and most having many fewer examples. There are on average 21 labeled objects in an image, so compared to datasets such as ImageNet [294] or PASCAL [102], the scenes we are considering are far more complex. Visual Genome is also labeled with object-object relationships and object-attribute relationships which we use for GSNN.

In our experiments, we create a subset from Visual Genome which we call Visual Genome multi-label dataset or VGML. In VGML, we take the 200 most common objects in the dataset and the 100 most common attributes and also add any COCO categories not in those 300 for a total of 316 visual concepts. Our task is then multi-label classification: for each image predict which subset of the 316 total categories appear in the scene. We randomly split the images into a roughly 80-20 train/test split. Since we used pre-trained detectors from COCO, we ensure none of our test images overlap with our detector's training images.

We also evaluate out method on the more standard COCO dataset [198] to show that our approach is useful on multiple datasets and that our method does not rely on graphs built specifically for our datasets. We train and test in the multi-label setting [237], and evaluate on the minival set [287].

[2]v1.0

Table 3.1: Mean Average Precision for multi-label classification on Visual Genome Multi-Label dataset. Numbers for VGG baseline, VGG baseline with detections, GSNN using Visual Genome graph and GSNN using a combined Visual Genome and WordNet graph.

Method	mAP
VGG	30.57
VGG+Det	31.40
GSNN-VG	32.83
GSNN-VG+WN	**33.00**

Table 3.2: Mean Average Precision for multi-label classification on COCO. Numbers for VGG baseline, VGG baseline with detections, GSNN using Visual Genome graph and GSNN using combined Visual Genome and WordNet graph.

Method	mAP
VGG	69.86
VGG+Det	73.93
GSNN-VG	**77.57**
GSNN-VG+WN	75.73

3.4.2 Training Details

We jointly train all parts of the pipeline (except for the detectors). All models are trained with Stochastic Gradient Descent, except GSNN which is trained using ADAM [168]. We use an initial learning rate of 0.05, 0.005 for the VGG net before $fc7$, decreasing by a factor of 0.1 every 10 epochs, an L2 penalty of $1e^{-6}$ and a momentum of 0.5. We set our GSNN hidden state size to 10, importance discount factor γ to 0.3, number of time steps T to 3, initial confidence threshold to 0.5 and our expand number P to 5. Our GSNN importance and output networks are single layer networks with sigmoid activations. All networks were trained for 20 epochs with a batch size of 16.

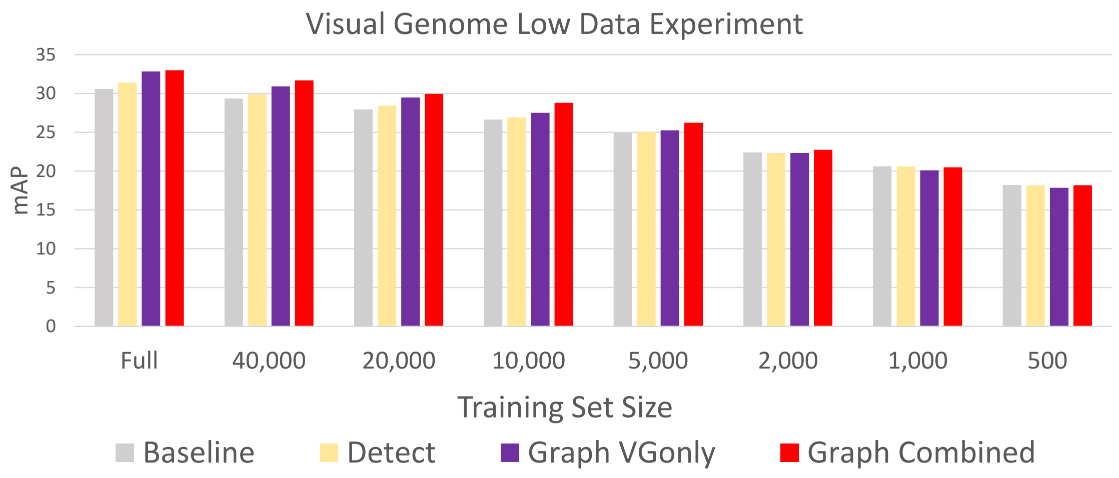


Figure 3.4: Mean Average Precision on Visual Genome in the low data setting. Shows performance for all methods for the full dataset, 40,000, 20,000, 10,000, 5,000, 2,000, 1,000, and 500 training examples.

3.4.3 Quantitative Evaluation

Table 3.1 shows the result of our method on Visual Genome multi-label classification. In this experiment, the combined Visual Genome, WordNet graph outperforms the Visual Genome graph. This suggests that including the outside semantic knowledge from WordNet and performing explicit reasoning on a knowledge graph allows our model to learn better representations compared to the other models.

We also perform experiments to test the effect of limiting the size of the training dataset has on performance. Figure 3.4 shows the results of this experiment on Visual Genome, varying the training set size from the entire training set (approximately 80,000), all the way down to 500 examples. Choosing the subsets of examples for these experiments is done randomly, but each training set is a subset of the larger ones—e.g. all of the examples in the 1,000 set are also in the 2,000 set. We see that, until the 1,000 sample set, the GSNN-based methods all outperform baselines. At 1,000 and 500 examples, all of the methods perform equally. Given the long-tail nature of Visual Genome, it is likely that for fewer than 2,000 samples, many categories do not have enough examples for any method to learn well. This experiment indicates that our method is able to improve even in the low-data case up to a point.

In Table 3.2, we show results on the COCO multi-label dataset. We can see that the

Table 3.3: Mean Average Precision for multi-label classification on COCO, using only odd and even detectors.

Method	even mAP	odd mAP
VGG+Det	71.87	71.73
GSNN-VG	73.00	73.43
GSNN-VG+WN	**73.59**	**73.97**

boost from using graph knowledge is more significant than it was on Visual Genome. One possible explanation is that the Visual Genome knowledge graph provides significant information which helps improve the performance on the COCO dataset itself. In the previous Visual Genome experiment, much of the graph information is contained in the labels and images themselves. One other interesting result is that the Visual Genome graph outperforms the combined graph for COCO, though both outperform baselines. One possible reason is that the original VGML graph is smaller, cleaner, and contains more relevant information than the combined graph. Furthermore, in the VGML experiment, WordNet is new outside information for the algorithm helping boost the performance.

One possible concern is the over-dependence of the graph reasoning on the set of 80 COCO detectors and initial detections. Therefore, we performed an ablation experiment to see how sensitive our method is to having all of the initial detections. We reran the COCO experiments with both graphs using two different subsets of COCO detectors. The first subset is just the even COCO categories and the second subset is just the odd categories. We see from Table 3.3 that GSNN methods again outperform the baselines.

As one might suspect, our method does not perform uniformly on all categories, but rather does better on some categories and worse on others. Figure 3.5 shows the differences in average precision for each category between our GSNN model with the combined graph and the detection baseline for the VGML experiment. Figure 3.6 shows the same for our COCO experiment. Performance on some classes improves greatly, such as "fork" in our VGML experiment and "scissors" in our COCO experiment. These and other good results on "knife" and "toothbrush" seem to indicate that the graph reasoning helps especially with small objects in the image.

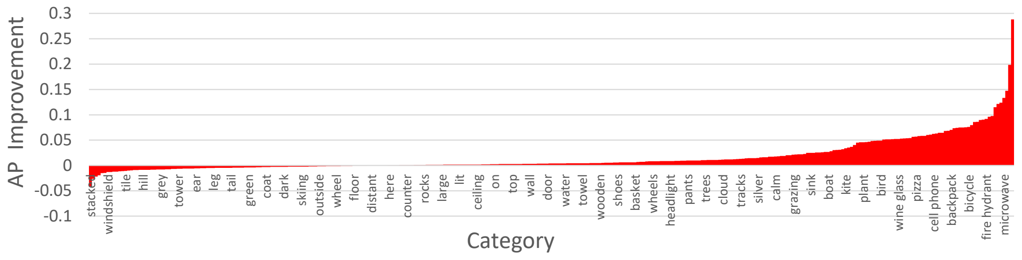


Figure 3.5: Difference in Average Precision for each of the 316 labels in VGML between our GSNN combined graph model and detection baseline for the Visual Genome experiment. Top categories: scissors, donut, frisbee, microwave, fork. Bottom categories: stacked, tiled, light brown, ocean, grassy.

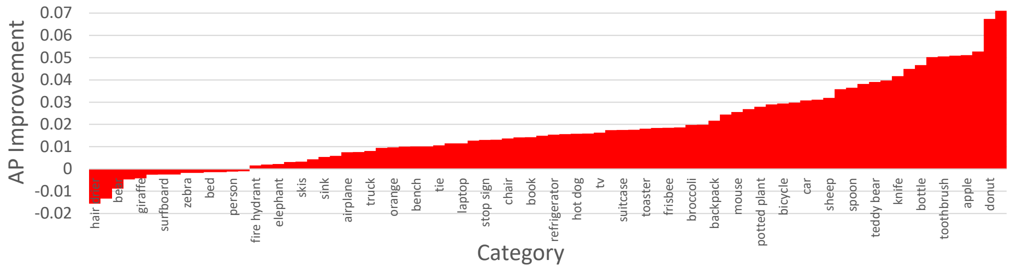


Figure 3.6: Difference in Average Precision for each of the 80 labels in COCO between our GSNN VG graph model and detection baseline for the COCO experiment. Top categories: fork, donut, cup, apple, microwave. Bottom categories: hairdryer, parking meter, bear, kite, and giraffe.

In the next section, we analyze our GSNN models on several examples to try to gain a better intuition as to what the GSNN model is doing and why it does well or poorly on certain examples.

3.4.4 Qualitative Evaluation

One way to analyse the GSNN is to look at the sensitivities of parameters in our model with respect to a particular output. Given a single image I, and a single label of interest y_i that appears in the image, we would like to know how information travels through the GSNN and what nodes and edges it uses. We examined the sensitivity of the output to hidden states and detections by computing the partial derivatives $\frac{\partial y_i}{\partial h^{(1)}}$ $\frac{\partial y_i}{\partial h^{(2)}}$ $\frac{\partial y_i}{\partial x_{det}}$ with respect to the category of interest. These values tell

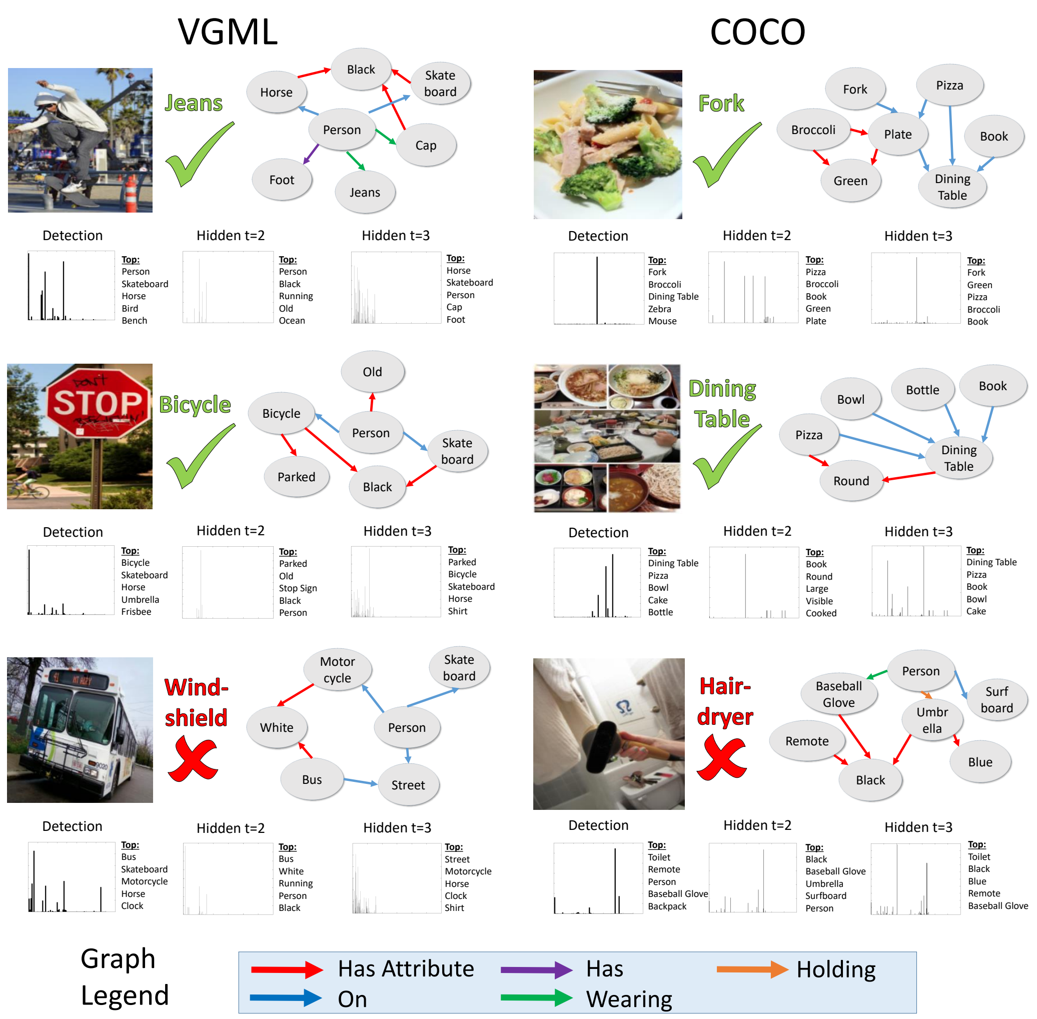


Figure 3.7: Sensitivity analysis of GSNN in VGML experiment (left) and COCO experiment (right) with the combined graph and Visual Genome graphs respectively. Each example shows the image, part of the knowledge graph expanded during the classification, and the sensitivity values of the initial detections, and the hidden states at time steps 2 and 3 with respect to the output class listed. The top detections and hidden state nodes are printed for convenience since the x-axis is too large to list every class. The top and middle rows show the results for images and classes where the GSNN significantly outperforms the detection baseline to get an intuition for when our method is working. The bottom row shows images and classes where GSNN does worse than the detection baseline to get an idea of when our method fails and why.

us how a small change in the hidden state of a particular node affects a particular output. We would expect to see, for instance, that for labeling elephant, we see a high sensitivity for the hidden states corresponding to grey and trunk.

In this section, we show the sensitivity analysis for the GSNN combined graph model on the VGML experiment and the Visual Genome graph on the COCO experiments. In particular, we examine some classes that performed well under GSNN compared to the detection baseline and a few that performed poorly to try to get a better intuition into why some categories improve more.

Figure 3.7 shows the graph sensitivity analysis for the experiments with VGML on the left and COCO on the right, showing four examples where GSNN does better and two where it does worse. Each example shows the image, the ground truth output we are analyzing and the sensitivities of the concept of interest with respect to the hidden states of the graph or detections. For convenience, we display the names of the top detections or hidden states. We also show part of the graph that was expanded, to see what relationships GSNN was using.

For the VGML experiment, the top left of Figure 3.7 shows that using the detection for person, GSNN is able to reason that jeans are more likely since jeans are usually on people in images using the "wearing" edge. It is also sensitive to skateboard and horse, and each of these has a second order connection to jeans through person, so it is likely able to capture the fact that people tend to wear jeans while on horses and skateboards. Note that the sensitivities are not the same as the actual detections, so it is not contradictory that horse has high sensitivity. The second row on the left shows a successful example for bicycle, using detections from person and skateboard and the fact that people tend to be "on" bicycles and skateboards. The last row shows a failure case for windshield. It correctly correlates with bus, but because the knowledge graph lacks a connection between bus and windshield, the graph network is unable to do better than the detection baseline. On the right, for the COCO experiment, the top example shows that fork is highly correlated with the detection for fork, which should not be surprising. However, it is able to reinforce this detection with the connections between broccoli and dining table, which are both two step connections to fork on the graph. Similarly, the middle example shows that the graph connections for pizza, bowl, and bottle being "on" dining table reinforce the detection of dining table. The bottom right shows another failure case. It is able to get the connection between the detection for toilet and hair dryer (both found in the bathroom), but the lack of good connections in the graph prevent the GSNN from improving over the baseline.

3.5 Conclusion

In this chapter, we present the Graph Search Neural Network (GSNN) as a way of efficiently using knowledge graphs as extra information to improve image classification. We provide analysis that examines the flow of information through the GSNN and provides insights into why our model improves performance.

In later chapters, we will extend on many of the ideas from this work of how to incorporate knowledge graphs into deep learning systems, but we extend these ideas to join multiple modalities and study problems which require more knowledge about the world.

Chapter 4

Knowledge in Language and Vision: Benchmarks

4.1 Introduction

In this chapter, we extend our study of knowledge systems to those that operate across multiple modalities, in this case vision and language. In particular, we also wanted to study problems where the the benefits of knowledge are more obvious.

Consider the question in Figure 4.1, which asks about the relation between the teddy bear and an American president. The information in the image here is not complete for answering the question. We need to link the image content to external knowledge sources, such as the sentences at the bottom of the figure taken from Wikipedia. Given the question, image, and Wikipedia sentences, there is now enough information to answer the question: Teddy Roosevelt!

Prior research had started to look at how to incorporate knowledge-based methods into VQA [248, 249, 356, 357]. These methods investigated incorporating knowledge bases and retrieval methods into VQA datasets with a set of associated facts for each question. In this contribution, we go one step forward and design a VQA dataset which requires VQA to perform reasoning using unstructured knowledge.

To enable research in this direction, we introduce a dataset, named Outside Knowledge VQA (OK-VQA), which includes only questions that require external

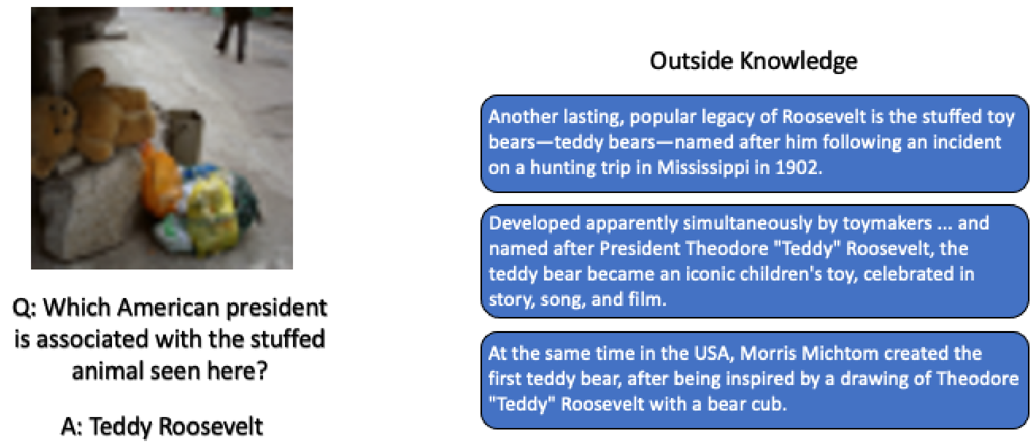

Q: Which American president is associated with the stuffed animal seen here?
A: Teddy Roosevelt
Outside Knowledge
Another lasting, popular legacy of Roosevelt is the stuffed toy bears—teddy bears—named after him following an incident on a hunting trip in Mississippi in 1902.
Developed apparently simultaneously by toymakers ... and named after President Theodore "Teddy" Roosevelt, the teddy bear became an iconic children's toy, celebrated in story, song, and film.
At the same time in the USA, Morris Michtom created the first teddy bear, after being inspired by a drawing of Theodore "Teddy" Roosevelt with a bear cub.

Figure 4.1: We propose a novel dataset for visual question answering, where the questions require external knowledge resources to be answered. In this example, the visual content of the image is not sufficient to answer the question. A set of facts about teddy bears makes the connection between teddy bear and the American president, which enables answering the question.

resources for answering them. On our dataset, we can start to evaluate the reasoning capabilities of models in scenarios where the answer cannot be obtained by only looking at the image. Answering OK-VQA questions is a challenging task since, in addition to understanding the question and the image, the model needs to: (1) learn what knowledge is necessary to answer the questions, (2) determine what query to do to retrieve the necessary knowledge from an outside source of knowledge, and (3) incorporate the knowledge from its original representation to answer the question.

The OK-VQA dataset consists of more than 14,000 questions that cover a variety of knowledge categories such as science & technology, history, and sports. We provide category breakdowns of our dataset, as well as other relevant statistics to examine its properties. We also analyze the then-standard VQA models and show their performance degrades on this dataset. Furthermore, we provide results for a set of baseline approaches that are based on simple knowledge retrieval. Our dataset is diverse, difficult, and to date the largest VQA dataset focused on knowledge-based VQA in natural images.

4.2 Related Work

Visual Question Answering (VQA). Visual question answering (VQA) has been one of the most popular topics in the computer vision community over the past few years. Early approaches to VQA combined recurrent networks with CNNs to integrate textual and visual data [219]. Attention-based models [115, 206, 370, 373, 381, 403] better guide the model in answering the questions by highlighting image regions that are relevant to the question. Modular networks [12, 143, 157] leverage the compositional nature of the language in deep neural networks. These approaches have been extended to the video domain as well [150, 242, 344]. [80, 124] address the problem of question answering in an interactive environment. None of these approaches, however, is designed for leveraging external knowledge so they cannot handle the cases that the image does not represent the full knowledge to answer the question.

The problem of using external knowledge for answering questions had been tackled by [193, 248, 249, 356, 357, 365]. These methods, however, only handle the knowledge that is represented by subject-relation-object or visual concept-relation-attribute triplets, and rely on supervision to do the retrieval of facts. In contrast, answering questions in our dataset requires handling unstructured knowledge resources.

VQA datasets. In the past decade, several datasets have been proposed for visual question answering [16, 117, 156, 179, 218, 286, 335, 357, 390, 403]. The DAQUAR dataset [218] includes template-based and natural questions for a set of indoor scenes. [16] proposed the VQA dataset, which is two orders of magnitude larger than DAQUAR and includes more diverse images and less constrained answers. FM-IQA [117] is another dataset that includes multi-lingual questions and answers. Visual Madlibs [390], constructs fill-in-the-blank templates for natural language descriptions. COCO-QA [286] is constructed automatically by converting image descriptions to questions. The idea of Visual 7W [403] is to provide object-level grounding for question-answer pairs as opposed to image-level associations between images and QA pairs. Visual Genome [179] provides dense annotations for image regions, attributes, relationships, etc. and provide free-form and region-based QA pairs for each image. MovieQA [335] is a movie-based QA dataset, where the QAs are based on information in the video clips, subtitles, scripts, etc. CLEVR [156] is a synthetic VQA dataset that mainly targets visual reasoning abilities. In contrast to all these datasets, we focus on questions that cannot be answered by the information in the associated image and require external knowledge to be answered.

Most similar to our dataset is FVQA [357]. While that work also tackles the difficult problem of creating a VQA dataset requiring outside knowledge, their method annotates questions by selecting a fact (a knowledge triplet such as "dog is mammal") from a fixed knowledge base. While this dataset is still quite useful for testing methods' ability to incorporate a knowledge base into a VQA system, our dataset tests methods' ability to retrieve relevant facts from the web, from a database, or some other source of knowledge that was not used to create the questions. Another issue is that triplets are not sufficient to represent general knowledge.

Building knowledge bases & Knowledge-based reasoning. Several knowledge bases have been created using visual data or for visual reasoning tasks [65, 94, 295, 401, 402, 404]. These knowledge bases are potentially helpful resources for answering questions in our dataset. Knowledge-based question answering has received much more attention in the NLP community (e.g., [23, 34, 61, 382, 387]).

4.3 Dataset Collection

In this section we explain how we collect a dataset which better measures performance of VQA systems requiring external knowledge. The common VQA datasets such as [16, 126] do not require much knowledge to answer a large majority of the questions. The dataset mostly contains questions such as "How many apples are there?", "What animal is this?", and "What color is the bowl?". While these are perfectly reasonable tasks for open-ended visual recognition, they do not test our algorithms' ability to reason about a scene or draw on information outside of the image. Thus, for our goal of combining visual recognition with information extraction from sources outside the image, we would not be able to evaluate knowledge-based systems as most questions do not require outside knowledge.

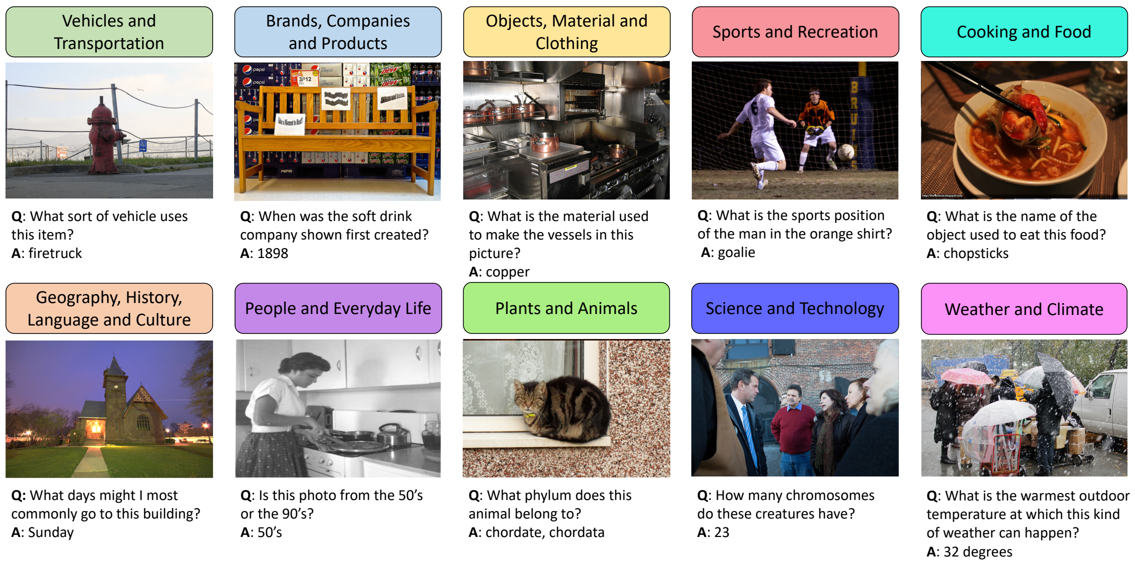
To see this specifically, we examine the "age annotations" that are provided for 10,000 questions in the VQA dataset [3]. For each question and image pair, an MTurk worker was asked how old someone would need to be to answer the question. While this is not a perfect metric, it is a reasonable approximation of the difficulty of a question and how much a person would have to know to answer a question. The analysis shows that more than 78% of the questions can be answered by people who are 10 years old or younger. This suggests that very little background knowledge is actually required to answer the vast majority of these questions.

Given that current VQA datasets do not test exactly what we are looking for, we collect a new dataset. We use random images from the COCO dataset [198], using the original 80k-40k training and validation splits for our train and test splits. The visual complexity of these images compared to other datasets make them ideal for labeling knowledge-based questions.

In the first round of labeling, we asked MTurk workers to write a question given an image. Similar to [16], we prompt users to come up with questions to fool a "smart robot." We also ask in the instructions that the question should be related to the image content. In addition, we prompt users not to ask what is in an image, or how many of something there is, and specify that the question should require some outside knowledge. In a second round of labeling, we asked 5 different MTurk workers to label each question-image pair with an answer.

Although this prompt yielded many high-quality questions, it also yielded a lot of low quality questions, for example, ones that asked basic questions such as counting, did not require looking at the image, or were nonsensical. To ensure that the dataset asked these difficult knowledge-requiring questions, the MTurk provided questions were manually filtered to get only questions requiring knowledge. From a pool of 86,700 questions, we filtered down to 34,921 questions.

One more factor to consider was the potential bias in the dataset. As discussed in many works, including [126], the VQAv1 dataset had a lot of bias. Famously, questions beginning with "Is there a ..." had a very strong bias towards "Yes." Similarly, in our unfiltered dataset, there were a lot of questions with a bias towards certain answers. For instance, in a lot of images where there is snowfall, the question would ask "What season is it?" Although there were other images (such as ones with deciduous trees with multi-colored leaves) with different answers, there was a clear bias towards "winter." To alleviate this problem, for train and test, we removed questions so that the answer distribution was uniform; specifically, we removed questions if there were more than 5 instances of that answer as the most common answer. This had the effect of removing a lot of the answer bias. It also had the effect of making the dataset


Vehicles and Transportation
Brands, Companies and Products
Objects, Material and Clothing
Sports and Recreation
Cooking and Food
Geography, History, Language and Culture
People and Everyday Life
Plants and Animals
Science and Technology
Weather and Climate

Q: What sort of vehicle uses this item?
A: firetruck

Q: When was the soft drink company shown first created?
A: 1898

Q: What is the material used to make the vessels in this picture?
A: copper

Q: What is the sports position of the man in the orange shirt?
A: goalie

Q: What is the name of the object used to eat this food?
A: chopsticks

Q: What days might I most commonly go to this building?
A: Sunday

Q: Is this photo from the 50's or the 90's?
A: 50's

Q: What phylum does this animal belong to?
A: chordate, chordata

Q: How many chromosomes do these creatures have?
A: 23

Q: What is the warmest outdoor temperature at which this kind of weather can happen?
A: 32 degrees

Figure 4.2: **Dataset examples.** Some example questions and their corresponding images and answers have been shown. We show one example question for each knowledge category.

harder by limiting the number of times VQA algorithms would see questions with a particular answer, making outside information more important. We also removed questions which had no inter-annotator agreement on the answer as we found that this was most often a sign that the question was too ambiguous. Performing this filtering brought us down to 9,009 questions in train and 5,046 questions in test for a total of 14,055 questions.

Figure 4.2 shows some of the collected questions, images, and answers from our dataset. You can see that these questions require at least one piece of background knowledge to answer. For instance, in the bottom left question, the system needs to recognize that the image is of a Christian church and know that those churches hold religious services on Sundays. That latter piece of knowledge should be obtained from external knowledge resources, and it cannot be inferred from the image and question alone.

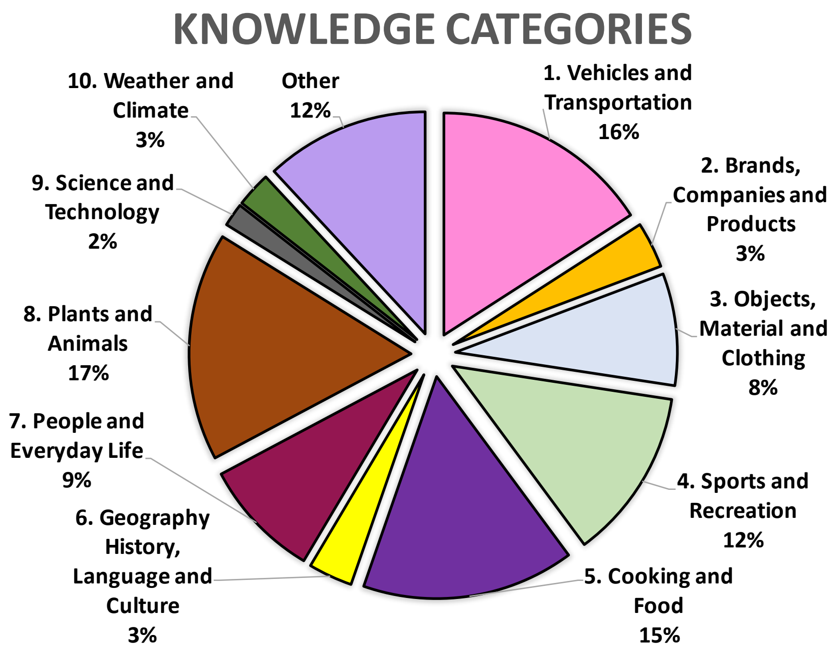


Figure 4.3: **Breakdown of questions in terms of knowledge categories.** We show the percentage of questions falling into each of our 10 knowledge categories.

4.4 Dataset Analysis

In this section, we explore the statistical properties of our dataset, and compare to other visual question answering datasets to show that our dataset is diverse, difficult, and, to the best of our knowledge, the largest VQA dataset specifically targeted for knowledge-based VQA on natural scenes.

4.4.1 Knowledge Categories

Requiring knowledge for VQA is a good start, but there are many different types of knowledge that humans have about the world that could come into play. There is commonsense knowledge: water is wet, couches are found in living rooms. There is geographical knowledge: the Eiffel Tower is in Paris, scientific knowledge: humans have 23 chromosomes, and historical knowledge: George Washington is the first U.S. president. To get a better understanding of the kinds of knowledge our dataset requires, we asked five MTurk workers to annotate each question as belonging to one

	Number of questions	Number of images	Knowledge based?	Answer type	Avg. A length	Avg. Q length
DAQUAR [218]	12,468	1,449	✘	Open	1.1	11.5
Visual Madlibs [390]	360,001	10,738	✘	FITB/MC	2.8	4.9
Visual 7W [403]	327,939	47,300	✘	MC	2.0	6.9
VQA (v2) [126]	1.1M	200K	✘	Open/MC	1.2	6.1
MovieQA [335]	14,944	408V	✘	MC	5.3	9.3
CLEVR [156]	999,968	100,000	✘	Open	1.0	18.4
KB-VQA [356]	2,402	700	✓	Open	2.0	6.8
FVQA [357]	5,826	2,190	✓	Open	1.2	9.5
OK-VQA (ours)	14,055	14,031	✓	Open	1.3	8.1

Table 4.1: **Comparison of various visual QA datasets.** We compare OK-VQA with some other VQA datasets. The bottom three rows correspond to knowledge-based VQA datasets. A length: answer length; Q length: question length; MC: multiple choice; FITB: fill in the blanks; KB: knowledge base.

of ten categories of knowledge that we specified: Vehicles and Transportation; Brands, Companies and Products; Objects, Materials and Clothing; Sports and Recreation; Cooking and Food; Geography, History, Language and Culture; People and Everyday Life, Plants and Animals; Science and Technology; and Weather and Climate. If no one category had a plurality of workers, it was categorized as “Other”. This also ensured that the final category labels are mutually exclusive. We show the distribution of questions across categories in Figure 4.3.

4.4.2 Comparison with Other VQA Datasets

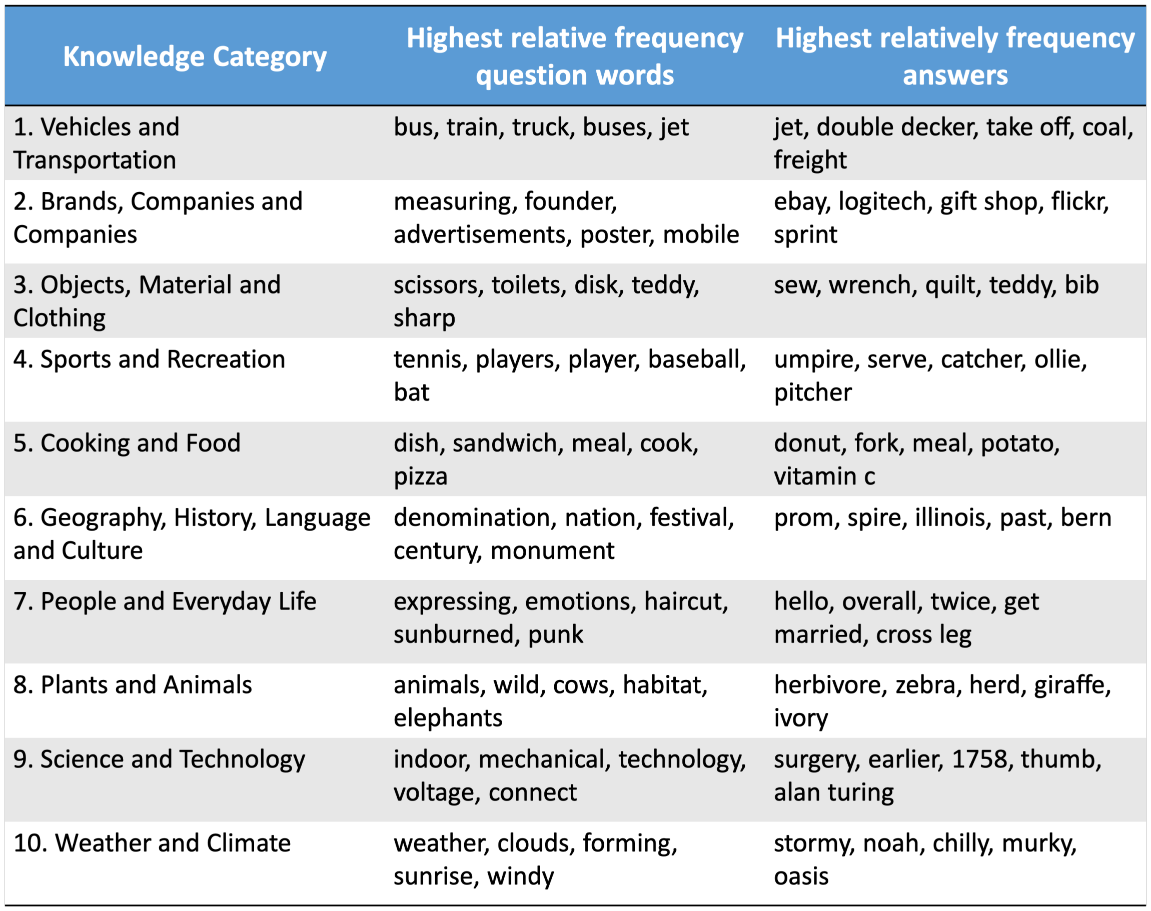
In Table 4.1 we look at a number of other visual question answering datasets and compare them to our dataset in a number of different ways. In the top section, we look at a number of datasets which do not explicitly try to include a knowledge component including the ubiquitous VQAv2 dataset [126], the first version of which was one of the first datasets to investigate visual question answering. Compared to these datasets, we have a comparable number of questions to DAQUAR [218] as well as MovieQA [335], and many more questions than knowledge-based datasets KB-VQA [356] and FVQA [357]. We have fewer questions compared to CLEVR [156] where the images, questions and answers are automatically generated, as well compared to more large-scale human annotated visual datasets such as VQAv2 [126], and Visual Madlibs [390]. Since we manually filtered our dataset to avoid the pitfalls of other datasets and to ensure our questions are knowledge-based and because we filtered down common answers to emphasize the long tail of answers, our dataset is more time-intensive and expensive to collect. We trade off size in this case for knowledge and difficulty.

We can see from the average question lengths and average answer lengths that our questions and answers are about comparable to KB-VQA [356] and FVQA [357] and longer than the other VQA datasets with the exception of DAQUAR and CLEVR (which are partially and fully automated from templates respectively). This makes sense since we would expect knowledge-based questions to be longer as they are typically not able to be as short as common questions in other datasets such as "How many objects are in the image?" or "What color is the couch?".

4.4.3 Question Statistics

We also collected statistics for our dataset by looking at the number of questions, and by looking at which were most frequent for each knowledge category. OK-VQA has 12,591 unique questions out of 14,055 total, and 7,178 unique question words. This indicates that we get a variety of different questions and answers in our dataset.

Finally, we show in Figure 4.4 question words and answers in each category that are most "unique" to get a better idea of what types of questions we have in each categories. We calculate these for each knowledge category by looking at the number of appearances within the category over the total number in the dataset to see which question words and answers had the highest relative frequency in their category. In question words, we see words specific to categories such as bus in Vehicles and Transportation, sandwich in Cooking and Food, and clouds in Weather and Climate. We also see that the answers are also extremely related to each category, such as herbivore in Plants and Animals, and umpire in Sports and Recreation.



Knowledge Category	Highest relative frequency question words	Highest relatively frequency answers
1. Vehicles and Transportation	bus, train, truck, buses, jet	jet, double decker, take off, coal, freight
2. Brands, Companies and Companies	measuring, founder, advertisements, poster, mobile	ebay, logitech, gift shop, flickr, sprint
3. Objects, Material and Clothing	scissors, toilets, disk, teddy, sharp	sew, wrench, quilt, teddy, bib
4. Sports and Recreation	tennis, players, player, baseball, bat	umpire, serve, catcher, ollie, pitcher
5. Cooking and Food	dish, sandwich, meal, cook, pizza	donut, fork, meal, potato, vitamin c
6. Geography, History, Language and Culture	denomination, nation, festival, century, monument	prom, spire, illinois, past, bern
7. People and Everyday Life	expressing, emotions, haircut, sunburned, punk	hello, overall, twice, get married, cross leg
8. Plants and Animals	animals, wild, cows, habitat, elephants	herbivore, zebra, herd, giraffe, ivory
9. Science and Technology	indoor, mechanical, technology, voltage, connect	surgery, earlier, 1758, thumb, alan turing
10. Weather and Climate	weather, clouds, forming, sunrise, windy	stormy, noah, chilly, murky, oasis

Figure 4.4: For each category we show the question words and answers that have the highest relative frequency across our knowledge categories (i.e. frequency in category divided by overall frequency).

4.5 Experiments

In this section, we evaluate various VQA approaches and provide results for some baselines, including knowledge-based ones.

4.5.1 Baselines

MUTAN [21]

Multimodal Tucker Fusion (MUTAN) model [21], a tensor-based method for VQA. Specifically, we use the attention version of MUTAN, and choose the parameters to match the single best performing model of [21].

BAN [166]

Bilinear Attention Networks for VQA. A VQA method that uses a co-attention mechanism between the question features and the bottom-up detection features of the image.

MLP

The MLP has 3 hidden layers with ReLU activations and hidden size 2048 that concatenates the image and question features after a skip-thought GRU after one fully connected layer each. Like MUTAN, it uses fc7 features from ResNet-152.

Q-Only

The same model as MLP, but only takes the question features.

ArticleNet (AN)

We consider a simple knowledge-based baseline that we refer to as ArticleNet. The idea is to retrieve some articles from Wikipedia for each question-image pair and then train a network to find the answer in the retrieved articles.

Retrieving articles is composed of three steps. First, we collect possible search queries for each question-image pair. We come up with all possible queries for each question by combining words from the question and words that are identified by pre-trained image and scene classifiers. Second, we use the Wikipedia search API to get the top retrieved article for each query. Third, for each query and article, we extract a small subset of each article that is most relevant for the query by selecting the sentences within the article that best correspond to our query based on the frequency of those query words in the sentence.

Once the sentences have been retrieved, the next step is to filter and encode them for use in VQA. Specifically, we train ArticleNet to predict whether and where the ground truth answers appear in the article and in each sentence. The architecture is shown in Figure 4.5. To find the answer to a question, we pick the top scoring

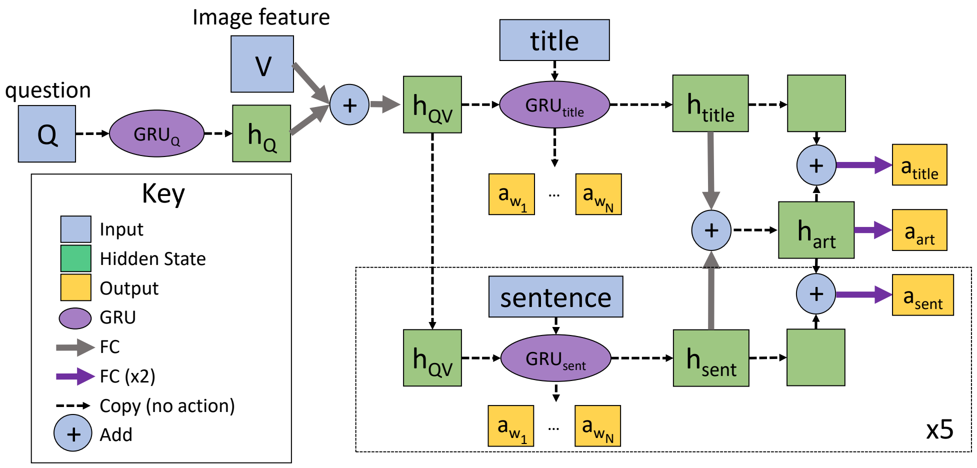


Figure 4.5: **ArticleNet architecture.** ArticleNet takes in the question Q and visual features V. All modules within the dotted line box share weights. The output of the GRUs is used to classify each word as the answer or not a_{w_i}. The final GRU hidden states h_{title} and h_{sent} are put through fully connected layers to predict if the answer is in the sentence a_{sent} or title a_{title}, and then are combined together and used to classify if the answer is in the article a_{art}.

word among the retrieved sentences. More specifically, we take the highest value of $a_{w_i}.a_{sent}$, where a_{w_i} is the score for the word being the answer and a_{sent} is the score for the sentence including the answer.

MUTAN + AN

We augment MUTAN with the top sentence hidden states (h_{sent} in Figure 4.5) from ArticleNet (AN). During VQA training and testing, we take the top predicted sentences (ignoring duplicate sentences), and feed them in the memory of an end-to-end memory network [327]. The output of the memory network is concatenated with the output of the first MUTAN fusion layer.

BAN + AN

Similarly, we incorporate the ArticleNet hidden states into BAN and incorporate it into VQA pipeline with another memory network. We concatenate output of the memory network with the BAN hidden state right before the final classification network.

Method	OK-VQA	VT	BCP	OMC	SR	CF	GHLC	PEL	PA	ST	WC	Other
Q-Only	14.93	14.64	14.19	11.78	15.94	16.92	11.91	14.02	14.28	19.76	25.74	13.51
MLP	20.67	21.33	15.81	17.76	24.69	21.81	11.91	17.15	21.33	19.29	29.92	19.81
ArticleNet (AN)	5.28	4.48	0.93	5.09	5.11	5.69	6.24	3.13	6.95	5.00	9.92	5.33
BAN [166]	25.17	23.79	17.67	22.43	30.58	27.90	**25.96**	20.33	25.60	20.95	**40.16**	22.46
MUTAN [21]	26.41	25.36	18.95	24.02	33.23	27.73	17.59	20.09	**30.44**	20.48	39.38	22.46
BAN + AN	25.61	24.45	19.88	21.59	30.79	29.12	20.57	21.54	26.42	**27.14**	38.29	22.16
MUTAN + AN	**27.84**	**25.56**	**23.95**	**26.87**	**33.44**	**29.94**	20.71	**25.05**	29.70	24.76	39.84	**23.62**
BAN/AN oracle	27.59	26.35	18.26	24.35	33.12	30.46	28.51	21.54	28.79	24.52	41.4	25.07
MUTAN/AN oracle	28.47	27.28	19.53	25.28	35.13	30.53	21.56	21.68	32.16	24.76	41.4	24.85

Table 4.2: **Benchmark results on OK-VQA.** We show the results for the full OK-VQA dataset and for each knowledge category: Vehicles and Transportation (VT); Brands, Companies and Products (BCP); Objects, Material and Clothing (OMC); Sports and Recreation (SR); Cooking and Food (CF); Geography, History, Language and Culture (GHLC); People and Everyday Life (PEL); Plants and Animals (PA); Science and Technology (ST); Weather and Climate (WC); and Other.

MUTAN/AN oracle

As an upper bound check, and to see potentially how much VQA models could benefit from the knowledge retrieved using ArticleNet, we also provide results on an oracle, which simply takes the raw ArticleNet and MUTAN predictions, taking the best answer (comparing to ground truth) from either.

BAN/AN oracle

Similar to the MUTAN/AN oracle, but we take the best answer from the raw ArticleNet and BAN instead, again taking the best answer for each question.

4.5.2 Benchmark Results

We report the results using the common VQA evaluation metric [16], but use each of our answer annotations twice, since we have 5 answer annotations versus 10 in [16]. We also stem the answers using Porter stemming to consolidate answers that are identical except for pluralization and conjugation as in [357]. We also show the breakdowns for each of our knowledge categories. The results are reported in Table 4.2.

The first observation is that no method gets close to numbers on standard VQA dataset such as VQA [126] (where the best real open-ended result for the 2018 competition is 72.41). Moreover, models such as MUTAN [21] and BAN [166],

Method	VQA score on OK-VQA
ResNet152	26.41
ResNet50	24.74
ResNet18	23.64
Q-Only	14.93

Table 4.3: Results on OK-VQA with different visual features.

which are specifically designed for VQA to learn high-level associations between the image and question, get far worse numbers on our dataset. This suggests that OK-VQA cannot be solved simply by coming up with a clever model, but actually requires methods that incorporate information from outside the image.

It is interesting to note that although the performance of the raw ArticleNet is low, it provides improvement when combined with the other models (MUTAN + AN and BAN + AN). From the oracle numbers, we can see that the knowledge retrieved by ArticleNet provides complementary information. These oracles are optimistic upper bounds using ArticleNet, but they show that smarter knowledge-retrieval approaches could have stronger performance on our dataset. Note that ArticleNet is not directly trained on the OK-VQA task and can only predict answers within the articles it has retrieved. So the relatively low performance on OK-VQA is not surprising.

Looking at the category breakdowns, we see that ArticleNet is particularly helpful for brands, science, and cooking categories, perhaps suggesting that these categories are better represented in Wikipedia. It should be noted that the major portion of our dataset requires knowledge outside Wikipedia such as commonsense or visual knowledge.

The Q-Only baseline performs significantly worse than the other VQA baselines, suggesting that visual features are indeed necessary and our procedure for reducing answer bias was effective.

4.5.3 Visual Feature Ablation

We also want to demonstrate the difficulty of the dataset from the perspective of visual features, so we show MUTAN results using different ResNet architectures. The previously reported result for MUTAN is based on ResNet152. We also show the

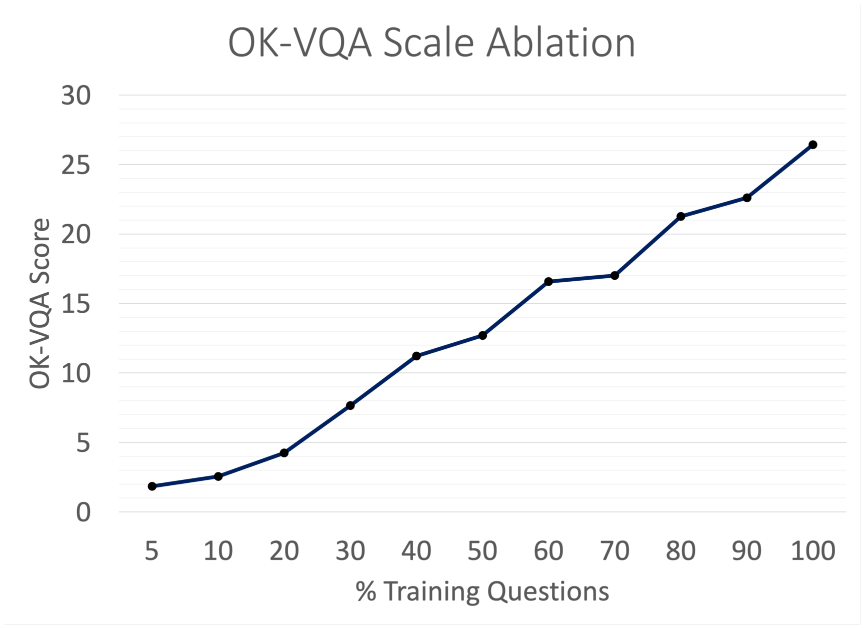


Figure 4.6: Results on OK-VQA using different sizes of the training set.

results using extracted features from ResNet50 and ResNet18 in Table 4.3. From this table it can be seen that going from ResNet50 to ResNet152 features only has a marginal improvement, and similarly going from ResNet18 to ResNet50. However, going from ResNet18 to no image (Q-Only) causes a large drop in performance. This suggests that our dataset is indeed visually grounded, but better image features do not hugely improve the results, suggesting the difficulty lies in the retrieving the relevant knowledge and reasoning required to answer the questions.

4.5.4 Scale Ablation

Finally, we investigate the degree to which the size of our dataset relates to its difficulty as opposed to the nature of the questions themselves. We first take a random subdivision of our training set and train MUTAN on progressively smaller subsets of the training data and evaluate on our original test set. Figure 4.6 shows the results.

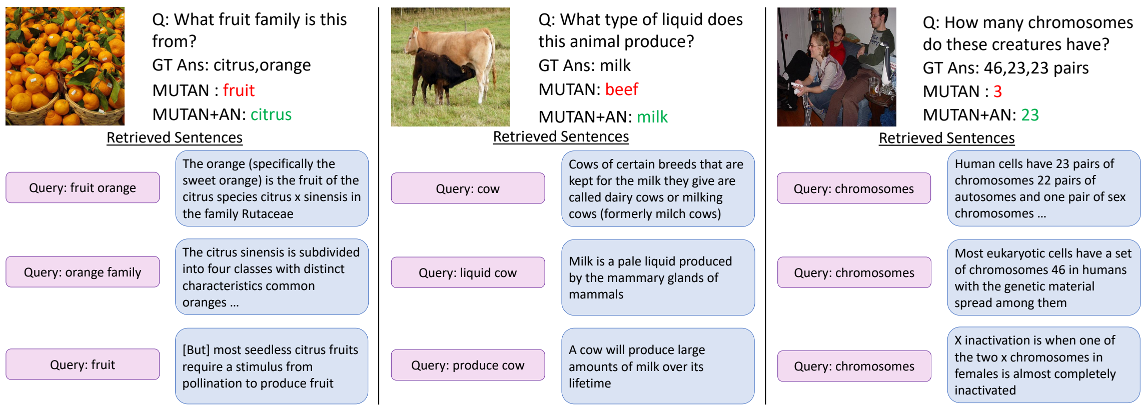

Q: What fruit family is this from?
GT Ans: citrus,orange
MUTAN : fruit
MUTAN+AN: citrus
Retrieved Sentences
Query: fruit orange
The orange (specifically the sweet orange) is the fruit of the citrus species citrus x sinensis in the family Rutaceae
Query: orange family
The citrus sinensis is subdivided into four classes with distinct characteristics common oranges …
Query: fruit
[But] most seedless citrus fruits require a stimulus from pollination to produce fruit
Q: What type of liquid does this animal produce?
GT Ans: milk
MUTAN: beef
MUTAN+AN: milk
Retrieved Sentences
Query: cow
Cows of certain breeds that are kept for the milk they give are called dairy cows or milking cows (formerly milch cows)
Query: liquid cow
Milk is a pale liquid produced by the mammary glands of mammals
Query: produce cow
A cow will produce large amounts of milk over its lifetime
Q: How many chromosomes do these creatures have?
GT Ans: 46,23,23 pairs
MUTAN : 3
MUTAN+AN: 23
Retrieved Sentences
Query: chromosomes
Human cells have 23 pairs of chromosomes 22 pairs of autosomes and one pair of sex chromosomes …
Query: chromosomes
Most eukaryotic cells have a set of chromosomes 46 in humans with the genetic material spread among them
Query: chromosomes
X inactivation is when one of the two x chromosomes in females is almost completely inactivated

Figure 4.7: **Qualitative results.** We show the result of MUTAN+AN compared to the MUTAN baseline answer and the ground truth answer ('GT Ans'). We show the query words that were used by ArticleNet (pink boxes) and the corresponding most relevant sentences (blue boxes).

4.5.5 Qualitative Examples

We show some qualitative examples in Figure 4.7 to see how outside knowledge helps VQA systems in a few examples. We compare MUTAN+AN method with MUTAN. The left example asks what "fruit family" the fruit in the image (oranges) comes from. We see that two sentences that directly contain the information that oranges are citrus fruits are retrieved —"The orange ... is a fruit of the citrus species" and "The citrus sinensis is subdivided into four classes [including] common oranges".

The middle example asks what liquid the animal (cow) produces. The first and third sentences tell us that cows produce milk, and the second sentence tells us that milk is a liquid. This gives the combined MUTAN+AN method enough information to correctly answer milk.

The example on the right asks how many chromosomes humans have. It is somewhat ambiguous whether it means how many individual chromosomes or how many pairs, so workers labeled both as answers. The retrieved articles are helpful here, retrieving two different articles referring to 23 pairs of chromosomes and 46 chromosomes total. The combined MUTAN+AN method correctly answers 23, while MUTAN guesses 3.

4.6 Conclusion

In this chapter, we address the task of knowledge-based visual question answering. We introduce a novel benchmark called OK-VQA for this task. Unlike the common VQA benchmarks, the information provided in the question and the corresponding images of OK-VQA is not sufficient to answer the questions, and answering the questions requires reasoning on external knowledge resources. We show that the performance of VQA models significantly drops on OK-VQA. We analyze the properties and statistics of the dataset and show that background knowledge can improve results on our dataset. Our experimental evaluations show that the proposed benchmark is quite challenging and that there is a large room for improvement.

In the next chapter, we will look at an approach we took to make progress on this dataset that incorporates different kinds of knowledge.

Chapter 5

Knowledge in Language and Vision: Methods

5.1 Introduction

In this chapter, we build directly on our dataset contribution from the previous chapter. Now that we have a better test of knowledge-aware vision and language systems, we introduce a method which combines both knowledge graphs (symbolic knowledge) and pretrained large language models (implicit knowledge).

Consider the example shown in Figure 5.1. To answer this question, we not only need to parse the question and understand the image but also use external knowledge. Early work in VQA focused on image and question parsing [3, 16, 115, 217, 219] assuming all required knowledge can be learned from the VQA training set. However, learning knowledge from image-question-answer triplets in the training data is not scalable and is liable to biases in the training data. We should exploit other external knowledge sources such as Wikipedia or knowledge graphs. In the last chapter, we introduced the OK-VQA dataset [224] which consists of these types of questions and allows us to study open-domain knowledge in VQA.

We have earlier defined two types of knowledge representation that can be useful for these types of questions: First we have implicit knowledge, knowledge which is embedded into some non-symbolic form such as the weights of a neural network

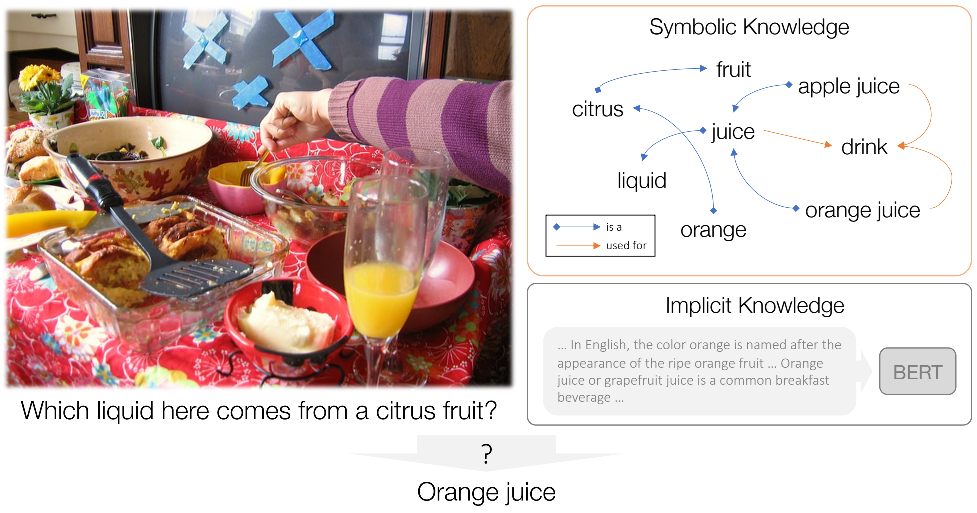

Symbolic Knowledge
fruit
apple juice
citrus
juice
drink
liquid
orange juice
orange
is a
used for
Implicit Knowledge
... In English, the color orange is named after the appearance of the ripe orange fruit ... Orange juice or grapefruit juice is a common breakfast beverage ...
BERT
Which liquid here comes from a citrus fruit?
?
Orange juice

Figure 5.1: An OK-VQA [224] example that requires external knowledge. Our KRISP model uses a symbolic knowledge graph as well as the implicit knowledge learned from large-scale BERT training to answer the question.

derived from annotated data or large-scale unsupervised language training. Recently, transformer- and specifically BERT- [90] based multi-modal VQA models have been proposed [195, 207, 208], which incorporate large scale language pretraining, implicitly capturing language based, as well as multimodal knowledge. This type of knowledge can be quite useful, but we find this form of implicitly learned knowledge is not sufficient to answer many knowledge-based questions as we will show. Perhaps this is not surprising if one considers that many facts are rare such as "Thomas Newcomen invented the steam engine" and learning them with implicit representations might be less efficient while there are external sources and knowledge bases that state it explicitly.

The other type of knowledge we look at here is explicit or symbolic knowledge in the form of knowledge graphs. Approaches that use this form of knowledge either take the symbolic knowledge and then embed-and-fuse them into a larger VQA model before answer prediction which no longer maintains the well-defined knowledge structures [194, 224], or by relying on a closed set of knowledge facts with strong annotation of source knowledge [249, 357, 365]. In the second case, the VQA dataset itself has ground truth "facts" associated with the question, so solving these questions often ends up being the problem of retrieving a fact from the closed set. In our method, we preserve the symbolic meaning of our knowledge from input until answer prediction. This allows us to use knowledge that is rare or is about rare entities as learning the reasoning logic with symbols is shared across all symbols. And unlike other work, we do not have a closed set or ground truth knowledge, so we must build a large diverse knowledge base for use by our model.

In this work, we develop an architecture, *KRISP (Knowledge Reasoning with Implicit and Symbolic rePresentations)*, to combine the implicit and symbolic knowledge. Specifically, KRISP uses (i) a multi-modal BERT-pretrained transformer to process the question and image, and take advantage of the implicit knowledge in BERT, and (ii) a graph network to make use of symbolic knowledge bases. To cover the wide variety of knowledge required in OK-VQA, we draw on four very different knowledge sources to construct our knowledge graph: DBPedia [18], ConceptNet [200], Visual Genome [179] and hasPart KB [24]. This covers crowdsourced data, visual data, encyclopedic data, knowledge about everyday objects, knowledge about science and knowledge about specific people, places and events. Finally, our method preserves the symbolic meaning of the knowledge by making predictions based on the hidden state of individual nodes in the knowledge graph and using a late-fusion strategy to combine the implicit and symbolic parts of the model.

5.2 Related Work

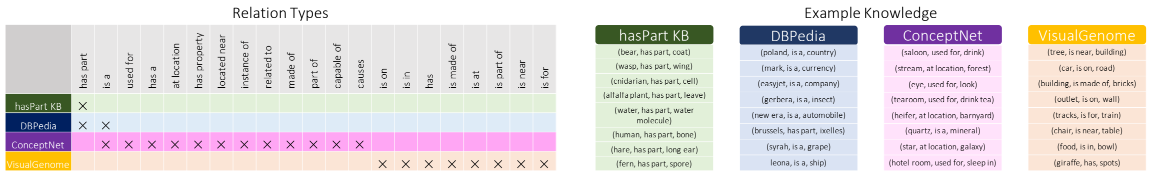
Multimodal Vision and Language Modeling. Approaches for multimodal vision and language tasks have explored diverse set of fusion strategies such as bilinear models (e.g. [115, 166]) or self-attention (e.g. [118]). Many recent works have been inspired by the success of transformer [347] and BERT [90] models for natural language tasks and proposed transformer-based fusion between image and text [5, 68, 192, 195, 207, 326, 334, 399]. Similar to these works as part of our method we train a multimodal transformer with BERT-pretraining to import the implicit knowledge learned by BERT and learn any knowledge encoded in the training data and study it on knowledge VQA.

Another line of work has been extracting programs from the question for explicit reasoning with modules [12] or extracting symbols from the image to reason over them [386]. These works focus on reasoning about things explicitly in the image but do not integrate external knowledge.

Knowledge-based VQA datasets. While open-ended VQA datasets (e.g. [16]) might require outside knowledge to answer some of its questions which cannot be learned from the dataset, there are a few datasets which focus specifically on knowledge based multi-modal reasoning. One is FVQA [357], where image-questions-answer triples are annotated with a fact-triple (e.g. "chair is furniture") from a fixed outside knowledge base, which allows deriving the answer. Specifically one of the two nodes (i.e. chair or furniture in this example) is the answer. In Chapter 4 we introduced OK-VQA [224] which stands for *Outside Knowledge VQA*, as the name suggests, focusing on knowledge which is not tied to a specific knowledge base. We focus our evaluation on OK-VQA due to its relatively large number of knowledge-based questions, as well as its challenging and open-ended nature.

Symbolic Knowledge for VQA. Symbolic knowledge from knowledge bases is commonly represented as graphs/knowledge bases [194, 248, 249, 356, 357] or textual knowledge sources such as Wikipedia [224, 365]. We can separate these into two directions: where symbols are retained until prediction and where they are not. [249, 356, 357] retain the symbols until the answers, allowing good generalization capabilities but require annotations of the "correct" knowledge fact and are difficult to generalize to open knowledge VQA. For improved generalization to open-domain VQA, [120, 194, 224, 365] embed the symbolic knowledge to an implicit embedding loosing the semantics of the symbols, but therefore are able to easily integrate the embedding with standard VQA approaches. Similar to our work, the recent work [120] relies on a multimodal transformer model (pretrained VilBERT [207]), however, similar to the other works it looses the semantics of the knowledge symbols when it integrates over them with an attention model. In contrast, our work shows how to take advantage of both the implicit and symbolic knowledge directions: We retain symbols until the end without the need of knowledge-fact annotations and integrate it with implicit knowledge and powerful reasoning abilities of multi-modal transformers.

Knowledge Bases & Knowledge in NLP. There have been many knowledge bases proposed for knowledge-based reasoning, both language-only and multi-modal [24, 65, 94, 179, 235, 295, 401, 402, 404]. In the NLP literature, there has been much work in question answering from knowledge sources [23, 34, 382] including for open-domain question answering [61, 358, 378, 379], and including mixed symbolic/implicit



Relation Types

	has part	is a	used for	has a	at location	has property	located near	instance of	related to	made of	part of	capable of	causes	is on	is in	has	is made of	is at	is part of	is near	is for
hasPart KB	×																				
DBPedia	×	×																			
ConceptNet		×	×	×	×	×	×	×	×	×	×	×	×								
VisualGenome														×	×	×	×	×	×	×	×

Example Knowledge

hasPart KB	DBPedia	ConceptNet	VisualGenome
(bear, has part, coat)	(poland, is a, country)	(saloon, used for, drink)	(tree, is near, building)
(wasp, has part, wing)	(mark, is a, currency)	(stream, at location, forest)	(car, is on, road)
(cnidarian, has part, cell)	(easyjet, is a, company)	(eye, used for, look)	(building, is made of, bricks)
(alfalfa plant, has part, leave)	(gerbera, is a, insect)	(tearoom, used for, drink tea)	(outlet, is on, wall)
(water, has part, water molecule)	(new era, is a, automobile)	(heifer, at location, barnyard)	(tracks, is for, train)
(human, has part, bone)	(brussels, has part, ixelles)	(quartz, is a, mineral)	(chair, is near, table)
(hare, has part, long ear)	(syrah, is a, grape)	(star, at location, galaxy)	(food, is in, bowl)
(fern, has part, spore)	leona, is a, ship)	(hotel room, used for, sleep in)	(giraffe, has, spots)

Figure 5.2: Example knowledge and edge types from our knowledge graph. The graph is built from four sources of explicit knowledge.

methods for question answering [164, 212].

5.3 Methodology

In this section we introduce our model: *Knowledge Reasoning with Implicit and Symbolic rePresentations* (KRISP). An overview of our model can be seen in Figure 5.3. We first introduce our transformer-based multi-modal implicit knowledge reasoning (Section 5.3.1), then discuss the symbolic knowledge sources and reasoning with symbols (Section 5.3.2), and then describe their integration in Section 5.3.3.

5.3.1 Reasoning with Implicit Knowledge

We want to incorporate implicit external knowledge as well as multi-modal knowledge which can be learned from training set in our model. Language models, and especially transformer-based language models, have shown to contain common sense and factual knowledge [153, 269] [1]. Most recent multi-modal models have also relied on the transformer architecture to learn vision-and-language alignment [195, 207]. We adopt this direction in our work and build a multi-modal transformer model, pretrained with BERT [90], which has been pretrained on the following language corpora to capture implicit knowledge: BookCorpus [405] (800M words) and English Wikipedia [1] (2.5B words). To learn multi-modal knowledge from the training set, our model is most closely related to the architecture used in [195]. We also explore multi-modal pretraining in Section 5.4.2.

[1]For a further discussion of implicit knowledge from and large language models generally, see Chapter 2.3.2.

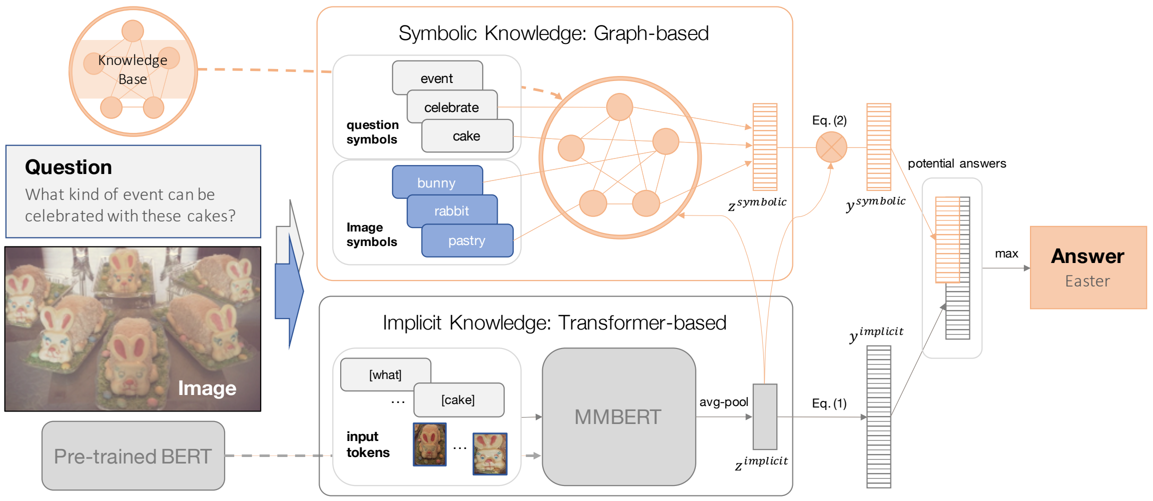


Figure 5.3: Our model: KRISP integrates implicit knowledge and reasoning (bottom) with explicit graph-based reasoning on a knowledge base (top). The implicit knowledge model receives the visual features and question encoding whereas the explicit knowledge model operates on image and question symbols. They predict answers according to Eq. 5.1&5.2 and we take the max overall prediction (see Section 5.3.3).

Question Encoding

We tokenize a question Q using WordPiece [366] as in BERT [90], giving us a sequence of $|Q|$ tokens and embed them with the pretrained BERT embeddings and append BERT's positional encoding, giving us a sequence of d-dimensional token representation $x_1^Q, ..., x_{|Q|}^Q$. We feed these into the transformer, finetuning the representation during training.

Visual Features

As with most VQA systems, we use visual features extracted on the dataset by a visual recognition system trained on other tasks. We use bottom-up features [10] collected from the classification head of a detection model, specifically Faster R-CNN [287]. Because of the overlap in OK-VQA test and Visual Genome/COCO [198] trainval, we trained our detection model from scratch on Visual Genome, using a new split of Visual Genome not containing OK-VQA test images. The detector uses feature pyramid networks [199], and is trained using the hyper-parameters used for the baselines in [151].

We input bounding box features extracted from the image as well as the question words to the transformer. We mean-pool the output of all transformer steps to get our combined implicit knowledge representation $z^{implicit}$.

5.3.2 Reasoning with Symbolic Knowledge

Visual Symbols

In addition to using a pretrained visual recognition system to get image features, we also extract visual concepts (i.e. the predictions). This not only allows us to get a set of concepts to use to prune our knowledge graph (see Section 5.3.2), it also gives us an entry point to get from the raw image to a set of symbols. This is significant—in order for our graph network to be able to reason about the question, it not only needs to reason about the question itself, but the entities in the image. For instance, if a question were to ask "what is a female one of these called?" in order use our knowledge that a female sheep is called an "ewe," the graph network needs to actually know that the thing in the picture is a sheep. As we will see, using these symbols is critical for our graph network to reason about the question.

There are a number of visual concepts we want to cover: places, objects, parts of objects and attributes. Therefore we run four classifiers and detectors trained on images from the following datasets: ImageNet [294] for objects, Places365 [397] for places, LVIS [131] for objects and object parts and Visual Genome [179] for objects, parts and attributes. This gives us a total of about 4000 visual concepts.

Knowledge Graph Construction

Unlike previous work such as [249], or in NLP work on datasets such as SQuAD [281] which study the problem of closed-system knowledge retrieval, we do not have a ground truth set of facts or knowledge which can be used to answer the question. We must make an additional choice of what knowledge sources to use and how to clean or filter them[2].

There are a few different kinds of knowledge that might help us on this task. One is what one might call trivia knowledge: facts about famous people, places or events.

[2]For a more complete set of knowledge sources and knowledge graphs used in AI, see Chapter 2.4

Another is commonsense knowledge: what are houses made of, what is a wheel part of. Another is scientific knowledge: what genus are dogs, what are different kinds of nutrients. Finally, situational knowledge: where do cars tend to be located, what tends to be inside bowls.

The first and largest source of knowledge we use is DBPedia [18], containing millions of knowledge triplets in its raw form. DBPedia is created automatically from data from Wikipedia [1]. This tends to give a lot of categorical information e.g. (Denmark, is_a, country), especially about proper nouns such as places, people, companies, films etc. The second source of knowledge is ConceptNet [200], a crowd-sourced project containing over 100,000 facts organized as knowledge triples collected by translating English-language facts into an organized triplet structure. It also contains as a subset the WordNet [235] ontology. This dataset contains commonsense knowledge about the world such as (dog, has_property, friendly). As we did in Chapter 3, we also use the scene graphs from Visual Genome [179] as another source of knowledge. As we did there, we take a split of Visual Genome that does not contain any OK-VQA test images. This knowledge source tends to give us more spatial relationships e.g. (boat, is_on, water) and common pairwise affordances e.g. (person, sits_on, coach). Finally, we use the new hasPart KB [24] to get part relationships between common objects such as (dog, has_part, whiskers) as well as scientific ones (molecules, has_part, atoms)[3]. We show example knowledge triplets from our in Figure 5.2.

With these knowledge sources, we can capture a large amount of knowledge about the world. But we then run into a problem of scale. In its raw form, DBPedia alone contains millions of edges, with the others containing a total of over 200,000 knowledge triplets. This first presents a technical problem—this graph is far too large to fit into GPU memory if we use a graph neural network model. But more fundamentally, while this knowledge graph contains a lot of useful information for our downstream task, it also includes a lot of irrelevant knowledge. In particular, DBPedia, being parsed automatically from Wikipedia pages, contains information about virtually every film, book, song and notable human in history. While some of those may be useful for particular questions, the vast majority is not.

To deal with these issues, we limit our knowledge graph to entities that are likely

[3]See Chapter 2.4.1 for more details on these and other knowledge graphs

to be helpful for our end task. First, we collect all of the symbolic entities from the dataset: in particular the question, answers and visual concepts that can be picked up by visual recognition systems (see Section 5.3.2). We then include edges that only include these concepts. After this filtering, we have a total of about 36,000 edges and 8,000 nodes.

Graph Network

Now we move to our symbolic knowledge representation. We want to treat our knowledge graph as input without having to decide on which few facts out of our entire graph might be relevant. So to process on our entire graph and decide this during training, we use a graph neural network to incorporate our knowledge [4]. In our network, each node of the graph network corresponds to one specific symbol representing one concept such as "dog" or "human" in our knowledge graph.

The idea is that the graph neural network can take in information about each specific symbol and use the knowledge edges to infer information about other symbols by passing information along the edges in the knowledge graph. And, in our graph neural network we share the network parameters across all symbols, meaning that unlike for other types of networks, the reasoning logic is shared across all symbols which should allow it to generalize better to rare symbols or graph edges.

We use the Relational Graph Convolutional Network (RGCN) [301] as the base graph network for our model. Unlike the related GCN [170], this model natively supports having different calculations between nodes for different edge types (an is_a relationship is treated differently than a has_a relationship) and edge directions (dog is_a animal is different than animal is_a dog). With this architecture we also avoid the large asymptotic runtime of other architectures with these properties such as [196] or [348].

Graph Inputs

For one particular question image pair, each node in the graph network receives 4 inputs.

[4]See Chapter 2.3.3 for a thorough discussion, explanation and mathematical formulation of graph neural networks.

1. An indicator 0/1 of whether the concept appears in the question.
2. The classifier probabilities for the node's concept, introduced above (or 0 if the concept is not detected in the particular image or not one of the classifier's concepts). With 4 image classifiers or detectors, the node receives 4 separate numbers.
3. The $300d$ word embedding (GloVe [266]) representation of that concept, or average embedding for multi-word concepts.
4. The implicit knowledge representation $z^{implicit}$ from Section 5.3.1 passed through a fully connected layer: $fc(z^{implicit})$ with ReLU activation to reduce the size of this feature to 128 for efficient graph computation.

Following the standard formulation of graph neural networks, we write the input to the graph neural networks (described above) as $X{=}H^{(0)}$ where X is a $\mathbb{R}^{n \times d_s}$ matrix with n node inputs of size $d_s = 433$. Then for each layer of the RGCN, we have a non-linear function $H^{(l+1)}{=}f(H^{(l)}, KG)$ where KG is the knowledge graph. The RGCN convolution uses different weight matrices for different edge types and for different directions. As a result the semantic difference between an is-a relationship and a has-a relationship as well as the direction of those edges is captured in the structure of the network and different transformations are learned for each. After all RGCN layers are computed we end up with $H^{(L)}{=}G$ which is a $\mathbb{R}^{n \times d_h}$ matrix which corresponds to having a hidden state of size f_h for each node (and therefore concept) in our graph.

5.3.3 Integrating Implicit and Symbolic Knowledge

Finally, given the output of our implicit transformer-based module $z^{implicit}$ and our explicit/symbolic module G, how do we get our final prediction? Our main insight to make a separate prediction for $z^{implicit}$ and for each node/concept in the knowledge graph.

Implicit Answer Prediction

As is now commonplace among VQA methods, to get the implicit answer prediction, we do a final prediction layer and predict the answer within a set vocabulary of answers $V \in \mathbb{R}^a$ where a is the size of the answer vocabulary. We simply have:

$$y^{implicit} = \sigma(W z^{implicit} + b) \tag{5.1}$$

where σ is the sigmoid activation.

Symbolic Answer Prediction

To predict the answers for symbolic, we note that G can be rewritten as a hidden state node $z_i^{symbolic}$ for each node/concept i in the knowledge graph. Because each of these nodes corresponds to a word or multi-word symbol, we actually have nodes and corresponding hidden states that are possible answers to a VQA question. So for each hidden state that is in our answer vocab $V \in \mathbb{R}^a$ we make a prediction for it.

For each of these answer nodes i, we predict:

$$y_i^{symbolic} = \sigma((W^s z_i^{symbolic} + b^s)^T (W^z z^{implicit} + b^z)) \tag{5.2}$$

We additionally re-use the implicit hidden state $z^{implicit}$ to make this prediction. This gives us an additional late fusion between the implicit and symbolic parts of our model.

Final Prediction

Finally, given our final predictions $y^{implicit}$ and $y^{symbolic}$, we simply choose the final answer by choosing the highest scoring answer from both answer vectors. For training, we can simply optimize $y^{implicit}$ and $y^{symbolic}$ separately with a binary cross entropy loss end-to-end through the entire network. See Figure 5.3.

5.4 Experiments

5.4.1 Experimental Setup

For all experiments, we train our models with PyTorch [264] and the MMF Multimodal Framework [315]. We use PyTorch Geometric [108] for our graph neural network implementations. For consistency, for each result we train each model on 3 random seeds and take the average as the result.

For the purpose of state-of-the art comparisons in Table 5.1, we compare our main method on the 1.0 version of OK-VQA [224]. This was later updated to a 1.1 version, and all other experiments including ablations are done on this version. The only change between the versions is a change in how answer stemming is handled, resulting in a more coherent answer vocabulary. In particular, we observe that the new answer vocabulary has much fewer "non-word" stemming such as "buse" for busses and "poni tail" instead of "pony tail." Unless otherwise stated, an experiment is on version 1.1.

For many of our ablations and analysis we train just the Multi-modal BERT (MMBERT) model described in Section 5.3.1 by itself by scratch or we do multi-modal pre-training. Unless otherwise stated, this model and ours is always initialized from BERT.

In Section 5.4.3 we do a through ablation of KRISP comparing the different parts of the model and design choices we made. In Section 5.4.2 we add multimodal pretraining to our models to show how our model achieves state-of-the-art performance on OK-VQA. In Section 5.4.4 we show the results of a number of experiments to more thoroughly analyze our method, especially looking at its performance on rare answers. Finally in Section 5.4.5 we look at some specific questions and predictions from our model to get a more grounded idea of what our model does on real examples.

5.4.2 State-of-the-Art Comparisons

We provide the comparisons to the state-of-the-art of OK-VQA in Table 5.1. To achieve best results, like other works [120] we pretrain our network on other tasks. We find it the most effective to pretrain our models on the VQA dataset [126].

In order to compare to other works (all of which show results on v1.0), we compute the performance of on OK-VQA v1.0 as well. We see that our model achieves 38.35% accuracy versus the best previous state-state-of-the-art of 33.66% [120]. We also compare on v1.1 as well, re-running the MUTAN+AN model from [224] to get a comparison with KRISP.

Method	accuracy (v1.0)	accuracy (v1.1)
Q-Only	14.93	-
MLP	20.67	-
BAN [166]	25.17	-
BAN+AN [224]	25.61	-
BAN+KG-Aug [194]	26.71	-
MUTAN [21]	26.41	-
MUTAN+AN [224]	27.84	26.64
ConceptBERT [120]	33.66	-
KRISP (w/o mm pre.)	29.77	32.31
KRISP (with mm pre.)	**38.35**	**38.90**

Table 5.1: Benchmark results on OK-VQA

5.4.3 Model Analysis and Ablations

We first analyse our model to see where the improvement is coming from with several ablations, especially focusing on symbolic vs implicit knowledge and their integration. We want to understand which parts are working and why.

Ablation of Symbolic Knowledge

First, we see how much of the improvement comes from the Multi-modal BERT backbone of our model versus from the symbolic Graph Network. In Table 5.2 (lines 1&2), we see that KRISP combining implicit and symbolic knowledge improves significantly over the Multi-modal BERT by about 3%.

We should, however, make sure this improvement is due to the symbolic knowledge and not merely from a more complex or better architecture. While our KRISP only has slightly more parameters (116M parameters versus MMBERT with 113M), it does add at least some extra computation. To test this, we approximate a version of our method with only the architecture and not the underlying knowledge. To do this, we keep all network details the same, but instead of using the knowledge graph we constructed in Section 5.3.2, we use a randomly connected graph. We keep all of the nodes the same, but we randomize the edges connecting them. So in this version with a random graph, our graph network receives all of the same inputs and the outputs,

	Method	accuracy
1.	KRISP (ours)	**32.31**
	Ablation of Symbolic Knowledge	
2.	MMBERT	29.26
3.	KRISP w/ random graph	30.15
	Ablation of Implicit Knowledge	
4.	KRISP w/o BERT pretrain	26.28
5.	MMBERT w/o BERT pretrain	21.82
	Ablation of Network Architecture	
6.	KRISP no late fusion	31.10
7.	KRISP no MMBERT input	31.10
8.	KRISP no MMBERT input or late fusion	25.00
9.	KRISP no backprop into MMBERT	27.98
10.	KRISP with GCN	30.58
11.	KRISP feed graph into MMBERT	30.99
	Ablation of Graph Inputs	
12.	KRISP no Q to graph	31.74
13.	KRISP no I to graph	31.59
14.	KRISP no symbol input	30.26
15.	KRISP no w2v	31.95

Table 5.2: KRISP ablation on OK-VQA v1.1. We show the performance of our model compared with the implicit-only baseline (MMBERT). We also show ablations without BERT training, with a random knowledge graph, ablations on our model architecture, and ablations where we remove the question input to the graph network (no Q), the image inputs (no I) and both (no symbol).

but all connections are completely random. If the performance were just from the computation, we would expect this to work. Instead, we see from line 3 that the performance using the random graph drops significantly.

Ablation of Implicit Knowledge

Next we look at the implicit knowledge contained in the BERT versus our combined system to see how much of an effect it had. From Table 5.2 we can see that BERT is a crucial element. Without the BERT pretraining (lines 4&5), our method falls by 6% and the Multi-modal BERT falls by an even larger 7%. This shows that the implicit knowledge is an important component of our model. The difference between KRISP and Multi-modal BERT when neither has BERT pretraining is actually higher than the difference with BERT, about 4.5%, suggesting that there is some overlap in the knowledge contained in our knowledge graphs with the implicit knowledge in BERT, but most of that knowledge is non-overlapping.

Ablation of Network Architecture

Next, we want to get a sense of which parts of our architecture were important. As we can see, our particular architecture is critical: the use of MMBERT features as input to KRISP and the late fusion were both important. With just one of these, performance drops by about 1%, but without either (line 8), performance drops over 7%. Without at least one connection between the Multi-modal BERT and the graph network, there can be no fusion of the visual features and question and the graph network cannot incorporate any of the implicit knowledge in BERT. We also tried KRISP where these two ways of fusing were present, but we did not allow any backpropogation from the Graph Network to MMBERT (line 9). This also performs badly, as the graph network cannot correct errors coming from this input, but not as bad as removing these connections entirely (line 8).

We also tried a less powerful graph network: GCN [170] (line 10) which critically does not have directed edges or edge types. This baseline hurts performance by about 2% justifying our choice of a graph network that uses edge direction and type. We also have another architectural ablation, where we feed the graph network features directly to the Multi-modal BERT rather than having a separate answer prediction directly from the graph as in KRISP or any of the other baselines (line 11). This architecture performs much worse than our final model.

Ablation of Graph Inputs

Next we look at the symbolic and non-symbolic inputs to the knowledge graph nodes to see what effect those might have had in the next section of Table 5.2. First, we ablate the question indicator input (line 12) and the image confidences (line 13) described in Section 5.3.2. We find that removing one or the other drops performance, but not drastically; removing both (line 14) drops performance by about 2%, much more than the effect of dropping the MMBERT input to the graph. We also ablate the

	Method	accuracy
1.	KRISP max($y^{implicit}, y^{symbolic}$) (ours)	**32.31**
2.	KRISP $y^{implicit}$	31.47
3.	KRISP $y^{symbolic}$	29.36
4.	KRISP no backprop $y^{implicit}$	28.19
5.	KRISP oracle($y^{implicit} \| y^{symbolic}$)	36.71

Table 5.3: KRISP Subpart Analysis on OK-VQA v1.1. Here we show the OK-VQA accuracy of different parts of the model separately: just the MMBERT ($y^{implicit}$), just the graph network ($y^{symbolic}$). We also show the MMBERT only without a backpropogation signal between the two parts and an oracle best-case performance between the two parts.

word2vec inputs to nodes (line 15) and find that this part made the least difference, dropping it less than 1%.

Preserving Symbolic Meaning

One major claim we make is that symbolic and implicit knowledge are both necessary for this problem. The results without BERT training make the case pretty clearly that implicit, non-symbolic knowledge from BERT is critical. From the ablation of symbolic knowledge, we show that it is the symbolic knowledge (and not just the architecture) greatly contributes to the performance of our method. On the symbol input side, we show that removing the symbolic inputs (line 12) hurts performance, even more than removing the Multi-modal BERT hidden input (line 7) which contains information about the same image and question, but in a non-symbolic form. Finally we have a baseline (line 11) where instead of predicting separate outputs from the graph network and Multi-modal BERT, we directly connect the graph network into MMBERT, feeding a pooled graph hidden state into MMBERT as an input. This baseline does significantly worse. What these ablations have in common is that they remove the direct connection between the knowledge graph and the input and/or answer symbols. When the graph network is not able to connect the knowledge symbolically to the input symbols or the output symbols, we see that it performs worse. In addition, we know symbolic knowledge itself is useful because when we only change the connections between nodes and nothing else (line 3), performance drops drastically. Our entire graph module directly connects symbols in the input (question words and image symbols from classifiers) to symbols in the output (the answer words) and this seems critical to performance.

5.4.4 Quantitative Analysis

First we examine the parts of our model separately to see if we can learn anything about how the MMBERT and Graph Network parts of KRISP interact.

In Table 5.3 we look at the performance of different parts of our model (without retraining the model for lines 1,2,3,5). Since the MMBERT and Graph Network parts of KRISP produce separate predictions, we can analyze them separately. For instance, we find that despite the fact that the MMBERT part of our model does not receive input from the Graph Network, the MMBERT (Table 5.3, line 2) has a higher accuracy of 31.47% than the MMBERT baseline (Table 5.2, line 2), 29.26%. This we suspect is because this part of the network receives a back-propagation from the Graph Network part of the model and this extra component improves the quality of the MMBERT pooled feature because it is also trained to reduce the loss from the late fusion predictions. Indeed, if we remove the back-propagation signal (Table 5.3, line 4) we see that the accuracy of this part of the model drops down to 28.19%. We also see a direct improvement beyond this effect. Comparing the Multi-modal BERT (line 2) and Graph Network (line 3) -only accuracies, the Graph Network does a bit worse on its own, but not by a huge amount, and the Graph Network predictions are used 47% of the time in the joint model (line 1). Since the accuracy of the combined model is higher than each, it is able to choose the correct answer from between MMBERT and Graph Network. Finally, we see that if we had an oracle that always chose the best prediction from either the MMBERT or the Graph Network, we would improve the accuracy to 36.71%. Obviously this is not a realistic number to achieve since it uses ground truth, but it shows that the MMBERT and Graph Network predictions are non-redundant.

Long-Tail Analysis

Next, we try to see whether our explicit/implicit model performs any differently on the “long tail” of OK-VQA. OK-VQA itself is built as a long-tail dataset, specifically

Metric→ Method ↓	Frequency Rank All	Frequency Rank Correct	# Unique answers All	# Unique answers Correct
KRISP (ours)	**528.5**	**456.7**	**1349**	**780**
MMBERT	467.1	427.4	1247	719

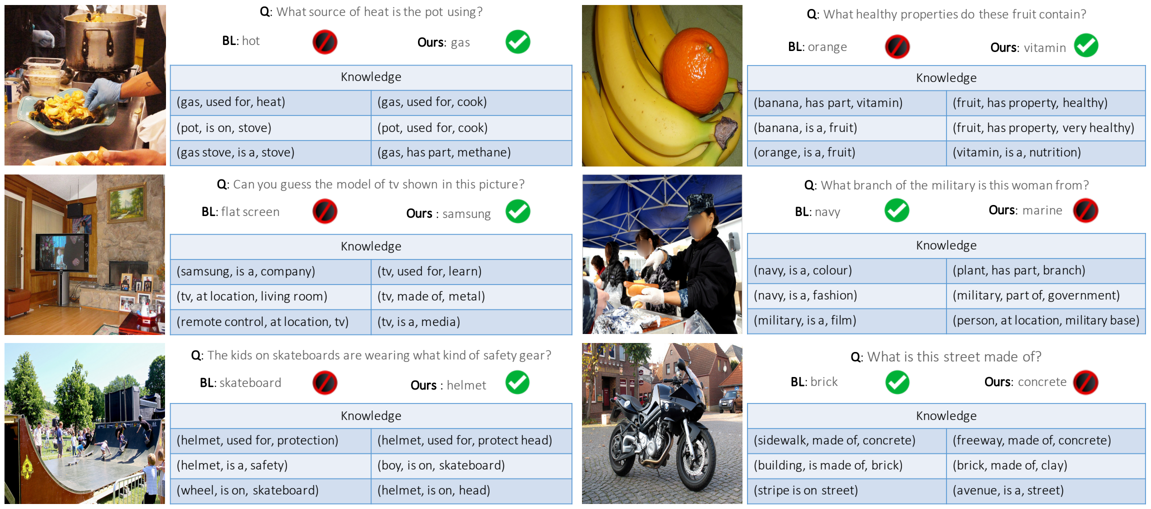
Table 5.4: Long-tail Analysis. We show KRISP and the non-symbolic MMBERT long-tail metrics for "all" predictions made by the model and for "correct" predictions. Higher is better.

rejecting answers that appear too many times to avoid models overfitting to the answer vocabulary, making it a good dataset to study knowledge-based VQA. Even with this filtering, some answers do appear more often than others, so we can try to study whether our method does better on rare answers.

In Table 5.4 we show metrics on KRISP versus the baseline Multi-modal BERT. First we use a metric we refer to as "Answer Frequency Rank". This simply means we order the answers in the dataset from most common to least common and assign them a rank from 1 for the most common to the total number of answers in the dataset. On this metric our model scores higher, which means it chooses on average less common answers. This is true whether one measures for all prediction or for only correct predictions. For a perhaps more intuitive metric we also look at the number of unique answers our model predicts versus the baseline. Here we predict 1349 versus 1247 or 780 versus 719 if we only look at correct predictions. These results indicate that our model is generalizing better to the long-tail.

5.4.5 Qualitative Analysis

Finally, we show examples to understand how the knowledge graph might be helping our model to answer questions. In the top left example in Figure 5.4 our model correctly answers that the source of heat for the pot is "gas." Looking at the knowledge graph, some knowledge that may be helpful is that gas is used for heat, and that both gas and pot are used to cook. The knowledge graph here connects directly from a word in the question to the answer. The next question asks what model the TV is and our model predicts Samsung. This is supported by an edge that indicates that Samsung is a company which makes it more likely to be a "model" of a product.



Q: What source of heat is the pot using?

BL: hot ✗ **Ours**: gas ✓

Knowledge	
(gas, used for, heat)	(gas, used for, cook)
(pot, is on, stove)	(pot, used for, cook)
(gas stove, is a, stove)	(gas, has part, methane)

Q: What healthy properties do these fruit contain?

BL: orange ✗ **Ours**: vitamin ✓

Knowledge	
(banana, has part, vitamin)	(fruit, has property, healthy)
(banana, is a, fruit)	(fruit, has property, very healthy)
(orange, is a, fruit)	(vitamin, is a, nutrition)

Q: Can you guess the model of tv shown in this picture?

BL: flat screen ✗ **Ours** : samsung ✓

Knowledge	
(samsung, is a, company)	(tv, used for, learn)
(tv, at location, living room)	(tv, made of, metal)
(remote control, at location, tv)	(tv, is a, media)

Q: What branch of the military is this woman from?

BL: navy ✓ **Ours**: marine ✗

Knowledge	
(navy, is a, colour)	(plant, has part, branch)
(navy, is a, fashion)	(military, part of, government)
(military, is a, film)	(person, at location, military base)

Q: The kids on skateboards are wearing what kind of safety gear?

BL: skateboard ✗ **Ours** : helmet ✓

Knowledge	
(helmet, used for, protection)	(helmet, used for, protect head)
(helmet, is a, safety)	(boy, is on, skateboard)
(wheel, is on, skateboard)	(helmet, is on, head)

Q: What is this street made of?

BL: brick ✓ **Ours**: concrete ✗

Knowledge	
(sidewalk, made of, concrete)	(freeway, made of, concrete)
(building, is made of, brick)	(brick, made of, clay)
(stripe is on street)	(avenue, is a, street)

Figure 5.4: Qualitative examples from KRISP. Showing predictions by our model and the implicit knowledge baseline Multi-modal BERT. We show the question, image, and answers given by both models. We also show knowledge in the graph related to the question, answers or image that seemed most relevant.

5.5 Conclusion

In this work we introduce *Knowledge Reasoning with Implicit and Symbolic rePresentations* (KRISP): a method for incorporating implicit and symbolic knowledge into Knowledge-Based VQA. We show it outperforms prior works on OK-VQA [224], the largest available open-domain knowledge VQA dataset. We show through extensive ablations that our particular architecture outperforms baselines and other alternatives by preserving the symbolic representations from input to prediction. Moreover, through experiments, analysis, and examples we find our model makes use of both implicit and symbolic knowledge to answer knowledge-based questions and generalizes to rare answers.

In the remaining chapters, we will shift our focus towards knowledge in embodied agents: those which take actions. Despite the change in setting, many of the methods and ideas from these first three contributions will remain relevant there.

Chapter 6

Knowledge in Action: RL

6.1 Introduction

In this chapter, we take our first look at systems which take actions. In our three-legged stool, we examine the role of knowledge for connecting the action and language legs. By exploiting the prior knowledge contained in language and word vectors, we see in this work that we are able to solve a complex multi-task crafting environment.

One of the most remarkable aspects of human intelligence is the ability to quickly adapt to new tasks and environments. From a young age, children are able to acquire new skills and solve new tasks through imitation and instruction [76, 145, 230]. The key is our ability to use language to learn abstract concepts and then reapply them in new settings. Inspired by this, one of the long term goals in AI is to build agents that can learn to accomplish new tasks and goals in an open-world setting using just a few examples or few instructions from humans. For example, if we had a health-care assistant robot, we might want to teach it how to bring us our favorite drink or make us a meal in just the way we like it, perhaps by showing it how to do this a few times and explaining the steps involved. However, the ability to adapt to new environments and tasks remains a distant dream.

Previous work have considered using language as a high-level representation for RL [13, 152]. However, these approaches typically use language generated from templates that are hard-coded into the simulators the agents are tested in, allowing the agents to receive virtually unlimited training data to learn language abstractions. But both

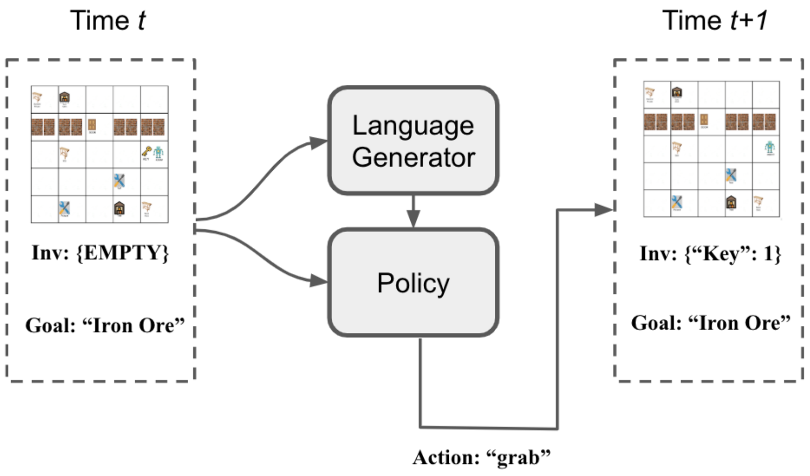


Figure 6.1: From state observation at time step t, the agent generates a natural language instruction "go to key and press grab," which guides the agent to grab the key. After the instruction is fulfilled and the agent grabs the key, the agent generates a new instruction at $t + 1$.

ideally and practically, instructions are a limited resource. If we want to build agents that can quickly adapt in open-world settings, they need to be able to learn from limited, real instruction data [211]. And unlike the clean ontologies generated in these previous approaches, human language is noisy and diverse; there are many ways to say the same thing. Approaches that aim to learn new tasks from humans must be able to use human-generated instructions.

In this work, we take a step towards agents that can learn from limited human instruction and demonstration by collecting a new dataset with natural language annotated tasks and corresponding game-play. The environment and dataset is designed to directly test multi-task and sub-task learning, as it consists of nearly 50 diverse crafting tasks. Crafts are designed to share similar features and sub-steps so we would be able to test whether the method is able to learn these shared features and reuse existing knowledge to solve new, but related tasks more efficiently. Our dataset is collected in a crafting-based environment and contains over $6,000$ game traces on 14 unique crafting tasks which serve as the training set. The other 35 crafting tasks will act as zero-shot tasks. The goal is for an agent to be able to learn one policy that is able to solve both tasks it was trained on as well as a variety of unseen tasks which contain similar sub-tasks as the training tasks.

To do this, we train a neural network to generate natural language instructions as a high-level representation of the sub-task, and then a policy to achieve the goal condition given these instructions. Figure 6.1 shows how our agent takes in the given state of the environment and a goal (Iron Ore), generates a language representation of the next instruction, and then uses the policy to select an action conditioned on the language representation - in this case to grab the key. We incorporate both imitation learning (IL) using both the language and human demonstrations and reinforcement learning (RL) rewards to train our agent to solve complicated multi-step tasks.

Our approach which learns from human demonstrations and language outperforms or matches baseline methods in the standard RL setting. We demonstrate that language and word embeddings can provide prior knowledge and be used to better generalize to new tasks. We show that our method is able to, by providing a high-level task language, allow for more efficient learning of multi-step tasks in the standard RL setting. We also show that the agent can learn few-shot tasks with only a few additional demos and instructions and generalize with no new training to new tasks in a zero-shot setting. Finally, we show that training with human-generated instructions gives us an interpretable explanation of the agent's behavior in cases of success and failure. With our dataset collection procedure and language-conditioned method, we demonstrate that using natural human language and word embeddings can allow agents to solve difficult RL problems and begin solving the generalization problem in RL.

6.2 Related Work

Sketches: Previous works on language descriptions of tasks and sub-tasks have generally relied on what [13] calls "sketches." A sketch specifies the necessary sub-tasks for a final task and is manually constructed for every task. The agent then relies on reward signals from the sketches in order to learn these predefined sub-tasks. However, in our setup, we want to infer such "sketches" from a limited number of instructions given by human demonstrations. This setting is not only more difficult but also more realistic for practical applications of RL where we might not have a predefined ontology and simulator, just a limited number of human-generated instructions. In addition, at test time, their true zero-shot task requires the sketch, whereas our method is able to generate the “sketches” in the form of high-level language with no additional training and supervision.

Language and RL: Similarly, other works have used synthetically generated sub-goals and descriptions to train their methods and suffer from similar problems of impracticality. [312] introduces a Stochastic Temporal Grammar to enable interpretable multi-task RL in the Minecraft environment. Similarly, the BabyAI platform [69] presents a synthetic language which models commands inside a grid-based environment. They utilize curriculum training to approach learning complex skills and demonstrate through experimentation in their environment that existing approaches of pure IL or pure RL are extremely sample inefficient. [71] extend Hindsight Experience Replay (HER) to language goals in the BabyAI platform to solve a single instruction generated from a hand-crafted language. The BabyAI environment is extended by [53] to include descriptive texts of the environment to improve the generalization of RL agents. [152] also uses procedural generated language using the MuJoCo physics engine and the CLEVR engine to learn a hierarchical representation for multi-task RL. [257] also tackles zero-shot generalizations, but like the others considers only procedurally-generated instructions, learning to use analogies to learn correspondences between similar sub-tasks.

The main work that also investigates using a limited number of human-generated instructions in RL environments is [142]. This work also uses natural language instructions in hierarchical decision making to play a real-time strategy game involving moving troop units across long time scales. This work uses only behavioral cloning with natural language instructions, whereas we use a mixture of RL and imitation learning. They also do not investigate the benefits of language in zero-shot or few-shot settings and do not demonstrate cross-task generalization as we do.

Others have utilized natural language for other tasks, including [362], [72], and [14] but not focused on the multi-task learning setting. [228] demonstrates the use of word embeddings to inform robotic motor control as evidence of particular promise for exploiting the relationship between language and control. [247] uses language descriptions of the environment to aid domain transfer. The sub-field of language and vision navigation specifically has investigated how to train agents to navigate to a particular location in an environment given templated or natural language [11, 59, 62, 229, 336, 389] or to navigate to a particular location to answer a question [80]. Similarly to this work, [253] uses human-generated language to find objects in a simulated environment, [396] and [44] reads a document (i.e. a players manual) to play a variety of games, and [213] trains agents to follow both image and language-based goals. All of these works require the agent to read some text at both train and test time and follow those instructions to achieve some goal. In contrast, at test time, our agent only receives a high-level goal, which is what item to craft. Our agent must take the high-level goal as input and generates its own instructions to solve the task. In other words, our task is both instruction following and instruction generation. Related to instruction generation, some work have explored more generally intrinsic motivation for goal generation [111, 113]. In our work, however, we learn the goals via the human language instructions.

Hierarchical RL: Hierarchical approaches as a way of learning abstractions is well studied in the Hierarchical reinforcement learning (HRL) literature [84, 263, 325]. This is typically done by predefining the low-level policies by hand, by using some proxy reward to learn a diverse set of useful low-level policies [103, 110, 135, 137, 223], or more generally learning options [331]. Our approach differs in that unlike in options and other frameworks, we generate language as a high-level state which conditions the agent's policy rather than handing control over to low-level policies directly.

RL + IL: Other works have shown the effectiveness of using a combination of reinforcement learning and imitation learning. [186] presents a hybrid hierarchical reinforcement learning and imitation learning algorithm for the game Montezuma's revenge by leveraging IL for the high-level controller and RL for the low-level controller demonstrating the potential for combining IL and RL to achieve the benefits of both algorithms. By learning meta-actions, the agent is able to learn to solve the complex game. However, their meta-actions were also hand specified.

6.3 Human Annotation Collection

The first step of our approach requires human demonstrations and instructions. To that requirement, we built an interface to collect human-annotated data to guide the learning model.

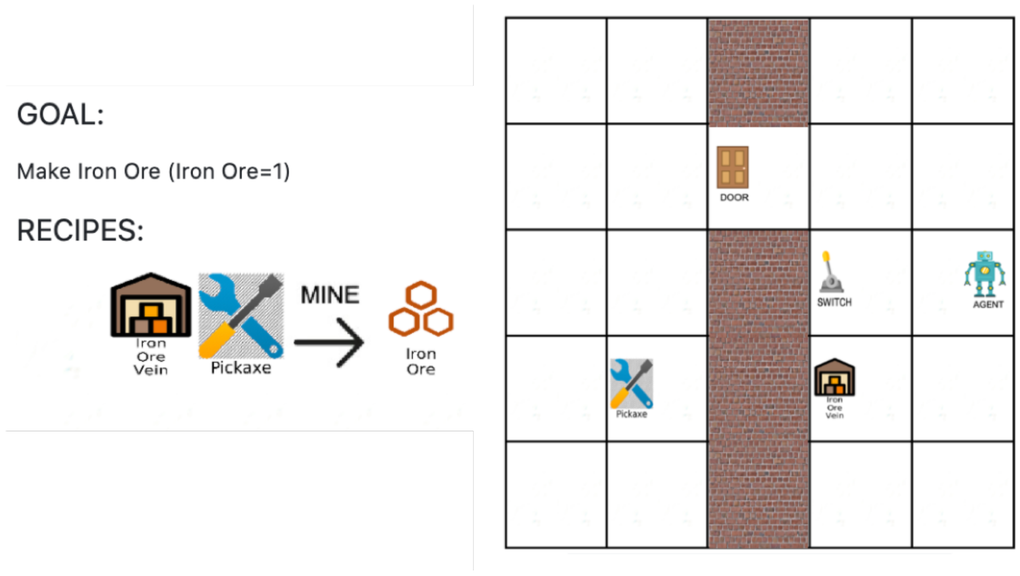


Figure 6.2: Example board and goal configuration where the goal is to make an iron ore. The worker uses the recipes provided to give appropriate instructions and execute accordingly. The agent is not given the recipe but must learn it or infer it based on its previous experience.

6.3.1 Crafting Environment

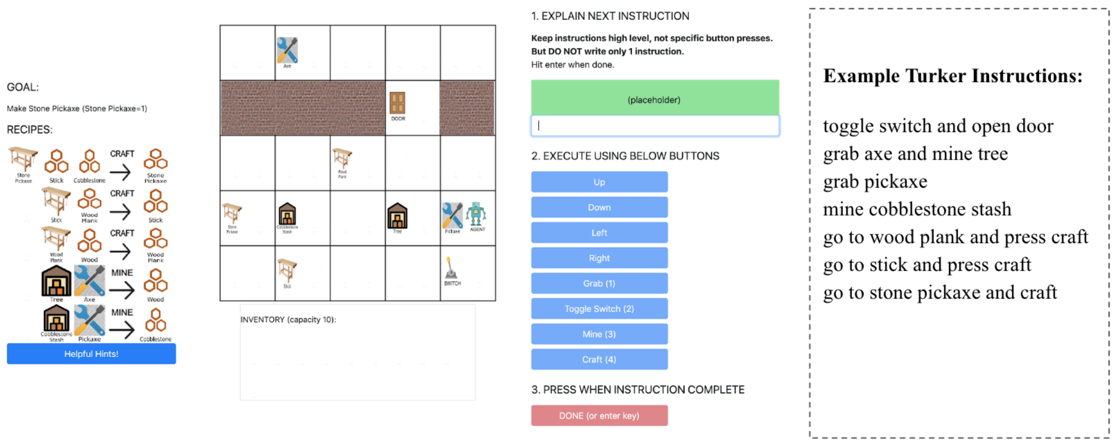
As shown in Figure 6.2, our environment is a Minecraft-inspired 5-by-5 gridworld. The Crafting Agent navigates the grid by moving up, down, left, and right. The agent can grab certain objects, like tools, if it is next to them and use the tools to mine resources. The agent must also use a key or switch to open doors blocking its path. Finally, the agent can also go to a crafting table to build final items. The agent can choose from 8 actions to execute: *up*, *down*, *left*, *right*, *toggle*, *grab*, *mine*, and *craft*. The environment is fully observable. Our crafting environment extends the crafting environment of [13] to include obstacles and crafts that are specified by material, introducing compositionally complex tasks (i.e. instead of "Make Axe", we have "Make Iron Axe" etc). In total, we consider about 50 crafting tasks, 14 of which we collect annotations for and 35 of which are used for test time. At the start of each game, all object/resource locations are fully randomized in the environment.

6.3.2 Crafting Task

The goal of the agent in our world is to complete crafts. By design, a crafting-based world allows for complexity and hierarchy in how the agent interacts with items in the gridworld. To craft an item, the agent must generally first pick up a tool, go to a resource, mine the resource, and then go to a table to craft the item. To make an iron ore, the agent must use the pickaxe at the Iron Ore Vein to mine Iron Ore to complete the task. The Iron Ore recipe is an example of a 1-step task because it creates one item. A 5-step task, like Diamond Pickaxe, involves the mining and/or crafting of 5 items. We capped the tasks at a maximum length of 5 recipe steps to limit the amount of time a worker would have to spend on the task. Note that each recipe step requires multiple time-steps to complete. Crafts are designed to share similar features and sub-steps to test whether the agent is able to learn these shared features and reuse existing knowledge to solve new, but related tasks more efficiently (these relations between tasks are detailed in Table 6.1 and in Figure 6.4). While the task may seem simple to human annotators to solve, such compositional tasks still pose difficulties for sparse-reward RL. We further increase the difficulty of this task by restricting the agent to a limited number of steps (100) to complete the task, leaving little room to make unrecoverable mistakes such as spending time collecting or using unnecessary resources.

6.3.3 Data Collection Process

Figure 6.3 shows our interface for human annotation. Given the goal craft, relevant recipes, and the initial board configuration, the worker provides step-by-step instructions accompanied by execution on the actual game board of each instruction. The workflow would be to type one instruction, execute the instruction, then type the next instruction, and execute until the goal was completed. The data collection interface and a corresponding example set of natural language instructions provided by a Turker is illustrated on the rightmost side of Figure 6.3. This is but one way that a Turker might choose to break down the 5-step crafting task.


GOAL:
Make Stone Pickaxe (Stone Pickaxe=1)
RECIPES:
CRAFT
MINE
Helpful Hints!
DOOR
AGENT
SWITCH
INVENTORY (capacity 10):
1. EXPLAIN NEXT INSTRUCTION
Keep instructions high level, not specific button presses.
But DO NOT write only 1 instruction.
Hit enter when done.
(placeholder)
2. EXECUTE USING BELOW BUTTONS
Up
Down
Left
Right
Grab (1)
Toggle Switch (2)
Mine (3)
Craft (4)
3. PRESS WHEN INSTRUCTION COMPLETE
DONE (or enter key)

Example Turker Instructions:

toggle switch and open door
grab axe and mine tree
grab pickaxe
mine cobblestone stash
go to wood plank and press craft
go to stick and press craft
go to stone pickaxe and craft

Figure 6.3: (Left) Example view of game interface that the worker would see on AMT. On the left the worker is given the goal and recipes; the board is in the middle; the worker provides annotations on the right. (Right) Example sequence of instructions provided by the Turker for the given task of Stone Pickaxe.

6.3.4 Dataset Analysis

Between the 14 crafts, we collected 6,322 games on AMT. In total, this dataset contains 195,405 state-action pairs and 35,901 total instructions. Figure 6.2 gives an example of the type of task (Make Iron Ore) that would be presented to the worker, which includes the goal, recipes, and the current board. We provide details of how the tasks are related in Figure 6.4 and Table 6.1 which shows how the tasks are related in terms of sub-tasks. This might be because of a similar material (i.e. Iron) or a similar craft (i.e. Stairs). In Table 6.2 we show the average number of steps and instructions required for each n-step task. Unsurprisingly, we find that the number of instructions and actions required increases with the number steps.

6.4 Methodology

Our proposed approach to solving these multi-step crafting tasks is to learn from human-generated natural language instructions and demonstrations. The model is first pretrained using imitation learning (IL) and then fine-tuned using sparse-reward in reinforcement learning (RL). The goal of the agent is to learn one policy that is

Table 6.1: List of recipes for which we have collected annotations, labeled by the number of steps needed to complete it and other recipes which may share sub-tasks of underlying structure.

ID	Recipe Name	Steps	Related Crafts by ID
1	Gold Ore	1	2
2	Iron Ore	1	1, 8
3	Diamond Boots	2	12, 14
4	Brick Stairs	2	5, 7
5	Cobblestone Stairs	2	4, 7, 13
6	Wooden Door	3	7
7	Wood Stairs	3	4, 5, 6
8	Iron Ingot	3	2
9	Leather Leggins	3	10, 11, 12
10	Leather Chestplate	3	9, 11, 12
11	Leather Helmet	3	9, 10, 12
12	Leather Boots	3	3, 9, 10, 11
13	Stone Pickaxe	5	5, 14
14	Diamond Pickaxe	5	3, 13

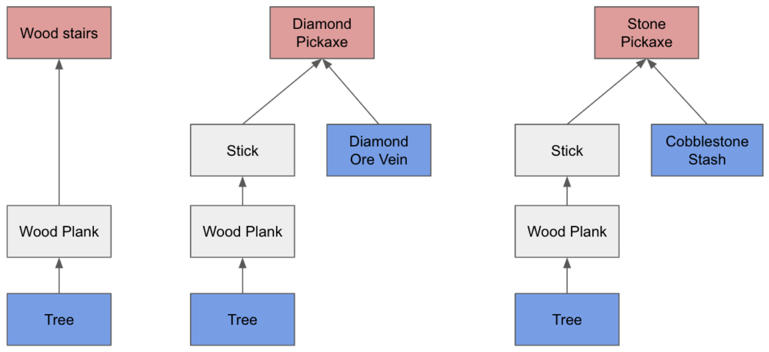


Figure 6.4: A more in-depth example of 3 out of the 14 training tasks to show how the subtasks are related (red boxes = final craft, blue boxes = raw material).

able to solve a variety of tasks (around 50) in the environment including ones it has not seen when only trained on a subset of the total tasks.

Table 6.2: Summary statistics for tasks of varying difficulty.

Steps	Average # of Instructions	Average # of Actions
1-step	3.7	15.4
2-step	4.9	21.5
3-step	6.1	27.6
5-step	8.8	40.1

6.4.1 Data Preprocessing

From the dataset, which had 6,322 game traces, we extracted 195,405 state-action pairs and 35,901 total instructions. This is done by matching an action to the corresponding state within a trace as well as the high-level natural language instruction. Each instruction was edited using a spell checker package to reduce the size of the final vocabulary.

6.4.2 Architecture

As outlined in Figure 6.5, we factor the agent into a hierarchical set-up with a language generator at the high-level and policy conditioned on the language at the low-level. At each time step, the state encoder produces a vector representation that is then used as input to both the language generator and language-conditioned policy. Relevant information about the state, including the grid, inventory, and goal are encoded. Items which are relevant for crafting are embedded using a 300-dimension GloVe embedding, summing the embeddings for multiple word items (i.e. Iron Ore Vein). Non-crafting items, such as door, wall, or key, are represented using a one-hot vector.

State Encoding

As shown in Figure 6.6, the relevant information about the state that is encoded includes the 5x5 grid, inventory, and goal. We have two representations of the 5x5 grid: one with items relevant for crafting and another with a one-hot representation of non-crafting-related items, such as a door, wall, or key. All crafting-related items on the board, inventory, and goal are embedded using a 300-dimension GloVe embedding,

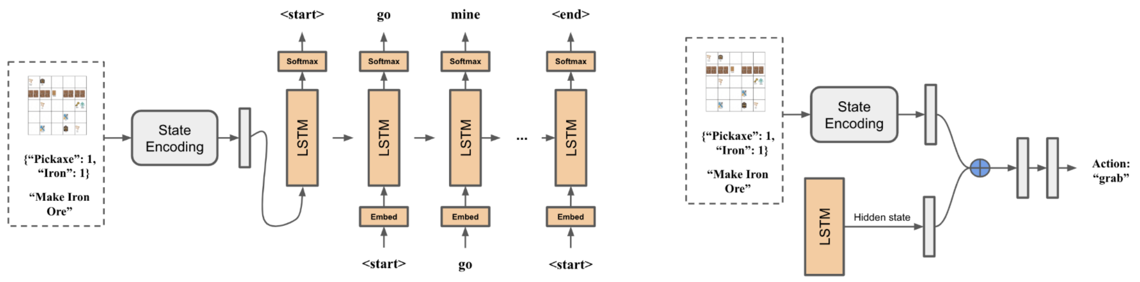


Figure 6.5: (Left) High-level language generator. (Right) Low-level policy conditioned on language.

summing the embeddings for multiple word items (i.e. Iron Ore Vein). The intuition for this distinction is that for generalization, crafting items should be associated in terms of compositionality, whereas non-crafting items are standalone.

To compute the state encoding, we first passed the two grids, inventory, and goal through separate fully connected layers to reduce to the same dimension and concatenated along the vectors. The final size of the state encoding tensor is (27, 128), where 25 are for the grid, 1 for inventory, and 1 for goal.

6.4.3 Training

Imitation Learning Pre-training

We warm-start the model using the human demonstrations. Language is generated at the high-level with an encoder-decoder framework. The encoding from the state encoder is decoded by an LSTM which generates a natural language instruction. The target language instruction is the AMT worker's provided instruction. In our dataset, the vocabulary size was 212, after filtering for words that appeared at least 5 times. At test time, we do not have access to the ground truth instructions, so instead the LSTM decoder feeds back the previously generated word as the next input and terminates when the stop token is generated. From the language generator module, we extract the last hidden state of the generated instruction. The hidden state is concatenated with the encoded state and passed through a series of fully connected layers. The final layer outputs the action. In the supervised training phase, the full model is trained by backpropagating through a language and action loss (cross

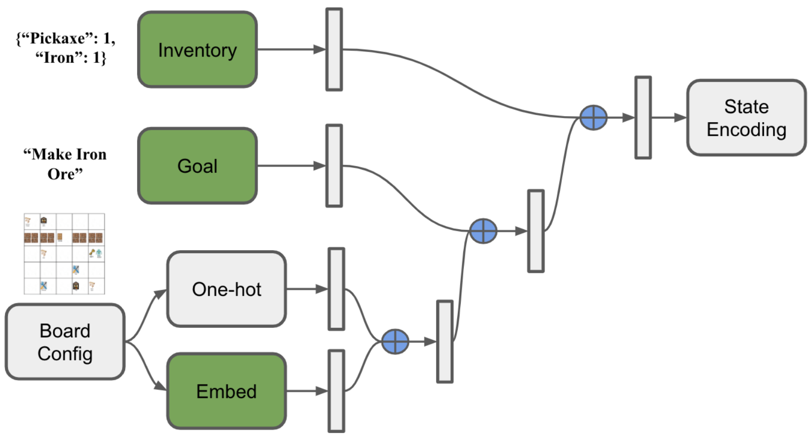


Figure 6.6: At each time step we encode state relevant observations including the goal, inventory, and grid. This encoding is utilized by both the language generator and the language-conditioned policy. The boxes in green, denote the observations that were encoded using the GloVe embedding.

entropy loss).

Reinforcement Learning Fine-tuning

We use the proximal policy optimization (PPO) algorithm [303] in RL with the reward defined below to learn an optimal policy to map from state encoding to output action. The maximum number of steps in an episode is set to 100. We utilize a training set-up which samples from all tasks (1-step through 5-step tasks). In preliminary experiments, we observe that sampling from 3-step tasks alone, for example, poses too complex of an exploration problem for the model to receive any reward. We define a sparse reward, where the agent only receives a reward when it has completed the full craft. In RL fine-tuning, we freeze the language generator component because there is no more language supervision provided in the simulated environment. We also find that empirically backpropagating the loss through language distorts the output language as there is no constraint for it to continue to be similar to human language. All training hyperparameters and other details are provided in the conference publication [64].

6.5 Experiments

6.5.1 Baseline Comparisons

We compare our method against five baselines (1-5) which are reduced forms of our method to evaluate the necessity of each component. We also consider two baselines (6-7), which swap out the language generator for alternative high-level tasks, to evaluate the usefulness of language as a selected high-level task. These baselines have the additional training that our method received, as well as the implicit compositionality, but without language. In both baselines (6-7), we perform the same training steps as with our method.

1. **IL**: The IL baseline uses the same low-level architecture as our method, without a high-level hidden state. The model learns to map state encoding to an output action.
2. **IL w/ Generative Language**: IL w/ Generative Language is the supervised baseline of our method, which does not include RL reward. This baseline allows us to observe and compare the benefit of having a reward to train in simulation when the model has access to both actions and language instructions.
3. **IL w/ Discriminative Language**: We compare our method to a closely adapted version of the method proposed in [142] which similarly uses language in the high-level. Rather than generate language, their high-level language is selected from a set of instructions from the collected user annotation. They consider instruction sets of sizes $N = \{50, 250, 500\}$ and find the best performance on the largest instruction set $N = 500$ which is the size we use in our implementation.
4. **RL**: Another baseline we consider is the reinforcement learning (RL) setting where the agent is provided no demonstrations but has access to sparse-reward in RL. The architecture we use here is the same as the IL architecture. This baseline demonstrates the capacity to learn the crafting tasks without any human demonstrations and allows us to see whether human demonstrations are useful.

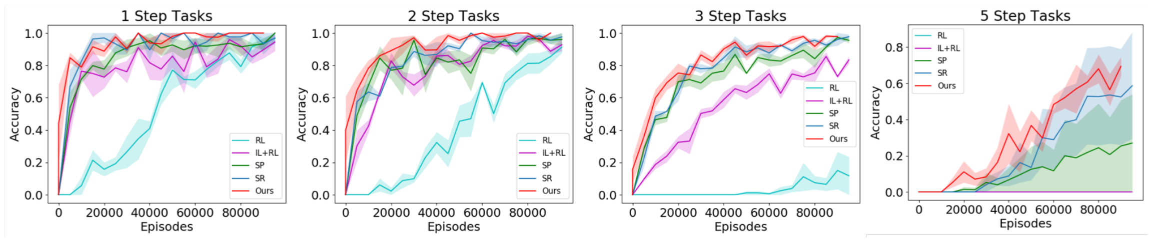


Figure 6.7: Comparing baselines with our method on accuracy. Human demonstrations are necessary to complete tasks with 3 or more steps. Averaged over 3 runs.

5. **IL + RL**: We also consider a baseline that does not incorporate language which is IL+RL. In IL+RL, we pretrain the same IL architecture using the human demonstrations as a warm-start to RL. It is important to note that this baseline does not include the natural language instructions as a part of training. We extract all of the state-action pairs at train a supervised model on the data as in the IL model and then we utilize the RL sparse-reward to fine-tune.
6. **State Reconstruction (SR)**: We train an autoencoder to perform state reconstruction. The autoencoder reconstructs the state encoding and the vector at the bottleneck of the autoencoder is used as the hidden layer for the policy. SR as a baseline allows us to consider latent representations in the state encoding as a signal for the policy.
7. **State Prediction (SP)**: We train a recurrent network, with the same architecture as our language generator, to perform state prediction. The model stores the past 3 states from time T to predict the $T + 1$ state. So at time T, the states $T - 2$, $T - 1$, and T are used to predict state $T + 1$. From the LSTM, the hidden state is extracted in the same manner as our IL + RL w/ Lang model. SP as a baseline allows us to compare against another recurrent high-level method with the additional computation power.

6.5.2 Standard setting

We evaluate the various methods on crafts which we have collected human demonstrations for to benchmark comparative performance in our environment. An initial analysis is to first consider how much the IL model is able to learn from human demonstrations alone, so we consider IL, IL with Generative Language, and IL with

Table 6.3: Accuracy of IL (with and without language) evaluated over 100 games with 3 different seeds.

Steps	IL no language	IL Gen. Language	IL Disc. Language
1-step	18.00%± 3.55%	19.33% ± 1.89%	20.00% ± 1.35%
2-step	0.00%± 0.00%	9.33% ± 2.05%	8.33%± 0.98%
3-step	0.00%± 0.00%	4.33% ± 0.47%	0.00%± 0.00%
5-step	0.00%± 0.00%	0.00%± 0.00%	0.00%± 0.00%

Discriminative Language (see Table 6.3). None of these approaches are able to solve the most difficult 5-step tasks or the simpler tasks consistently, with an average of about 18-19% success rate for 1-step tasks. We believe the 3 and 5-step tasks are difficult enough such that annotations alone were not able to capture the diversity of board configurations for the variety of crafts given that the board is randomly initialized each time. However, based on an analysis of the language selected (see Table 6.7), the generated language is more interpretable and made more sense in a zero shot setting. Given the language is fixed after this point, all remaining experiments moving forward use generative language.

As shown in Figure 6.7, our method performs well against baselines. We find that human demonstrations are necessary to guide learning because the learned behavior for RL is essential to arbitrarily walk around the grid and interact with items. For simple 1 and 2 step tasks, this is a feasible strategy for the allotted steps for an episode. However, there is little room for error in the most difficult 5-step tasks, as even human demonstrations take on average 40 steps to solve. We also find that for the standard setting, incorporating a high-level network allows the model to achieve good results when comparing our method to SP and SR.

In Figure 6.8 we show the result of our method when we ablate the number of demonstrations we use. This lets us see how many demonstrations we would feasibly need for the model to learn how to solve the crafting tasks. As we decrease the amount of data provided, we find that there is greater variance in the policy's ability to complete the task, but the performance only significantly degrades when we start using only 25% of the data on the hardest tasks.

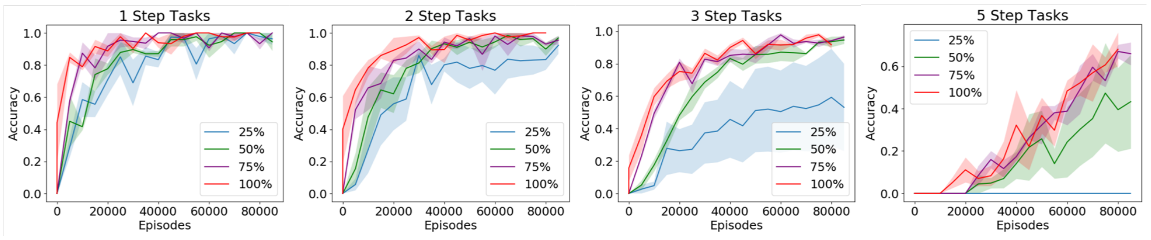


Figure 6.8: Ablation of our method with varying amounts of human annotations (25%, 50%, 75% and 100%). For each fraction, we sample that number of demonstrations from the dataset for each type of task. Averaged over 3 runs.

6.5.3 Zero-Shot

Our method is able to use natural language instructions to improve performance on difficult tasks in the standard setting. But how well is our method able to do on completely new tasks not seen during training? We investigate our performance on zero-shot tasks, where the agent receives no human demonstrations or instructions, and no rewards on these tasks. The agent has to try to complete these tasks that it has never seen before and cannot train on at all. These unseen tasks do share sub-task structure with tasks which were seen in the training process, so the desired behavior is for the model to reuse subpolicies seen in other contexts for this new context. For example, in training the agent might have seen demonstrations or received rewards for a task like “Cobblestone Stairs” and “Iron Ingot.” At test time, we can evaluate the agent on an item like “Cobblestone Ingot”, which has never been seen by the agent. The agent should be able to infer the sub-task breakdown given prior knowledge of similar tasks.

We present 35 examples of unseen tasks in Table 6.4. We find that overall our method outperforms all other baselines. While SR and SP were able to match our method’s performance in standard setting, they are not able to generalize. SR and SP are viable solutions to learn complex tasks in the standard RL setting, but the representations these models learned do not aid in generalizing to unseen tasks. Here, we believe, using language and word embeddings as prior knowledge is key because it creates a representation that better abstracts to new tasks. We see that in the cases of unseen tasks, the model indeed is able to generate language that properly corresponds to these new combinations of materials and items, particularly decomposing the complex item into sub-tasks that were previously seen in the training phase (see Table 6.8).

Table 6.4: Accuracy evaluated on 100 games for 35 unseen crafts. Our method outperforms baselines. We do not list IL or IL w/ Language results which are 0% for all tasks.

	Diamond Stairs	Cobblestone Boots	Stone Boots	Brick Boots	Cobblestone Leggins	Stone Leggins	Brick Leggins	Diamond Leggins	Cobblestone Door	Stone Door	Brick Door	Diamond Door	Cobblestone Chestplate	Stone Chestplate	Brick Chestplate	Diamond Chestplate	Cobblestone Helmet	Stone Helmet	**2-AVG**
Steps	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
RL	93	91	95	90	92	92	91	81	0	0	0	0	0	0	13	0	72	28	46
IL+RL	92	98	85	94	91	83	95	97	21	96	37	18	97	82	97	93	94	91	81
SP	0	20	0	33	37	2	69	2	90	0	98	0	89	1	78	1	74	2	33
SR	96	64	0	0	67	70	60	99	0	79	88	74	69	37	16	98	70	0	55
Ours	**99**	**99**	**100**	**100**	**93**	**99**	**100**	**100**	**97**	**98**	**99**	**99**	**99**	**100**	**97**	**100**	**99**	**97**	**98**

	Leather Stairs	Wood Boots	Cobblestone Ingot	Stone Ingot	Gold Ingot	Brick Ingot	Diamond Ingot	Wood Ingot	Wood Leggins	Leather Door	Wood Chestplate	Wood Helmet	**3-AVG**	Brick Pickaxe	Leather Pickaxe	Wood Pickaxe	Iron Pickaxe	Gold Pickaxe	**5-AVG**	**Overall-AVG**
Steps	3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5	–
RL	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23
IL+RL	90	87	0	0	0	0	0	0	85	29	86	87	39	0	0	0	0	0	0	55
SP	89	0	**12**	0	4	**10**	0	**47**	0	1	26	0	16	0	0	0	0	0	0	22
SR	1	0	2	0	12	0	0	0	0	38	6	0	5	0	0	0	0	0	0	30
Ours	**97**	**98**	2	**3**	**18**	0	**40**	0	**96**	**39**	**95**	**98**	**49**	**36**	0	0	0	**14**	**10**	**69**

6.5.4 Demonstration Only and Few-Shot

In the demonstration only, we assume that we have access to only human demonstrations for some subset of tasks. From the entire pool of 14 tasks we collected demonstrations for, we withhold 3 tasks (around 20% of total tasks) for testing. These 3 tasks consist of a one, two, and three step task. We run results on 3 permutations of withholding 3 tasks. For each of the 3 withheld tasks, we include these demonstrations in the supervised training phase but do not provide reward in RL fine-tuning. We vary the amount of demonstrations that are provided: 5%, 10%, and 100%. The most generous case is to assume that the model has access to all demonstrations that were collected in the dataset. Per task, the total number of demonstrations was about 300-500. Additionally we considered a more strict few-shot case where we reduce the number of demonstrations to 20-40 which is about 5-10% of the original number of demonstrations. We do not include 5-step tasks because we only collected demonstrations for two 5-step tasks. From the results in Table 6.5, we can see that our method outperforms baselines in its ability to utilize the few demonstrations to improve performance.

Table 6.5: Evaluation of few-shot tasks for our method against baseline comparisons. We consider three settings for how many demonstrations are given to the model: 5% (20 demos), 10% (40 demos), 100%. Results are averaged across 3 seeds.

	IL			IL w/ Lang			IL+RL			SP			SR			Ours		
Steps	5%	10%	100%	5%	10%	100%	5%	10%	100%	5%	10%	100%	5%	10%	100%	5%	10%	100%
1-step	16%	18%	18%	17%	19%	19%	96%	91%	98%	96%	98%	97%	53%	84%	94%	97%	90%	95%
2-step	4%	3%	0%	5%	5%	9%	66%	64%	66%	53%	64%	71%	10%	40%	63%	87%	73%	82%
3-step	1%	2%	0%	1%	3%	4%	1%	23%	22%	10%	27%	46%	0%	31%	50%	5%	47%	74%

6.5.5 Reward Only

Finally, for completeness, we consider the scenario where the agent receives a reward but no demonstrations. The tasks which we select for this setting are sampled from the unseen tasks list. We choose 3 2-5 step crafts. We evaluate this scenario on our method against other baselines which train using a reward signal. In Table 6.6, we evaluate on tasks for which we do not have demonstrations and fine-tune a trained model with the reward signal for these tasks. This setting is not very interesting from a generalization perspective, since rewards are a far more expensive resource compared to demonstrations and instructions. We don't include 1-step tasks since that is able to be solved easily by RL alone (see 1-step results in Figure 6.7). IL and IL w/ Language is not included because this reduces to the zero-shot setting.

Table 6.6: Comparison of 2-5 step tasks where only reward is provided to the agent. We believe IL+RL is not able to adapt to these new tasks, given reward only, since it has overfit to the original training tasks. We find that our method outperforms baselines in this setting.

Steps	RL	IL+RL	Ours
2-step	92.00%±0.81%	0%	95.33%±0.94%
3-step	71.67%±0.47%	0%	88.00%±1.41%
5-step	0.00%±0.00%	0%	65.00%±5.67%

6.5.6 Interpretability

One key benefit of incorporating natural language into the model is the ability for humans to interpret how the model is making decisions. We observe that the generated instructions closely match those of the recipes that we provide to the annotators in the data collection phase in both train (Table 6.7) and test (Table 6.8) settings. Figure 6.9 presents example instructions generated by our model.

Table 6.7: Step-by-step generated high-level instructions for seen crafts.

Goal: Gold Ore go to key and press grab. go to pickaxe and grab. go to gold ore vein and mine.	Goal: Brick Stairs go to pickaxe and press grab. go to the brick factor and mine brick. go to brick stairs and press craft.
Goal: Diamond Pickaxe go to axe and press grab. go to key grab it go to door and open door. go to tools and click grab to take each one. go to tree and press mine. go to stocks click mine to harvest. go to tree and mine. go to wood plank and press craft. go to stick bench and craft stick.	Goal: Wooden Door go to the axe and grab it. go to the switch and open door. go to the axe and grab it. go to the tree. go to the tree and press mine. go to wood plank and press craft. go to wood plank bench and craft wooden door.
Goal: Leather Helmet go to sword and click grab to take it. go to key and press grab. go to sword and click grab to take it. go to rabbit and press mine. go to leather and press craft. go to leather boots bench and craft leather.	Goal: Diamond Boots go to key and press grab. go to pickaxe and press grab. go to diamond ore vein and mine. go to diamond boots and press craft. go to diamond bench and craft diamond boots.
Goal: Iron Ore go to key and press grab. go to pickaxe and press grab. go to iron ore vein and press mine.	Goal: Cobblestone Stairs go to key and press grab. go to pickaxe and press grab. go to cobblestone stash and press mine. go to cobblestone stairs and press craft.
Goal: Wood Stairs go to axe and press grab. go to tree and mine. go to wood plank and press craft. go to wood stairs and press craft.	Goal: Leather Chestplate go to sword and press grab. go to rabbit and mine. go to leather and craft. go to leather chestplate and craft.
Goal: Leather Leggins go to sword and click grab to take it. go to rabbit and press mine. go to leather and press craft. go to leather bench and craft leather	Goal: Iron Ingot go to key and press grab. go to pickaxe and press grab. go to iron ore vein and mine. go to iron ingot and craft.

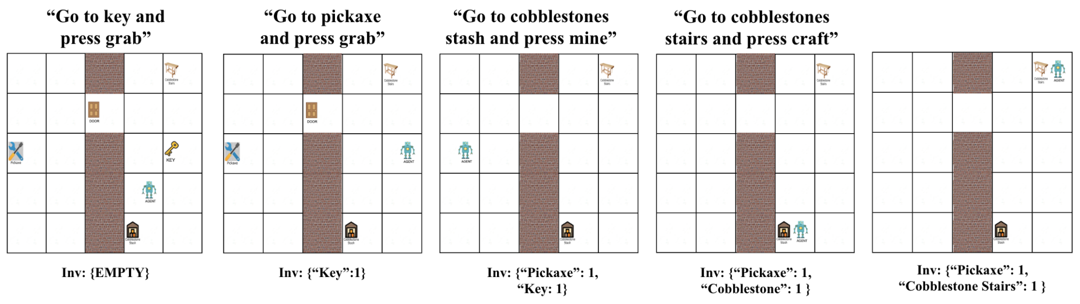

"Go to key and press grab"
"Go to pickaxe and press grab"
"Go to cobblestones stash and press mine"
"Go to cobblestones stairs and press craft"
Inv: {EMPTY}
Inv: {"Key":1}
Inv: {"Pickaxe": 1, "Key: 1}
Inv: {"Pickaxe": 1, "Cobblestone": 1 }
Inv: {"Pickaxe": 1, "Cobblestone Stairs": 1 }

Figure 6.9: Generated language at test time for a 2-step craft. We only display key frames of the trajectory which led to changes in the language. These key frames match changes in the inventory to the object mentioned in the generated instruction. Qualitatively, the generated instructions are consistent during what we would describe as a sub-task. Quantitatively, the network will spend on average 4.8 steps in the environment for the same generated language output.

Table 6.8: Step-by-step generated high-level instructions for unseen crafts.

Goal: Cobblestone Boots go to key and press grab. go to pickaxe and press grab. go to cobblestone stash and mine. go to workbench and press craft.	Goal: Diamond Leggins go to pickaxe and press grab. go to diamond ore vein and mine. go to diamond boots and press craft.
Goal: Leather Stairs go to sword and press grab. go to rabbit and mine the rabbit. go to leather and press craft.	Goal: Stone Helmet go to pickaxe and press grab. go to the cobblestones stash and mine. go to the workbench and craft.
Goal: Diamond Ingot go to pickaxe and press grab. go to diamond ore vein. go to the workbench and craft.	Goal: Brick Door go to pickaxe and press grab. go to the brick factory and mine the brick. go to the brick stairs and craft.
Goal: Brick Pickaxe go to the pickaxe and grab it go to the axe and press grab. go to the tree. go to the tree and mine. go to the brick factory and mine. go to the wood plank and craft. go to the stick bench and craft stick. go to stick and craft.	Goal: Gold Pickaxe go to the pickaxe and press grab. go to the axe and grab it. go to the tree. go to stocks and click mine to harvest <unk>. go to the tree and mine the tree. go to wood plank and press craft. go to stick and press craft.
Goal: Diamond Stairs go to key and press grab. go to pickaxe and press grab. go to the diamond ore vein and mine. go to the bench and craft.	Goal: Wood Chestplate go to key and grab it. go to axe and grab it. go to the tree. go to tree and mine. go to wood plank and craft.

6.6 Conclusion

In this work, we present a dataset of human demonstrations and natural language instructions to solve hierarchical tasks in a crafting-based world. We also describe a hierarchical model to enable efficient learning from this data through a combined supervised and reinforcement learning approach. In general, we find that leveraging human demonstrations and the knowledge from word embeddings allows the model to drastically outperform RL baselines. Additionally, our results demonstrate that natural language not only allows the model to explain its decisions but the additional prior knowledge improves the model's performance on the most difficult crafting tasks and further allows generalization to unseen tasks. We also demonstrate the model's ability to expand its skillset through few additional human demonstrations.

While we demonstrate our approach's success in a grid-based crafting environment, we believe that our method is able to be adapted towards generalizable, multi-task learning in a variety of other environments. In the next chapter, we discuss applying prior knowledge (in this case both word embeddings and knowledge graphs) to robotic grasping.

Chapter 7

Knowledge in Action: Robotics

7.1 Introduction

In this chapter, we continue exploring the use of knowledge for action, this time in the much more difficult setting of robotics, which also necessitates the use of a visual modality. We extend prior work on semantic grasping by introducing a new dataset and incorporating knowledge into the grasping methodology.

We have seen tremendous progress in the fundamental task of robotic grasping in recent years. State-of-the-art grasping algorithms have shown generalization to object instances [160, 215, 273, 393], viewpoints [191], DOF constraints [241, 245, 338], unknown environments [130] and even adversarial objects [354]. The key reason for the success of these approaches is large-scale learning. Typically data is sampled from analytical approaches in simulation [215, 241] or using a self-supervised framework [191, 273]. Despite these recent successes, there is still a significant gap between how humans grasp objects and how robots perform picking. Most techniques plan for stable grasps assume grasping to be the end goal. However, when humans grasp an object, we do so with a particular purpose in mind and grasping is just the first step as a means to that end. For example, when humans grasp a cup, we use the handle to drink from it though several other stable grasps exist. Humans also use objects creatively, such as scooping with a bowl or hammering with a heavy mug. Different tasks may require completely different grasps for the same object. To effectively operate in human homes and complete multiple tasks, a personal robot would have to learn from humans to generalize grasping to several tasks and skills beyond a tool's prototypical use. For instance, if the robot is cooking and needs to stir a pot of pasta but doesn't have a spoon at hand, it can use an alternate tool, such as a knife. To truly get to human-level grasping, we must study not just stable grasping or grasping for an object's primary use-case but rather how to grasp depending on both the task and the object.

What are the bottlenecks in Task-Oriented robotic grasping? One of the biggest hurdles is the need for human-labeled data. Unlike self-supervised or analytical approaches for which force sensing or contact models can provide labels for stable grasps, here we need humans to identify how an object can be grasped for multiple tasks. There has been a lot of recent work in this area, including [42, 105, 202]. Brahmbhatt et al. [42] used thermal imaging in a curated setup to study human grasping contacts on 50 3D printed objects for two tasks. Fang et al. [105] proposed to jointly learn a task-oriented grasping network and manipulation policy in simulation with reinforcement learning and demonstrated the framework on two-goal tasks with two object categories. Liu et al. [202] proposed a data-driven approach to learning the complex relationships between grasps, objects, tasks, and broadened semantic contexts. However, their approach required pixel-wise affordance segmentation [95] for a small set of known object categories, which is challenging to generalize and get supervision for.

Despite this progress in learning from human grasping, there are still significant gaps, both from a data and methods perspective. On the data side, existing datasets are limited in terms of the number of object instances, but especially in the number of tasks and object classes collected. Yet, even if we scale the datasets, it is unclear if current approaches will generalize to new object categories and tasks in the real world. We tackle both problems: first, we collect a dataset that is diverse both in terms of objects and tasks and an order of magnitude larger than previous datasets. Second, we exploit the semantic knowledge of objects and tasks to create a system that can generalize to new object instances, classes, and new tasks. To the best of our knowledge, this work is one of the first efforts in demonstrating robust generalization in task-oriented grasping, especially with semantic knowledge.

First, we collect a large-scale dataset which we call TaskGrasp. We increase the number of real objects from 50 in prior works [42] to 191, and collect RGB-D point cloud observations and object-centric 6-DOF grasps for the task-oriented grasping problem. We also scale the number of object classes from 40 [42] to 75 and resolve each of these to the WordNet ontology [235]. And perhaps most importantly, we scale the number of tasks from 2-7 in prior works [42, 105, 202] to 56. This expanded dataset both gives a better benchmark for task-oriented grasping and allows us to study generalization by expanding the number of object categories and tasks.

Now that we have developed this semantically richer dataset, we can also begin to incorporate knowledge and semantics into the task of task oriented grasping. In order to generalize to a new object or task, we need to have some prior knowledge about it. For instance, if we knew that mugs and bowls were both containers, we might infer that we should apply the scoop action in a similar way. To this end we propose a method, called GCNGrasp, that incorporates semantic knowledge into the end-to-end learning of task-oriented grasping from object point clouds. In particular, we use a Graph Convolutional Network (GCN) [170] to reason about a knowledge graph that encodes relations between objects and tasks, and further leverage word embeddings trained on large-scale language tasks to provide additional prior information. Our GCNGrasp model shows a significant improvement of 12% and 3.5% on held-out tasks and object categories, respectively, compared to baselines which do not incorporate semantics. We also show that our method and dataset are applicable for actual robots by executing task-oriented stable grasps on a 7-DOF Sawyer Robot on unknown objects.

7.2 Related Work

Task-Oriented Grasping: Prior work in Task-Oriented Grasping can be grouped into analytic methods, data-driven approaches using object state information, and frameworks learning from observations. Early work in analytic grasping proposed task wrench spaces with task-oriented grasp quality metrics [36]. Data-driven approaches have been proposed to improve generalization, though a large body of work has relied on object state information. Song et al. [318] used generative Bayesian Networks to model the relations between objects, grasps and tasks; Antanas et al. [15] and Ardón et al. [17] leveraged probabilistic logic languages to reason about grasp regions affording different tasks through semantic relations. However, both methods require grounding geometric information about objects to semantic representations and can only reason about semantic knowledge alone. A related line of work has used object parts and affordance detection [89, 95, 173, 182]. Do et al. [95] leveraged the affordances of object parts to define the correspondences between affordances and grasp types (e.g., rim grasp for parts with contain or scoop affordance). Detry et al. [89] trained a separate affordance detection model using synthetic data to detect suitable grasp regions for each task. While we do not provide explicit supervision for object affordance, we demonstrate that our model achieves an implicit understanding.

More recent works have learned task-oriented grasping from just RGB-D observations of objects. Dang and Allen [78] proposed an example-based approach which learns task-oriented grasps by storing visual and tactile data of grasps. Hjelm et al. [140] proposed a discriminative model based on visual features of objects. Jang et al. [149] proposed an end-to-end learning method of grasping objects from specific categories in a bin. To accelerate learning from observations, there have been efforts in scaling datasets as discussed previously [42, 105, 202]. The computer vision community has also focused on annotating datasets for inferring human grasp pose estimation from visual data [144, 174, 305] with the aim that it could be adapted to robotic grasping with kinematic retargeting. In this work, we propose an expanded dataset in terms of the number of object categories and tasks to study generalization. We also present a unified framework that jointly learns from semantic knowledge and geometric observations.

Semantic Knowledge in Vision and Robotics: In Chapter 2.7 we discuss the prior work in knowledge in computer vision and in Chapter 2.8 we discuss the work in RL and robotics. What we find is extensive work on using word embeddings, both in CV [114] and robotics [228] as well as a variety of techniques in both that use class hierarchies [401]. Knowledge graphs are less common, however, in the robotics community. Methods leveraged more specific knowledge in a variety of robotic tasks, such as affordance learning [239] and visual-semantic navigation [377]. Similar to Antanas et al. [15] and Ardón et al. [17], we reason about semantic knowledge for task-oriented grasping, but we leverage semantic knowledge for generalization to novel object classes and tasks.

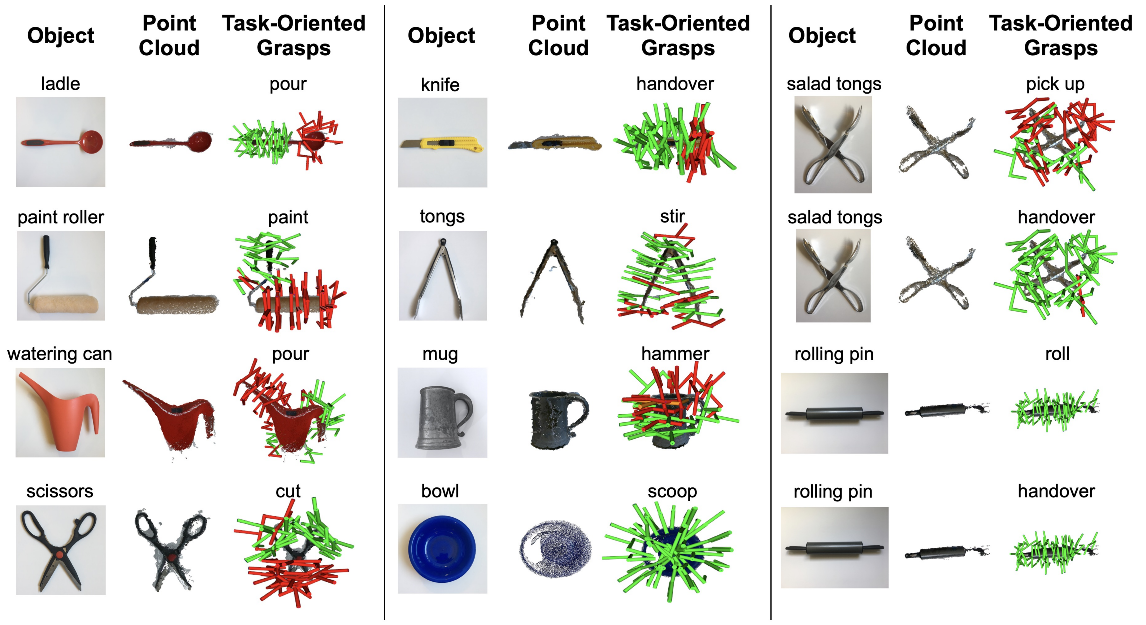


Figure 7.1: Example point clouds and grasps from our TaskGrasp dataset. Column 7-9 shows how grasps vary with tasks for a salad tongs (with higher diversity) and a rolling pin (with lower diversity). Green and Red means successful and incorrect task-oriented grasps respectively.

7.3 Dataset

In this section we describe our dataset: TaskGrasp, specifically its properties, collection and annotation methodology. As shown in Table 7.1, TaskGrasp is the largest and most diverse dataset for task-oriented grasping to date with respect to number of objects, categories and tasks.

TaskGrasp contains 191 individual household and kitchen objects comprising 75 distinct object categories and varying in size, geometry, material, and visual appearance. Figure 7.2 shows the class of each object and its proportion in the dataset. We collect RGB-D point clouds for each object, and automatically annotate $250K$ stable grasps. We also curate a list of 56 everyday tasks that impose different semantic constraints on grasping and annotate for each grasp whether that grasp is appropriate for each particular task.

Table 7.1: Comparing recent Task-Oriented Grasping Datasets

	ContactDB [42]	SG14000 [202]	TOG-Net [105]	TaskGrasp (Ours)
Semantic Knowledge	✗	✗	✗	✓
Object Categories	40	5	2	75
Objects	50	44	18K (synthetic)	191
Tasks	2	7	2	56
Grasps	3750	14K	1.5M	250K
Grasp Type	Contact Map	$SE(3)$	Planar	$SE(3)$

7.3.1 Data Acquisition on a Robot

After selecting our 191 objects by browsing various homegoods stores, we scan the objects to acquire their point clouds. A RealSense D415 eye-in-hand camera mounted on a LoCoBot [243] is used for 3D scanning. The object is placed on a transparent mount in front of the robot, which is commanded to different poses along the object approach direction to capture point clouds from multiple viewpoints. This setup helps to capture more of the object geometry under self-occlusion, which in turn increases the coverage of grasp samples. The multi-view observations are registered using robot kinematics and further refined with the iterative closest point algorithm. After table plane segmentation, 600 object-centric stable grasps are then sampled [337] from the object point cloud. 25 grasps are selected with farthest point sampling (to maximize grasp coverage) for annotation. These grasps are chosen as a representative, albeit limited, grasp set for the object to trade off between dataset size and budget.

7.3.2 Data Annotation by Crowdsourcing

We use Amazon Mechanical Turk (AMT) to crowdsource labels for the 250K stable grasps. Instead of exhaustively labeling each task-object combination ($\sim 10K$) , we reduce the annotation cost with a two-stage procedure. We use the insight that the pre-condition for a task-oriented grasp is that the object has to be capable of the task in the first place. First, we gather labels for whether a task is suitable for each object. Second, for this filtered subset of task-object combinations, we collect labels for the 25 task-oriented grasps per object. To ensure annotation quality, we assign each labeling task to three annotators and use gold standard questions (questions that we know the answers to) to filter annotators with low accuracy. For both stages, we take a majority vote between the annotators. We measure agreement with Randolph's free-marginal multirater kappa [283]. Kappa values for the two stages are 0.65 and 0.62 respectively (0.0 meaning agreement equal to chance, and 1.0 indicating perfect agreement above chance), which suggests good agreement between annotators.

7.3.3 Analysis

In Figure 7.1 we show prototypical examples from TaskGrasp.

Diversity of Grasps: As a result of the large number of objects and tasks, TaskGrasp contains a wide variety of task-oriented grasps. On average, each object is suitable for 7 tasks. As shown in Figure 7.1, these tasks involve both prototypical (a ladle for pouring) and creative use of objects (tongs for stirring), imposing drastically different semantic constraints on grasping. These examples also demonstrate the complex geometries presented in real world objects, which pose another challenge for generalization.

We also quantitatively measure grasp diversity by analyzing the effect of tasks on grasps. Since different tasks provide different labels for the same set of stable grasps on each object, we compute Randolph's kappa [283] on these labels as a measure of agreement between tasks, i.e., how likely grasps for one task (e.g., stir) agree with grasps for another task (e.g., cut). Ranging from 0.19 to 0.93, kappa values of the objects suggest that the effect of tasks vary greatly for different objects. Column 7-9 in Figure 7.1 show how grasps vary with tasks for a salad tongs with a kappa value of 0.38 and a rolling pin with kappa value of 0.97. In TaskGrasp, 25% of the objects have kappa values lower than 0.5 and these objects require significantly different grasps for different tasks.

Semantic Knowledge of Objects and Tasks: We also provide semantic knowledge about objects and tasks in the dataset. Objects are manually mapped to WordNet synsets [235] which represent a semantic hierarchy, as shown in Figure 7.2. Each of the 75 leaf synsets in the hierarchy represents a distinct object class and is linked to 2.5 objects on average. Building on the hypernym paths from WordNet, the semantic hierarchy includes a rich set of object concepts interlinked by "Is-A" relations. This provides useful semantic knowledge for task-oriented grasping as objects in the same subtree of the hierarchy often share similar functionalities or

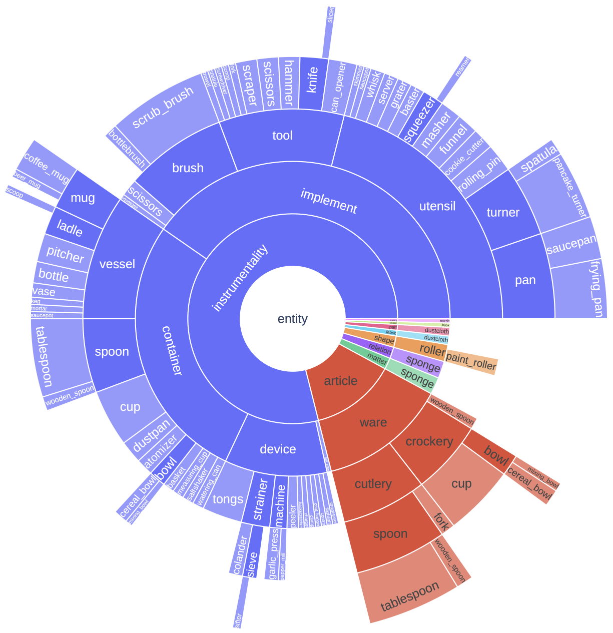


Figure 7.2: Semantic hierarchy of objects. Each level of the hierarchy is represented by one ring with the innermost circle as the root of the hierarchy. The angle of each segment is proportional to the number of objects.

geometric properties. For example, mug, ladle, and bottle are in the vessel subtree and can all be used to hold liquid. In addition, we connect a task to an object class through "Used-For" relations if any object in the class is considered suitable for the task from the first stage of our crowdsourcing.

7.4 Methodology

We consider the problem of generating grasps for task-oriented grasping given the object point cloud and task constraints. Specifically, we want to estimate the grasp distribution $P(G^*|X, \mathcal{T})$, where X is the point cloud input, $\mathcal{T}$ are the constraints imposed by goal tasks, and G^* is the space of successful grasps. Following convention in related work [241, 338], we represent grasps $g \in G^*$ as the grasp pose $(R, T) \in SE(3)$ [1] of a parallel-jaw gripper with its fingers open which when closing will lead to a stable grasp. We further factorize the estimation of $P(G^*|X, \mathcal{T})$ into task-agnostic grasp

[1]SE(3) stands for 3D special euclidean space and is represented as a 3D rotation and a 3D rotation.

sampling $P(G^*|X)$ and task-oriented grasp evaluation $P(S|X, \mathcal{T}, g)$. The primary benefit of this factorization is that it allows us to leverage prior work in stable grasp generation.

In this section, we describe our method (GCNGrasp) for Task-Oriented grasping. Our method is composed of:

1. A Grasp and Object Encoder built on a PointNet++ architecture [275] to encode the object point cloud and grasp
2. A Graph Convolutional Network [170] which takes the encoded object shape and grasp as input as well as a knowledge graph $\mathcal{G}$ encoding the semantic relationships between object categories, tasks and hierarchies
3. A Grasp Evaluator which outputs the final grasp prediction. See Fig 7.3 for an overview of the framework.

7.4.1 Grasp and Object Encoder

Our input observations are object point clouds and we want to reason about $SE(3)$ grasps. Qi et al. [275] proposed the PointNet++ architecture to efficiently represent 3D data which we use to learn a representation for the object point cloud and 6-DOF grasp poses. The grasp g is defined in the object frame and six control points are selected on the gripper to form a gripper point cloud X_g. Similar to Mousavian et al. [241], X_g is concatenated with the object point cloud X with an extra latent indicator vector to represent whether a point is part of the gripper or the object. The PointNet layer reasons about the relative spatial information between the grasp and the object. It outputs a embedding which is used initialize the grasp node (orange in Fig 7.3) in the graph.

7.4.2 Graph Convolutional Network

We use the Graph Convolutional Network (GCN) model from Kipf and Welling [170], which is a neural network structured on the shape of the input graph[2]. By structuring a neural network to pass information between adjacent nodes, we use

[2]We more thoroughly discuss graph neural network models in Chapter 2.3.3 and explain GCN equations there. For clarity, we repeat parts of that section here.

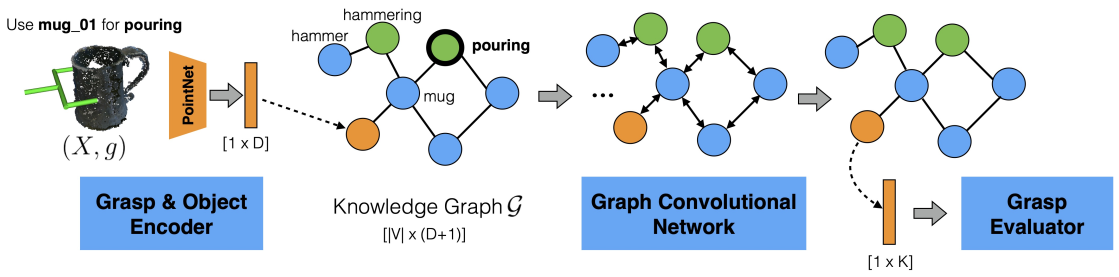


Figure 7.3: Overview of our Task-Oriented grasping framework using semantic knowledge graphs.

the input graph to correctly reason about the relationship between the object classes and the target task. The first input of a GCN is the graph itself $\mathcal{G} = (V, E)$. In our application, we use a knowledge graph constructed from two sources: the task-object class relationships in our dataset and the object hierarchy from WordNet [235]. The grasp nodes (orange in Fig 7.3) are added online to the existing knowledge graph $\mathcal{G}$ by connecting edges to the corresponding object class nodes. The graph is represented as a binary adjacency matrix A, which we normalize to obtain $\hat{A}$ following [170]. Each node of the graph also has a $D-$dimensional embedding which is used by the GCN. The target tasks are specified using an extra indicator latent variable that is concatenated with this embedding to get a vector of size $D + 1$. Except for the grasp nodes, we initialize the matrix with the word embeddings corresponding to each concept in the knowledge graph (e.g. "mug"). We use ConceptNet numberbatch [321] for the word embeddings. The embedding vectors are stacked across nodes along the first dimension to get the input matrix $\mathcal{X} \in \mathcal{R}^{|V| \times (D+1)}$. The output of the GCN are K-dimensional embeddings for each node $\mathcal{Z} \in \mathcal{R}^{|V| \times K}$. The node embeddings are propagated to their neighbours using message passing in each convolutional layer:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \tag{7.1}$$

where σ is the ReLU activation function, $H^{(0)} = \mathcal{X}$ and $H^{(L)} = \mathcal{Z}$ where L is the number of layers.

7.4.3 Grasp Evaluator

After the GCN, we are left with a node-level embedding $\mathcal{Z}$. We use the embedding corresponding to the grasp node z_g to train the final grasp evaluator $P(S|z_g)$, where S is the grasp score. This module has three fully connected layers with K units and a final sigmoid layer. The entire model, including the grasp and object encoder, GCN and grasp evaluator, is optimized with ADAM using a binary cross entropy loss.

7.4.4 Implementation Details

The point clouds were downsampled to 4096 points during training. They were also mean centered and unit-scaled. The PointNet module consists of three set abstraction layers and the number of points sampled are 512, 128 and all points. The set abstraction layers are followed by three fully connected layers with sizes [1024, 512, D]. Each set abstraction layer has three fully connected layers to learn features. The point clouds were perturbed with random rotations, jitter and dropout for data augmentation and to build robustness when testing on novel objects in unknown poses. We choose D=300 and K=128, and L=6 as the parameters for our GCN network.

7.5 Experiments

7.5.1 Zero-Shot Generalization

A central goal of both our dataset and our method is to show that we can learn task-oriented grasping models which generalize to novel objects, classes and tasks. In an ideal robotics system, we should be able to correctly grasp a novel object from a novel object class, or even grasp for a novel task. To test this, we measure our system and baselines in three different held-out test settings: held-out object instances, held-out object categories, and held-out tasks.

These held-out settings are of increasing difficulty in terms of zero-shot generalization. For each setting, we perform k-fold cross validation (k=4), such that each category (a task, object class, or object instance, based on the setting) will be held out exactly once. In each fold, grasps from 25% of the categories will be used for testing while remaining grasps will be used for training and validation.

In all experiments, we only evaluate tasks that are valid for a given input object class. This makes sense from an evaluation perspective as it separates the problem of predicting applicable tasks for objects from task-driven grasping. It also makes the comparison to methods using object-task information fair since the models do not have to decide whether the object-task pair is valid.

Evaluation Metrics

Since k-fold cross validation in any held-out setting will evaluate all grasps in the dataset, we can compute Average Precision (AP) scores for any category, i.e., any object instance, object class, or task. We then compute an mAP averaged over object instances, mAP averaged over object classes, and mAP over tasks. We show all three metrics for each of our three settings in Tables 7.2a,7.2b,7.2c, but emphasize the mAP metric that corresponds to what category is being held out.

Baselines

We compare our approach to the following models:

1. Random, which represents grasping strategies that focus on grasp stability and ignore task constraints. Results are averaged over five random seeds.
2. Semantic Grasp Network (SGN), which learns to reason about context of grasps (e.g., constraints imposed by objects and tasks) from data. This model is adapted from [202], with the difference that the input to the model is replaced with geometric embedding from our shape encoder and word embeddings of the task and the object class. Note that embeddings of tasks and object classes are both learned from training data.
3. SGN + *word embedding*, which uses ConceptNet [321] numberbatch as pretrained word embeddings for object classes and tasks.

Table 7.2: Results on TaskGrasp

(a) Object Instance Generalization

Model	**Test Performance (mAP)**		
	Instances	Classes	Tasks
Random	59.75	60.28	54.76
SGN [202]	78.51	75.08	68.80
SGN + word embedding	79.74	77.91	**74.36**
GCNGrasp (ours)	**80.25**	**77.94**	73.71

(b) Object Class Generalization

Model	**Test Performance (mAP)**		
	Instances	Classes	Tasks
Random	59.32	58.73	52.27
SGN [202]	74.20	72.95	62.55
SGN + word embedding	77.21	75.51	**63.73**
GCNGrasp (ours)	**78.81**	**76.57**	57.36

(c) Task Generalization

Model	**Test Performance (mAP)**		
	Instances	Classes	Tasks
Random	59.06	58.24	52.37
SGN [202]	75.17	71.59	63.35
SGN + word embedding	78.06	74.49	70.55
GCNGrasp (ours)	**81.50**	**79.56**	**76.01**

7.5.2 Analysis

First, to get context for our results in Table 7.2, we see that random grasp prediction achieves approximately 50-60% mean average precision, establishing a floor for the other methods. Because the number of positive and negative grasps in the dataset is about even, random guessing is able to achieve a seemingly high mAP. In a dataset with more negatives we would expect this number to be much lower.

Our method outperforms baselines in all three settings. This confirms that our method can effectively leverage the knowledge graph to generalize to novel object

Table 7.3: Ablation on Semantic Knowledge

Model	**Graph**		**Held-out Setting**		
	Nodes	Edges	Task	Class	Instance
GCN + tasks + WordNet	345	989	76.01	**76.57**	80.25
GCN + tasks	131	693	**77.54**	75.86	**81.46**
GCN + WordNet	155	106	71.77	70.00	78.66

instances, object classes, and tasks. SGN + *word embedding* also outperforms SGN, suggesting that implicit distributional knowledge provides a prior that is useful for generalization. Despite the benefit of distributional knowledge, it still only represents semantic similarities between concepts. In contrast, the knowledge graph directly stores relations between the relevant objects and tasks, and exploiting this additional structured knowledge allows our model to achieve better zero-shot generalization than SGN + *word embedding*.

When comparing our method with SGN and SGN + *word embedding*, we observe increasingly larger margins in performance from the held-out instance to the held-out class setting. As objects from different classes have more variance in terms of geometric and visual features than objects from the same class, semantic knowledge becomes more important in unifying these objects. The difference in performance between our method and these two baselines on the held-out task setting reached 12.6% and 5.46% respectively, affirming that semantic knowledge is especially crucial for generalizing disparate constraints from different tasks.

Ablations on Knowledge Graph

We investigated how performance is affected by changing the knowledge graph used in our model. Specifically, we compared the default knowledge graph with a knowledge graph containing only the semantic hierarchy of objects and a knowledge graph containing only the relations between object classes and tasks. The results from the three held-out settings are summarized in Table 7.3 (we only show the mAP metrics corresponding to the held-out category). From these results, we observe that edges between object classes and tasks were the most important knowledge for generalizing to novel tasks and instances, though every task we tested was valid for the target object class. This suggests that knowledge about which objects could generally be used for which tasks provide important information for discovering similarities

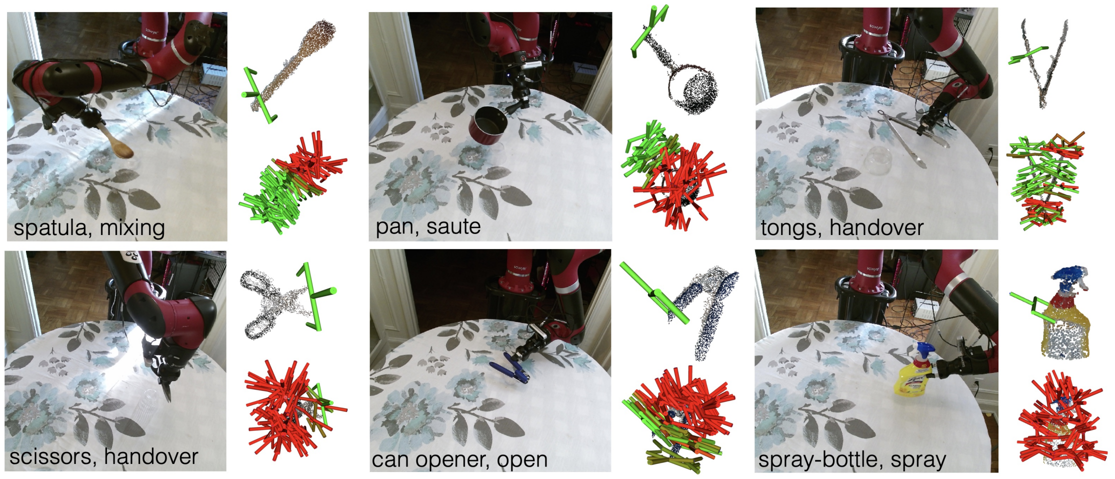


Figure 7.4: Robot executions of example task-oriented grasps on unknown objects. For each execution, the top 3D visualization shows the grasp that was executed (which had the best evaluator score) and the bottom shows all the stable grasp candidates colored by their scores (green is higher).

between tasks. In the held-out object class setting, additional knowledge from the object hierarchy helped generalize to novel object classes by associating known classes and novel classes through the WordNet hierarchy.

7.5.3 Real Robot Evaluation

We run experiments to show that our approach and dataset transfer to a real robot. We test our approach on novel objects not from the dataset and in unknown poses. We place each object (without clutter) on a table in front of the robot. After table plane segmentation to obtain the object point cloud, 600 stable grasps are sampled and 50 candidates are selected using farthest point sampling for evaluation. We evaluate the grasps on our best performing GCNGrasp model from the held-out task ablations (Table 7.2). Our hardware setup comprises of a 7-DOF Sawyer Robot with a 2-fingered Robotiq gripper and a Intel RealSense D415 RGB-D camera mounted on the gripper wrist. Inference for the 50 grasps takes around 3 seconds on a desktop with an NVIDIA GTX 1080 Ti GPU and the grasp with the best score is executed on the robot. Fig 7.4 shows the executed task-oriented grasps on unknown objects. Even though our dataset objects were collected only in one canonical pose, our approach is able to generalize to new grasps and in unknown poses due to data augmentation during training. Based on the grasp evaluator scores from Fig 7.4, our model is also able to interpolate between modes in the continuous $SE(3)$ space to reason about task-oriented grasping.

7.6 Conclusion

We present the TaskGrasp dataset to study generalization in Task-Oriented grasping. The dataset is diverse and an order of magnitude larger than previous datasets. We also present a framework for jointly learning from geometric observations and semantic knowledge to generalize to new object instances, classes and even new tasks.

In the next and final contribution chapter, we reverse the usual direction in these contributions: knowledge to end task. In the final contribution, we show how action policies can be used themselves to generate knowledge.

Chapter 8

Learning Knowledge from Actions

8.1 Introduction

We end our thesis contributions in this chapter with our investigation on learning knowledge from action in an embodied environment. In this chapter, we investigate how agents can learn to verify hypothesis/knowledge by learning a specific action policy that allows the agent to learn about its world.

Empirical research on early learning [123, 180] has shown that infants build an understanding of the world around by constantly formulating hypotheses about how some physical aspect of the world might work and then proving or disproving them through deliberate play. Through this process the child builds up an abstract consistent causal understanding (i.e. knowledge) of the world. This contrasts with the manner in which current ML systems operate. Both traditional i.i.d and interactive learning settings use a single user-specified objective function that codifies a high-level task, and the optimization routine finds the set of parameters (weights) which maximizes performance on the task. The learned representation (knowledge of how the world works) is embedded in the weights of the model, which makes it harder to inspect, hypothesize or even enforce domain constraints that might exist. On the other hand, hypothesis generation and testing is a process explored in classical approaches to AI [40]. In this work we take a modest step towards combining the reasoning mechanisms of the classical approach with modern statistical ML, with the goal of building an agent capable of testing hypotheses about its world.

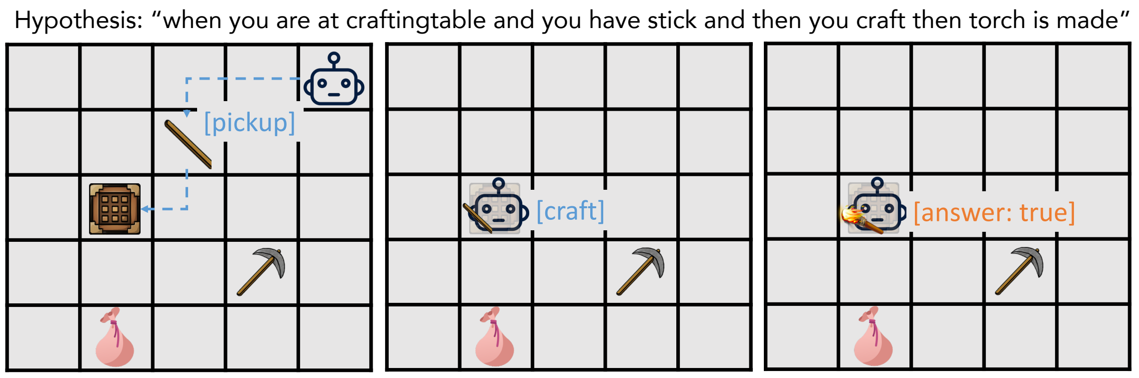

Hypothesis: "when you are at craftingtable and you have stick and then you craft then torch is made"
[pickup]
[craft]
[answer: true]

Figure 8.1: Example of a "Crafting" world. The agent must verify a given hypothesis (provided as a text string) about the causal relationships in the world. Acting according a learned policy, the agent manipulates the initial state to one that allows a predictor (also learned) to reliably determine if the hypothesis is true or not. The learning of policy and predictor is aided by a pretraining phase, during which an intermediate reward signal is provided by utilizing hypotheses that factor into {*pre-condition state*, *action sequence*, *post-condition state*}.

The problem we address is illustrated in Figure 8.1. Agents are placed in an world which has several interactive elements. They are provided with a hypothesis (an "action sentence" [265]) about the underlying mechanics of the world via a text string [1] (e.g. "$\mathcal{A}$ will be true if we do $\mathcal{B}$"). The task is to determine if the hypothesis is true or not.

This problem cannot be solved without *interaction* with a dynamic world (comparing the state before and after taking action $\mathcal{B}$).

A key novelty in our work is formulating the task in a manner that permits the application of modern RL methods, allowing raw state observations to be used rather than abstract Boolean expressions of events. To do this, we use a model composed of two different deep parametric functions which are learned through interaction: (i) an action policy that generates observations relevant to verification of the hypothesis and (ii) a prediction function which uses the observations to predict whether the hypothesis is true or false.

We first show that even in simple environments, agents trained end-to-end using

[1]We note that while the core idea of using RL for hypothesis verification is incognizant of how we specify and represent hypothesis – in this work, we use natural language as a medium since it gives flexibility over using logical relations, and is consistent with our learning-based approach.

deep reinforcement learning methods cannot learn policies that can generate observations to verify the hypothesis. To remedy this, we exploit the underlying structure of hypotheses – they can often be formulated as a triplet of a pre-condition ($\mathcal{P}$), an action sequence (collectively $\mathcal{B}$), and a post-condition ($\mathcal{A}$) that is causally related to the pre-condition and actions. Using this common structure, we are able to seed our action policy to learn behaviors which alter the truth of the pre-condition and post-condition. This allows agents to learn policies that can generate meaningful observations for training the prediction function. We further show that these policies can be adapted to learn to verify more general hypotheses that do not necessarily fit into the triplet structure.

8.2 Related Work

Knowledge representation and reasoning (KRR) [40] is a central theme of traditional AI[2]. Commonsense reasoning [82, 83, 200] approaches, e.g. CYC [190], codify everyday knowledge into a schema that permits inference and question answering. However, the underlying operations are logic-based and occur purely within the structured representation, having no mechanism for interaction with an external world. Expert systems [122] instead focus on narrow domains of knowledge, but are similarly self-contained. Logic-based planning methods [73, 109] generate abstract plans that could be regarded as action sequences for an agent. By contrast, our approach is statistical in nature, relying on reinforcement learning (RL) to guide the agent.

Our approach builds on the recent interest [119, 221] in neural-symbolic approaches that combine neural networks with symbolic representations. In particular, some recent works [209, 395] have attempted to combine RL with KRR, for tasks such as navigation and dialogue. These take the world dynamics learned by RL and make them usable in declarative form within the knowledge base, which is then used to improve the underlying RL policy. In contrast, in our approach, the role of RL is to verify a formal statement about the world. Our work also shares some similarity with [177], where ML methods are used to learn mappings from world states to

[2]See Chapter 2.2 for a more background

representations a planner can use.

Causality and RL: There are now extensive and sophisticated formalizations of (statistical) causality [265]. These provide a framework for an agent to draw conclusions about its world, and verify hypothesis as in this work. This is the approach taken in [81], where RL is used to train an agent that operates directly on a causal Bayesian network (CBN) in order to predict the results of interventions on the values on its nodes.

In contrast, the approach in this work is to sidestep this formalization entirely, with the hope of training agents who test hypotheses without building explicit CBNs. Unlike [81], our agents intervene on the actual world (where interventions make take many actions), rather than the abstract CBN. Nevertheless, we find that it is necessary to add inductive bias to the training of the agent; here we use the pretraining on ($\mathcal{P}$, $\mathcal{B}$, $\mathcal{A}$) triplets. These approaches are complementary- one could combine explicit generation and analysis of CBNs as an abstract representation of an environment with our training protocols.

Our work is thus most similar to [87], which uses reinforcement learning directly on the world, and the agent gets reward for answering questions that require experimentation. However, in that work (and in [81]) , the “question” in each world is the same; and thus while learning to interact led to higher answer accuracy, random experimental policies could still find correct answers. On the other hand, in this work, the space of questions possible for any given world is combinatorial, and random experimentation (and indeed vanilla reinforcement learning) is insufficient to answer questions.

Cognitive Development: Empirical research on early learning [123, 180] has shown that infants build an understanding of the world around them in ways that parallel the scientific process: constantly formulating hypotheses about how some physical aspect of the world might work and then proving or disproving them through deliberate play. Through this process the child builds up an abstract consistent causal understanding of the world. Violations of this understanding elicit surprise that can be measured by researchers [322].

Automated Knowledge Base completion: This work is also related to knowledge base completion [33, 104][3], especially as formulated in [288]. Instead of using

[3]See Chapter 2.6

other facts in the knowledge base or a text corpus to predict edges in the KB, here the agent needs to act in a world and observe the results of those actions. This recalls [238], where the system verifies facts it has previously hypothesized by searching for corroboration in the corpus.

Automation of the scientific process has been tried in several domains. Robotic exploration of chemical reactivity was demonstrated [127] using ML techniques. [167] developed a robot scientist that explored genomics hypotheses about yeast and experimentally tested them using laboratory automation. In biochemistry [345] used Bayesian methods for optimal experiment design. More generally, the Automated Statistician project [324] uses a Bayesian approach to reason about different hypotheses for explaining the data, with the aim of creating interpretable knowledge.

Embodied Question and Answering: The problem studied in this work is closely related to the embodied visual question answering problem in [80]. Indeed, our basic formulation is a particular case of the most general formulation of embodied QA, as the agent is rewarded for successfully answering questions about the world that require interaction. However, the form of the questions is different than those considered in that work, as they may require drawing a conclusion about the *dynamics* of the world, rather than a static property. Even the questions about static properties we are interested in have a different flavor, as they encode rules, rather than statements about the current configuration. Our approach is built around hypothesis-conclusion structure special to these questions.

There is also a large body of work on (non-embodied) visual QA [158, 364] and text-based QA [282]. From this, most relevant to our work is [365] who use a structured knowledge base to augment standard statistical QA techniques.

Language grounding: Our approach requires us to solve the language grounding problem, albeit in a simplified form due to templated language/limited vocabulary. Most other works such as [11, 59, 336] are focused on instruction following in known or unknown environments.

8.3 The Hypothesis Verification Problem

In this section we introduce the hypothesis verification problem, first informally then formally in 8.3.1.

An agent is spawned in a world sampled from a distribution over possible worlds. In the case of “Crafting”, shown in Figure 8.1, there are various items lying around that the agent can pick up and combine using a “craft” action. The exact dynamics are changed for every newly instantiated world, so in one world, taking a craft action with a stick might produce a torch, and in another, it might produce a pickaxe. At the start of each episode, the agent is given a hypothesis about the world, such as the one shown at the top of Figure 8.1. The agent gets a reward when it correctly answers if that hypothesis is true or false. Because the dynamics and rules change each episode, the agent must learn to interact with the world in order to decide if the hypothesis is true or not. In Figure 8.1, the agent picks up the stick and does a craft action to see that a torch is created. It then has enough information to determine that the hypothesis is indeed true, and the agent receives the reward for verifying the hypothesis correctly.

8.3.1 Formal Definition

Here we formally introduce the problem of hypothesis verification as a Partially Observable Markov Decision Process (POMDP).

We first define a world as a set of states and actions with Markovian dynamics; that is, an MDP without a notion of reward. We define an environment $\mathcal{E}$ as a distribution over a set of worlds $\mathcal{W}$ and hypotheses $\mathcal{H}$. A world $W \in \mathcal{W}$ is specified by rules L which describe the dynamics of the world; and data C which specifies other elements of the configuration of the world. So for example, in the crafting environment, L might consist of the crafting rules, and C might be an initial placement of the agent and objects. Thus each world in the environment can be written $W = \{C, L\}$. A hypothesis is a function from $\mathcal{W}$ to {true, false}; in this work they will be generated via templated language and depend on L. For each environment, all worlds will have the same action space A (e.g. $\{\mathsf{up}, \mathsf{down}, \mathsf{left}, \mathsf{right}, \mathsf{pickup}, \mathsf{craft}, \mathsf{true}, \mathsf{false}\}$ for Crafting), although they may not have the same sets of states. The special actions true, and false will be described in the next paragraph. Figure 8.1 shows a particular Crafting world W and hypothesis h.

Now $\mathcal{E}$ is an episodic POMDP where each episode consists of sampling a W and

h. The episode ends when the agent executes either the true or false action[4]. Given a world W and hypothesis h, an agent gets reward

$$R_{Hyp} = \begin{cases} +1 & a = h(W) \\ -1 & a = \neg h(W) \\ 0 & otherwise \end{cases}$$

8.4 Methodology

8.4.1 RL baseline

Given the formulation as a POMDP, we could try to solve it with an RL agent that has $a = \pi(S, h)$, where $S_t = \{s_t, s_{t-1}, \dots s_{t-K}\}$, s is an observation of the current world, K is a history window size, and h is the hypothesis for the current world. Because we have access to the ground truth for whether the hypothesis is true at the end of the episode through the reward, we can train a network with supervised learning to predict true and false. We define our prediction network $f(S, h)$ which takes in the last K observed observations of the environment and the hypothesis and predicts whether or not the hypothesis is true.

Despite its simplicity, we find that training such an agent from the environment reward to be challenging. In order determine whether a hypothesis is true or not, we need to take the correct sequence of actions related to the hypothesis. But in order to know that a particular sequence of actions was the right one, we need to be able to correctly predict the hypothesis. Guessing with no information gives a zero average reward, and until it learns good predictor, there is no signal to guide the policy. In practice, as shown in Figure 8.5, an RL baseline is not able to solve the task, since it can neither learn the right policy nor the right predictor to verify the hypothesis.

8.4.2 Pretraining using Triplet Hypotheses

In light of the difficulties directly training an RL model using terminal reward alone, we take advantage of the fact that many causal statements about the world have the

[4]To avoid confusion, when we refer to the action of the agent deciding if the hypothesis is true or false, we will refer to those in bold as true or false.

form:

$$(\textit{pre-condition}, \textit{action sequence}) \implies \textit{post-condition}$$

This structure can be converted into a reward function that can be used to pretrain the agent policy π. The idea is to reward the agent for taking actions which alter the truth of the pre-condition and post-condition (i.e. changing the world state so that pre/post-conditions are met or not). If it matches the pre-condition state and takes the action, if the statement is true, the post-condition should toggle from false to true in the world. Similarly, if post-condition changes but the pre-condition did not change, the statement must be false. This can be formalized in the following reward function to encourage the agent to toggle pre-condition and post-condition states:

$$R_{\mathsf{pre}} = \begin{cases} +C & a = \mathsf{stop}\ \& \\ & \text{pre-condition changed in last } K \text{ steps} \\ 0 & \textit{otherwise} \end{cases}$$

$$R_{\mathsf{pre+post}} = \begin{cases} +C & a = \mathsf{stop}\ \&\ \text{post+pre-condition} \\ & \text{changed in last } K \text{ steps} \\ 0 & \textit{otherwise} \end{cases}$$

This encourages the policy π to change the pre-condition and post-conditions (via pre-condition) in the last K steps, so that a predictor looking at the last K observations will be able to deduce the truth value of the hypothesis. More generally, training with this reward function forces the policy network to ground text concepts (e.g. the text "stick" means [object_stick]) and also capture the causal rules within the world. Consequently, following pretraining, the policy network can then be fine-tuned using the original reward function R_{Hyp}. Furthermore, since the policy network is no longer random, a robust prediction network f can also be learned. While not all hypotheses fit into the triplet format, we show in the experiments that the knowledge captured by the policy and prediction networks during this phase of training can generalize to less structured forms of hypotheses. We have a set of hypotheses for each world that contains only these triplet-structured hypotheses. We use this set for our pretraining.

8.4.3 Training using Triplet Hypothesis

After pretraining the policy π, we then further train π, as well as the prediction network f using the same set of triplet hypotheses, but now using R_{Hyp} instead of R_{pre} or $R_{\mathsf{pre+post}}$. Two variants are explored: (i) "fixed" – keep π fixed but train the prediction network f and (ii) "finetune" – finetune π and train f. Performance in this phase is used to select promising models for subsequent stages of training. Specifically, runs achieving less than 90% accuracy are eliminated.

8.4.4 Adaptation to Non-triplet Hypotheses

Next, we want to show that we can adapt our networks to hypotheses other than those that fall neatly into the triplet structure. To adapt to the larger set of hypotheses, we start with the networks from Section 8.4.3. During this training stage, the triplet-form constraint is relaxed and training proceeds with both triplet and non-triplet hypotheses (see Section 8.5.2), using an even split between the two types.

8.4.5 Network Architecture

Although other works such as [59] have investigated language-conditioned RL (usually in the form of instruction following), our hypothesis conditioned problem proved to be challenging, and required some novelty in network architectures. Figure 8.2 shows our network diagrams.

For the policy network, it was important to use key-value attention. That is: the hypothesis is fed into a seq2vec model and is used as the *key* of a dot-product attention mechanism. The state (the grid locations of the items in the world and the inventory of the agent if applicable) is fed as input to N parallel MLPs. The output of the MLPs are then fed as the *values* of the attention mechanism. The output of the module is then fed into the final hidden layer of the actor-critic network.

For the prediction network, we use the popular transformer architecture [347]. Our prediction network encodes both the hypothesis and past observations (after they are passed through a one layer network) using transformer encoders. These sequences are then combined using a transformer to generate a final hidden state as output which is then fed to a final prediction layer and sigmoid function to get our

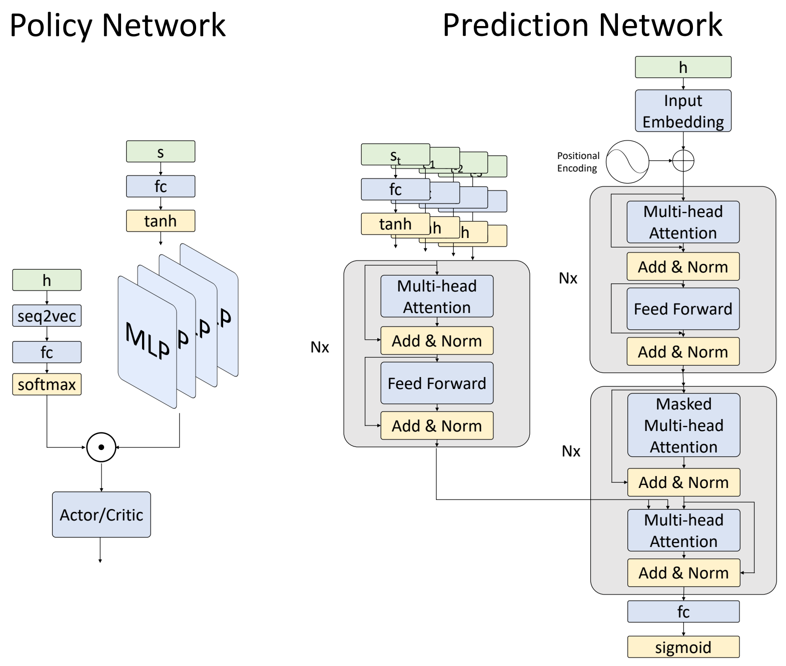


Figure 8.2: Network architecture for our policy network (left) and prediction network (right)

binary prediction.

8.5 Evaluation Environments

In this section we describe the environments we use to evaluate the approach from Section 8.4. The environments E are designed so that (a) the prior probability $p(h = true) = 0.5$ and (b) the initial state s_0 does not contain information about h.

8.5.1 Environments

We created four different environments for hypothesis verification. ColorSwitch, Pushblock and Crafting are all gridworld-based environments. A fourth enviornment is created by adapting the standard Cartpole task to include interactive elements. See Figure 8.3.

ColorSwitch

The agent is placed in a world with one or more color switches which are randomly either "on" or "off" and a door which is either open or closed. The agent is able to move and toggle the switch positions. One of the switches in the world, when in the correct position (can be either on or off) will cause the door to open. The other switches have no effect. Hypotheses in this environment relate to the color and position of switches and how that opens or closes the door.

Pushblock

The agent is placed in a world with a block which can be pushed by the agent, and a door. The agent can move and push on the block. The door opens when the block is in a particular part of the grid: "up" – top two rows, "down" – bottom two rows, "left" – leftmost two rows, "right" – rightmost two rows. The hypotheses in this environment related to the position of the pushblock and how that affects the door.

Crafting

The agent is placed in a world with crafting rules similar to that of the popular Minecraft game. The agent is spawned along with a number of crafting items, and a crafting location. The agent is able to move, pick up items into its inventory and use the crafting location using special crafting actions. There is some true "recipe" which produces some new item in the agent's inventory.

Cartpole

This is the standard classic control cartpole problem where a pole is attached by an un-actuated joint to a cart. The regular actions are "left" and "right" and if the pole falls, the episode ends. In our modification, there are "zones" (an interval on the x axis) where the physical laws of the cartpole change by either changing the gravity constant, or applying a "wind force" blowing the cart in one direction. Like in ColorSwitch, the zones are specified by color. Typically one color zone has an effect and the other is a decoy zone that has no effect. The hypotheses related to which color zones correspond to what changes to the physics.

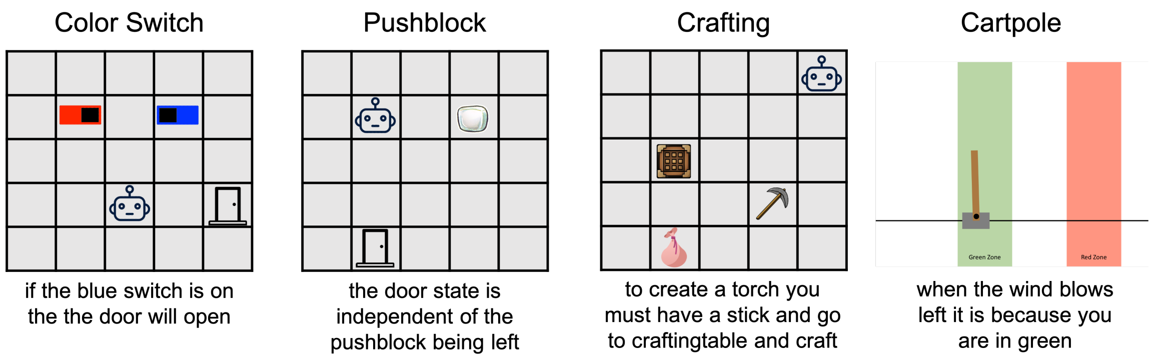

Color Switch
Pushblock
Crafting
Cartpole
Green Zone
Red Zone
if the blue switch is on the the door will open
the door state is independent of the pushblock being left
to create a torch you must have a stick and go to craftingtable and craft
when the wind blows left it is because you are in green

Figure 8.3: Examples of the four environments used in our experiements: Color Switch, Pushblock, Crafting and Cartpole.

In the grid world environments, items are randomly generated in a 5 by 5 grid. The world observation is given by a 1-hot vector for each grid location and inventory. The hypothesis is encoded as sequence of tokens. In Cartpole the state is the standard dynamics as well as a 1-hot vector specifying the location and color of the zones.

8.5.2 Hypothesis Construction

We now describe how the hypotheses for each world in each environment are automatically generated via templates. Three different varieties are considered:

1. Triplet Hypotheses
2. General Templates
3. Special Case Templates[5]

Triplet hypotheses

Here the hypotheses takes the form of an explicit logical statement: (pre-condition, action sequence) $\implies$ post-condition. When the pre-condition is true, and the action sequence is performed, the post-condition will become true. To generate triplet hypotheses, we:

1. Randomly select a pre-condition template from a set list

[5]See appendix of the original paper [225] for all of the possible templates for an environment and further details.

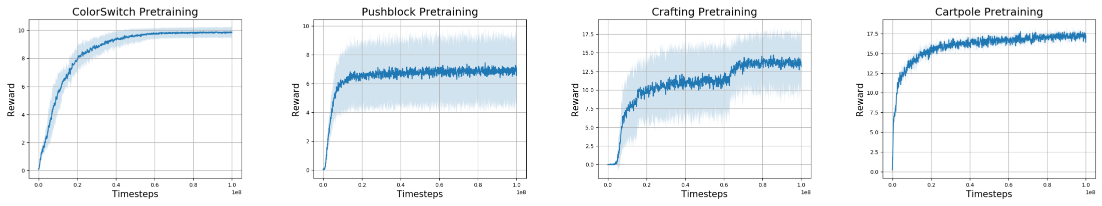


Figure 8.4: Pretraining reward ($R_{\mathsf{pre}} + R_{\mathsf{pre+post}}$) on our four environments.
.

2. Randomly select an action template
3. Randomly select a post-condition template
4. Fill in any entities in the final template

In our example from Figure 8.1 this would be ("when you are at crafting table and you have stick"; "and then you craft"; "then torch is made").

General templates

Instead of drawing a template from the triplet form, a single template for the hypothesis is drawn and the values populated. For instance, in Pushblock, a template might be "the door can only be opened when the pushblock is PUSHBLOCK_POSITION" and then "left" would be drawn for PUSHBLOCK_POSITION. These templates are more general than the triplet ones in that they have no explicit (pre-condition, action sequence and post-condition) structure.

Special cases

We also use more difficult and general hypothesis templates. These cannot be neatly fit into a triplet format by rewording, and may not fully describe the rules of the world. Some examples of these harder templates are: negating effects (e.g. "door is not open"), negating conditions (e.g. "switch is not on") and independence (e.g. "door independent of blue switch").

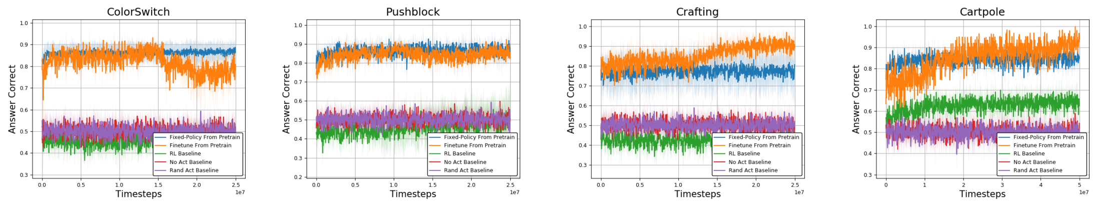


Figure 8.5: Hypothesis prediction accuracy on *both* triplet *and* non-triplet hypotheses for the Color Switch, Pushblock, Crafting and Cartpole environments, using R_{Hyp} reward for training.

8.6 Experiments

Figure 8.4 shows results from learning with pretraining rewards $R_{\mathsf{pre+post}}$ and R_{post}. There is relatively little variance, with all runs achieving near the theoretical maximal rewards. Note that for Pushblock, sometimes the block can be stuck against a wall, so not all worlds are solvable. For Crafting and Cartpole, $R_{\mathsf{pre+post}}$ is not always achievable if true and distractor items are far away from each other. For the three gridworld environments we show results and variance runs on 25 random seeds. For Cartpole, we show results on 5 random seeds.

In Figure 8.5, we show the results on non-triplet adaptation (Section 8.4.4). As discussed in Section 8.5.2, this stage includes the more difficult, non-triplet templates not seen during pretraining or during triplet hypothesis training. We also break down the final hypothesis prediction accuracy for our methods in Table 8.1. This allows us to see whether our methods were able to adapt to non-triplet hypotheses.

RL Baseline

Figure 8.5 shows the RL baseline at chance-level performance, the only exception being Cartpole were it achieves $\sim 60\%$, versus the $\sim 90\%$ of our approaches. This poor performance is not surprising given the difficulty of training both policy and predictor from scratch.

Other Baselines

We also include two other simple baselines "no act" and "random act." The "no act" baseline simply takes the `stop` action at $t = 0$, forcing the prediction network to give

Table 8.1: Average Hypothesis Prediction scores, broken down by triplet (pretrained) and non-triplet (not seen in pretraining)

	Method	**Overall**	**Triplet Acc.**	**Non-triplet Acc.**
Color Switch	Fixed Policy	86.6%	91.1%	82.1%
	Finetuned Policy	77.5%	79.7%	75.4%
Pushblock	Fixed Policy	86.9%	87.9%	85.9%
	Finetuned Policy	85.6%	86.3%	84.8%
Crafting	Fixed Policy	77.3%	92.8%	61.8%
	Finetuned Policy	90.7%	98.4%	83.0%
Cartpole	Fixed Policy	84.2%	92.0%	76.3%
	Finetuned Policy	92.5%	93.4%	91.6%

an answer from just the first observation. This fails because the agent needs to take actions in the world to be able to predict the hypothesis accurately. For "random act", a random policy is used (i.e. uniform distribution across actions). This similarly fails as random actions are extremely unlikely to yield observations that allow for the verification of the hypothesis. This also confirms that we do not accidentally leak the true value of h into the initial state of the world.

Discussion

From these results, we can clearly see that naive RL and other baselines cannot efficiently solve hypothesis verification tasks. When we use our pretraining method, we use the insight that hypotheses often have a clear causal structure that can be exploited when they are formed as "triplet hypotheses." Not all hypotheses fall neatly into this form, and we may not have this form for all hypotheses. But if we have some that fit this form, we can gain a foothold that lets us make progress on this problem, and then later adapt them to other hypotheses. From Figure 8.5 we can see that this pretraining, triplet training and adaptation works. We also show in Section 8.6.1 that our choice of pretraining task was not arbitrary; other plausible pretraining rewards fail to achieve the same results as our method.

Looking at the different environments, we see that in Pushblock and Color Switch, even with the policy learned from the triplet pre/post reward, the agent is able to generalize and perform well on hypotheses not seen in the pretraining phase as we can see in Table 8.1.

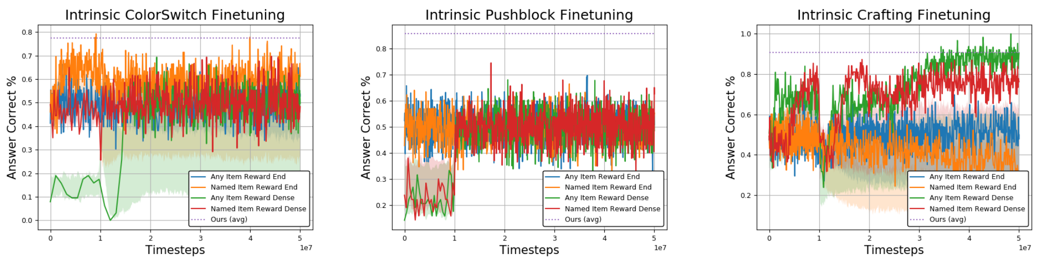


Figure 8.6: Final hypothesis accuracies using alternate forms of intrinsic pretraining versus our pretraining (purple). "End" = reward only received at end of episode. "Dense" = reward received immediately after changing item state.

In Crafting and Cartpole on the other hand, to do well on the unseen templates, the policy also needs to be fine-tuned. This tells us that when we do have to generalize to hypotheses we have not seen before, and even ones that differ greatly from our triplet hypotheses, adapting the policy as well as the prediction network is necessary. Recall that in our evaluation environments, we test very different hypotheses such as as negations and "independence" hypotheses not see in triplets. We see from Table 8.1 that indeed, our finetuned policies greatly outperformed the fixed policies on the non-triplet templates.

8.6.1 Alternate Forms of Pretraining

As an ablation, we test four variants of an "intrinsic" proxy reward to see if other pretraining schemes might perform equally well. We show results on the gridworld domains using 4 different intrinstic forms of pretraining:

1. Change any item state in the world; receive reward at end of episode
2. Change any item referenced in the hypothesis; receive reward at end of episode
3. Change any item state in the world; receive reward instantaneously
4. Change any item[6] referenced in the hypothesis; receive reward instantaneously.

In Figure 8.6 we show the accuracies on the final hypothesis verification task (from Section 8.4.4) for both triplet and non-triplet hypotheses, using the four intrinsic pretraining methods. We also plot the final average accuracy obtained by our adapted

[6]Here, "item" means any object that is not the agent (including crafting items, switches, pushblocks, etc.).

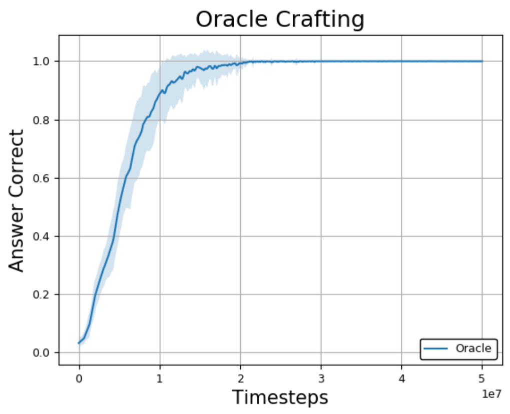


Figure 8.7: Results on training an RL using R_{Hyp} with oracle predictor on the Crafting environment. Mean and variance on 25 random seeds are shown.

methods from Figure 8.5. For the intrinsic pretrained policies the best run (from 25 seeds) is shown to show the best-possible case of the alternative methods.

For Crafting the dense intrinsic pretraining works about as well as ours. This can be explained by the fact that this particular form intrinsic pretraining directly rewards the agent for doing many of the operations in the actual Crafting task, i.e. picking up objects and crafting objects. However, averaging across the three environments, all the intrinsic pretraining methods do worse than our approach, showing the merits of our pretraining approach which exploits structure common to many hypotheses, yield an effective and general form of pretraining.

8.6.2 Oracle Predictor Ablation

In this experiment, we provide an “oracle” hypothesis predictor on the Crafting environment that, given the last K states of the world, will output the ground truth of the hypothesis if it is inferable from those frames. We then explore if RL is now able to learn directly using reward R_{Hyp}. First, we train a RL agent with access to the oracle. The RL agent must learn its action policy, but when it takes the stop action, the oracle is used to predict the hypothesis. Therefore, if the actions it has taken yield observations that permit verification of the hypothesis, it will automatically answer correctly and get the reward.

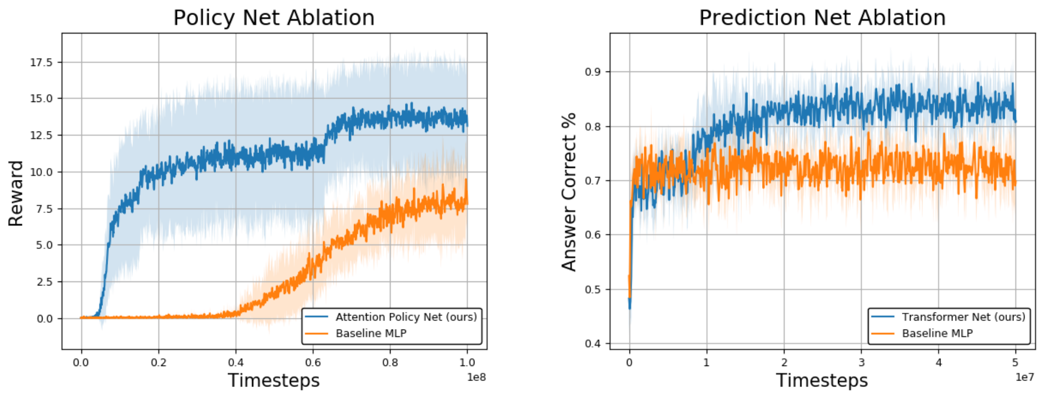


Figure 8.8: (left) policy network ablation (right) prediction network ablation.

8.6.3 Network Ablation

Figure 8.7 shows the oracle-equipped agent quickly converging on perfect performance. From this we surmise that if we know how to predict the hypothesis already, learning a policy that makes the oracle prediction possible is relatively straightforward. Mindful of the inability of the RL baseline (lacking an oracle) to converge to a good solution, this shows the joint nature of the problem (i.e. learning both policy and predictor) is what makes it challenging.

In Figure 8.8 we see the results of our network architecture ablation. As we can see, our new policy architecture described in Section 8.4.5 clearly outperforms a standard MLP policy network on the language-condition pretraining task. We also see that the transformer architecture outperforms the LSTM and MLP model on the final task when we hold the policy network constant.

8.7 Conclusion

In this work, we propose a tractable formulation of the problem of training agents that can interact with a world to test hypotheses. We show that generic RL techniques struggle with the problem, but by using the common structure of some hypotheses, we are able to develop a method that works in simple environments. Specifically, we use the fact that many hypotheses can be broken into triples of the form of (pre-condition, action sequence, post-condition). We also show that once pretrained using this factorization, agents can be adapted to verify more general hypotheses.

With this contribution, we have begun completing the loop and discussed techniques that use knowledge for downstream embodied tasks (vision, VQA, procedural crafting, robotic grasping) and now how agents can learn knowledge from its surroundings.

One major limitation of this work is the environment on which it was tested. Although our environments were sufficient to allow us to formulate the problem for RL and to identify early challenges in solving it, future work would need to move beyond these simple environments. First, our approach in this paper achieves around 90% accuracy on these tasks already. But more importantly, the "richness" of these environments limits what we can test. Here richness encompasses the never-ending list of attributes, properties, laws and causal relationships that make up the world. It includes things such as richness of perception (what exact shade of blue is this item, what is the texture, is it soft or hard), physics (what happens to an object when you drop it, can this object break), and more indirect causal relationships (when the light is turned off how does that affect our ability to view objects at a distance). You could not run a never-ending learning [238] system here because it would run out of new things to discover.

So to truly have a never-ending agent which can learn things endlessly (or at least for a very long time), we really need more rich environments. One way to do this would be to either use a rich simulator [2, 175, 367] or set up a robot platform in a setting with a variety of objects and tasks to perform. However, once we have added this additional richness, we will still run into exactly the same problem we faced in Section 8.4.1 that an initial random policy will not provide enough of a reward signal. In fact, this problem is much worse because these more realistic environments will actually be much more difficult than the ones presented in this chapter. In order to do knowledge acquisition, we will also need to have agents have some prior prediction or policy ability to properly do the exploration. And because this environment becomes more realistic, our works on using knowledge for vision, language, and action will become more relevant and useful. This gives us the opportunity to fully close the loop: to learn knowledge about the world and then use that knowledge to perform action, perception and language tasks.

In our final conclusion to this thesis, we will discuss in detail how we might begin to set up these more realistic environments, and how we might then be able to combine knowledge-learning from this chapter with the use of knowledge in vision, language and action as we did in previous chapters.

Chapter 9

Conclusions

9.1 Summary of Contributions

Throughout this thesis, we have examined the role that knowledge plays in embodied AI, in vision, language and action and introduced our contributions in these areas. First we introduced our contribution to using knowledge for vision. We showed that we were able to improve performance on multi-label image classification by exploiting knowledge from our graph relating various visual categories that appear in the dataset. In the next contributions we looked at using knowledge for vision and language, first by introducing a benchmark dataset called Outside Knowledge Visual Question Answering (OK-VQA) which provides a benchmark for works which incorporate outside knowledge into a joint recognition and question answering problem. We then and present KRISP to integrate knowledge graphs and large language models models to cover the wide variety of knowledge needed to solve these knowledge-based questions.

We then examined the action modality. We exploited the knowledge from language and showed that by training our model to mimic the natural language instructions of humans, we can learn long-trajectory crafting tasks and generalize to unseen tasks in a zero-shot setting and to learn quickly from a few demonstrations. We then demonstrated that by incorporating knowledge graphs, our GCNGrasp framework can generalize to new object instances, classes and even new tasks for task oriented grasping. Finally, we formulated “hypothesis verification” as a reinforcement learning problem that trains agents to ground proposed knowledge about their world.

9.2 Future Directions: Bringing it all Together

Before we conclude, we would like to sketch out where we think our ideas on knowledge in embodied agents might take us. In this thesis, we have shown contributions where we have used knowledge for a downstream task: image classification, VQA, RL, robotic grasping and shown places where it helps us join modalities such as vision and language, language and action, and action in vision. We have even shown how we might use an agent's action policy to learn knowledge from its environment. In some sense, what we have done is show a number of case studies in these modalities.

But one thing that is so far missing is a universal system that joins all of these things together. At the beginning of this thesis, we imagined a robot which had a wide number of capabilities. It could speak to humans, do a wide variety of household chores, it could make plans and it could use knowledge. We still believe this is the right kind of goal to aim for as a community. We want to show that knowledge is not only useful for the specific end tasks we have demonstrated, but that knowledge in fact allows for a unified system to do all of these tasks. We showed that an agent can learn knowledge from its environment, but to reach human capabilities of knowledge, we want an agent that can both gather knowledge, combine it with prior knowledge and then apply it again to its environment.

To do this, we really need to address two things. First, what is the right environment or test-bed for the confluence of these ideas. Our end goal is somewhat clear (a robot that can do anything), but in research, we must always find the next stepping stone on the way to that goal. So what is both bringing in these modalities and bringing in ideas of both creating and using knowledge, but is more achievable in a shorter time frame. Second, what is the right task setup or evaluation setup? What are the immediate goals to accomplish in this environment and how can we know that we are successful. Again, what is the right balance between moving in the direction we want and being achievable in a reasonable time-frame?

9.2.1 Environments

At the end of Chapter 8, we discussed the limitations of our environments. Specifically we talked about the lack of richness. In a rich environment, there are a near infinite number of things that an agent can know or learn about. This includes a variety of different objects in its environment, diversity in sensory observations (color, texture, sound, feel, visibility), and diversity in causal relationships, rules and world physics (gravity, wind, breakability, weight, affordance, movability). The more rich an environment is, the more things an agent is able to do in it, the more knowledge there is to learn and use and the more difficult the potential range of tasks is. In order for us to test our knowledge agent and to give it sufficient things to learn, we require a rich environment.

The first path we might conceivably go down is the simulator path. The idea would be to make use of or create a simulated environment that has richness in enough of the areas you care about to give agents enough opportunity to learn and perform tasks. The choice of environment here is critical to the kinds of richness you would be able to make use of. On one end you have environments which try to maximize visual or physics-related realism of simulated robotics platforms such as the IKEA Furniture Assembly Environment [188], SAPIEN [368], Flightmare [319] or RLBench [148]. On the other end of the spectrum, you have environments which sacrifice in either visual or physics realism or richness for more semantic or causal richness. For instance MineRL [132] based on the Minecraft video game, AndroidEnv [343] where the actions are the interactions with a simulated smartphone and observations are the display, TextWorld [75], a collection of text-adventure games where actions and observations are English words and sentences and NetHack [181] which is a simple ASCII observation space overlaid on a text-adventure game setup. In between these extremes you have environments such as the Interactive Intelligence Playroom [2], Habitat 2.0 [332], AI2 THOR[175] and Interactive Gibson [367] which to different degrees trade off visual realism, precise physics and action spaces and semantically rich worlds.

This approach has many positives. It is very controllable, which would allow for the precise creation of training and testing scenarios. Relative to true robotics setup, it is far more convenient, cheap, and allows for greater reproducibility by others. Less positively, it does likely sacrifice richness in at least one dimension. Some such as TextWorld have no physics or visual observations while physics-based environments often have very limited spaces for semantic exploration. To try to deal with this, we would want to have a strong notion of introducing "new richness" to the environment or to possible knowledge continuously over time. For instance, you might create an environment where every once in a while you could add a new object, or new types of hypotheses to have new things for the agents to discover.

The other possible direction one could take would be to actually have the environment be a real environment such as a lab or home that a mobile robot could explore and interact with. This does not completely solve the richness problem, since the environment is still limited by the rooms and objects you have in that testing space, but allows for all kinds of richness about the world that might otherwise be difficult to program. One major hurdle with this approach, however, is that you really need to have a strong enough initialization for the robot to really allow it to do these explorations, not unlike how in Chapter 8 policy initialization was the major challenge, even in simpler environments. To really allow robots that can explore and test hypotheses, they need to already be able to have some abilities to act in the world.

9.2.2 Every-task Learning

Whether the environment is a simulated one or a real robot platform, the next question is what our agents would actually do and how we evaluate them. Our suggestion is not only to set up agents for multi-task learning, but for every-task learning.

The first step is to set up an environment (either in simulation or in situ with a robot) with a wide array of different objects and different tasks possible for the agent to do. As an example, in Chapter 7 we designed a semantic grasping dataset including 56 final tasks that the robot would have to perform with that object. In that work, we only considered the initial grasp, but ideally we would want to have the agent actually perform all of those tasks. What these "low-level tasks" are here would depend on the environment we have chosen. The next level up to consider might be what you might call "high-level tasks." We make this distinction in Chapter 5 between high-level and low-level. A low-level task might be to crack an egg while a high-level task would be “make an omelette.” To set up our task, we should specify, or have annotators specify a large number of these high-level tasks that can be accomplished in the environment for training and evaluating our agents. Ideally our environment should be sufficiently rich that the space of possible tasks that can be accomplished should be functionally infinite.

At the same time, because we want to also be learning knowledge about the world; the world should also be set up such that rich knowledge about the world can be learned or known prior to the task. This can take many forms, from the commonsense knowledge from sources such as ConceceptNet [200] which we used in Chapter 5, affordances and what things can be done with which objects such as in Chapter 7, and even implicit knowledge about how a high-level task can be performed with low-level actions such as in Chapter 6.

With our tasks and knowledge set, as we discussed earlier, at the beginning of training, we provide our agent with the ability to perform a number of low-level skills, and prior knowledge about the world. Now what does our agent do, and how is it evaluated.?Our proposal is to set aside a large number of annotations for high-level and low-level skills and knowledge the agent should learn. We use some of these to train our agents and then use some for evaluation. The idea here is to build a coherent multi-task agent. It works in both directions of knowledge, learning from our set aside training tasks to both accomplish end tasks using its prior knowledge and learning to verify new knowledge in the environment.

How do we get an agent that has a large variety of skills like cracking eggs or picking things up and putting them down? One challenge is that besides the challenge of actually doing those low-level policies, how do you organize them? Knowledge can give you a way of figure out in the first place what were these skills that we actually needed to learn. With semantic knowledge, we can give names to these sub-policies, and by being able to name them, we also get knowledge from language (as we saw in Chapter 6) and from explicitly structured knowledge. The knowledge helps to figure out how you would explore a new object and where you would go. The knowledge also helps (as we’ve seen in Chapter 7) by providing a way to learn generalization between related tasks and object.

Obviously, this is just a sketch of the ways we and other researchers might go about building a fully-integrated knowledge agent. Missing from this are several key details and research questions. This includes:

1. What neural architecture or what other kinds of models does our learning agent use? How does it structure memory? How does it represent its observations?
2. How does our agent structure its knowledge? Do we use graphs, raw language, or some other representation?
3. How do we train our agents? Do we take a pure reward approach or perhaps use a mixture of supervised, unsupervised and reinforcement learning methods?
4. How do we collect a series of tasks and evaluations that both gives the agent enough to show that it can learn, but also does not trivialize the problem?

These and other questions remain to be solved. But as we said from the start, the first step in knowing which way to go is knowing where we want to get to.

9.3 Final Thoughts

In this thesis, we have shown ways knowledge can be used and learned in many different modalities and argued that knowledge needs to seen as a key part of AI going forward. We have done a number of case studies to show that background knowledge can be helpful in vision, language and action and that agents can learn knowledge about the world. We, and the many many others who work in this sub-field, have started laying the pieces on the board. But the long-term, even if it is 10 years or 20 or even a lifetime away, needs to be agents that can do everything and know anything. As the field moves forward, we hope that community moves towards understanding knowledge as a fundamental part of cognition and that we continue to develop our ideas and methods towards knowledge-capable AI. We hope that this thesis can be even a small part of arriving at that goal.

Bibliography

[1] Wikipedia: The free encyclopedia. https://www.wikipedia.org/. 3, 17, 24, 27, 85, 88

[2] Josh Abramson, Arun Ahuja, Iain Barr, Arthur Brussee, Federico Carnevale, Mary Cassin, Rachita Chhaparia, Stephen Clark, Bogdan Damoc, Andrew Dudzik, et al. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*, 2020. 157, 161

[3] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Devi Parikh, and Dhruv Batra. VQA: visual question answering. *IJCV*, 2017. 66, 81

[4] Alan Akbik and Thilo Michael. The Weltmodell: A data-driven commonsense knowledge base. In *LREC*, volume 2, page 5. Citeseer, 2014. 30

[5] Chris Alberti, Jeffrey Ling, Michael Collins, and David Reitter. Fusion of detected objects in text for visual question answering. In *EMNLP*, 2019. 37, 83

[6] Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial neural networks: concept learning*, pages 102–111. 1990. 20

[7] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021. 22

[8] Prithviraj Ammanabrolu and Mark Riedl. Playing text-adventure games with graph-based deep reinforcement learning. In *NAACL-HLT*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 42

[9] Prithviraj Ammanabrolu and Mark O. Riedl. Transfer in deep reinforcement learning using knowledge graphs. In *EMNLP*, 2019. 42

[10] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, pages 6077–6086, 2018. 86

[11] 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 *CVPR*, pages 3674–3683, 2018. 40, 105, 143

[12] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016. 65, 83

[13] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, pages 166–175. JMLR. org, 2017. 40, 101, 103, 106

[14] Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *NAACL-HLT*, pages 2166–2179, 2018. 40, 104

[15] Laura Antanas, Plinio Moreno, Marion Neumann, Rui Pimentel de Figueiredo, Kristian Kersting, José Santos-Victor, and Luc De Raedt. Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach. *Autonomous Robots*, pages 1–26, 2018. 42, 125, 126

[16] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. In *ICCV*, 2015. 38, 65, 66, 67, 75, 81, 84

[17] Paola Ardón, Èric Pairet, Ronald PA Petrick, Subramanian Ramamoorthy, and Katrin S Lohan. Learning grasp affordance reasoning through semantic relations. *IEEE Robotics and Automation Letters*, 4(4):4571–4578, 2019. 42, 125, 126

[18] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. 25, 83, 88

[19] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021. 22

[20] Ivana Balažević, Carl Allen, and Timothy M Hospedales. TuckER: Tensor factorization for knowledge graph completion. *EMNLP-IJCNLP*, 2019. 34

[21] Hedi Ben-younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. MUTAN: Multimodal tucker fusion for visual question answering. In *ICCV*, 2017. 72, 73, 75, 93

[22] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003. 17

[23] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013. 34, 66, 84

[24] Sumithra Bhakthavatsalam, Kyle Richardson, Niket Tandon, and Peter Clark. Do dogs have whiskers? a new knowledge base of haspart relations. *arXiv preprint arXiv:2006.07510*, 2020. 26, 83, 84, 88

[25] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks. 2019. 22

[26] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional ARMA filters. *TPAMI*, 2021. 22

[27] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020. 35

[28] Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A. Knepper, and Yoav Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, 2019. 41

[29] Valts Blukis, Ross A. Knepper, and Yoav Artzi. Few-shot object grounding and mapping for natural language robot instruction following. *arXiv preprint arXiv:2011.07384*, abs/2011.07384, 2020. 40

[30] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017. ISSN 2307-387X. 17

[31] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008. 26

[32] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, volume 25, 2011. 33

[33] Antoine Bordes, Nicolas Usunier, Alberto Garca-Durn, Jason Weston, and Oksana Yakhnenko. Irreflexive and hierarchical relations as translations. *arXiv preprint arXiv: 1304.7158*, 2013. 142

[34] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *EMNLP*, 2014. 34, 66, 84

[35] Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *ICDM*, 2005. 23

[36] Ch Borst, Max Fischer, and Gerd Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *ICRA*, volume 1, pages 319–325. IEEE, 2004. 125

[37] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, A. Çelikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In *ACL*, 2019. 4, 19, 27, 31

[38] Adrian Boteanu, David Kent, Anahita Mohseni-Kabir, Charles Rich, and Sonia Chernova. Towards robot adaptability in new situations. In *AAAI Fall Symposium Series*, 2015. 42

[39] Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. Inducing relational knowledge from BERT. In *AAAI*, volume 34, pages 7456–7463, 2020. 4, 19, 27, 31

[40] Ronald J. Brachman and Hector J. Levesque. Knowledge representation and reasoning. *Annual Review of Computer Science*, 1(1):255–287, 1986. 14, 139, 141

[41] Ronald J Brachman and Brian C Smith. Special issue on knowledge representation. *ACM SIGART Bulletin*, (70):1–138, 1980. 15

[42] Samarth Brahmbhatt, Cusuh Ham, Charlie Kemp, and James Hays. ContactDB: Analyzing and predicting grasp contact via thermal imaging. In *CVPR*, 2019. 124, 125, 126, 128

[43] SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. Learning high-level planning from text. In *ACL*, 2012. 42

[44] SRK Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012. 40, 105

[45] Thorsten Brants and Alex Franz. The Google web 1T 5-gram corpus version 1.1. *LDC2006T13*, 2006. 17

[46] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017. 22

[47] Encyclopaedia Britannica et al. Encyclopædia britannica, 1993. 12, 13, 14

[48] Marc Brockschmidt. GNN-FiLM: graph neural networks with feature-wise linear modulation. In *ICML*, pages 1144–1152. PMLR, 2020. 22

[49] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021. 23

[50] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 4, 19, 27, 28

[51] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014. 21, 46, 48

[52] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*, 2018. 22

[53] Tianshi Cao, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam. BabyAI++: Towards grounded-language learning beyond memorization. *arXiv*

preprint arXiv:2004.07200, 2020. 104

[54] Andrew Carlson, Justin Betteridge, Estevam Rafael Hruschka Junior, and Tom Mitchell. Coupling semi-supervised learning of categories and relations. In *NAACL-HLT*, pages 1–9, 2009. 30

[55] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010. 24, 27, 29, 30, 47, 50

[56] Lewis Carroll. *Alice's adventures in wonderland*. Macmillan, 1920. 1

[57] Joyce Y Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. Language to action: Towards interactive task learning with physical agents. In *IJCAI*, pages 2–9, 2018. 41

[58] C. Chang, Dean Huang, Danfei Xu, E. Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *ECCV*, 2020. 42

[59] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *AAAI*, 2018. 40, 105, 143, 147

[60] Ciprian Chelba, Tomas Mikolov, M. Schuster, Qi Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *INTERSPEECH*, abs/1312.3005, 2014. 29

[61] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *ACL*, 2017. 33, 34, 66, 84

[62] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011. 40, 105

[63] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, pages 1725–1735. PMLR, 2020. 22

[64] Valerie Chen, Abhinav Gupta, and Kenneth Marino. Ask your humans: Using human instructions to improve generalization in reinforcement learning. In *ICLR*, 2021. 9, 112

[65] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: extracting visual knowledge from web data. In *ICCV*, 2013. 27, 30, 47, 50, 66, 84

[66] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *CVPR*, pages 2027–2034, 2014. 30

[67] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *CVPR*, pages 7239–7248, 2018. 36

[68] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019. 37, 83

[69] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *ICLR*, 2019. 40, 104

[70] Jaemin Cho, Jiasen Lu, Dustin Schwenk, Hannaneh Hajishirzi, and Aniruddha Kembhavi. X-LXMERT: Paint, caption and answer questions with multi-modal transformers. In *EMNLP*, 2020. 37

[71] Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *arXiv preprint arXiv:1910.09451*, 2019. 40, 104

[72] John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, Jacob Andreas, John DeNero, Pieter Abbeel, and Sergey Levine. Guiding policies with language via meta-learning. In *ICLR*, 2018. 40, 104

[73] Zenon Colaco and Mohan Sridharan. Mixed logical and probabilistic reasoning for planning and explanation generation in robotics. *arXiv preprint arXiv:1508.00059*, 2015. 14, 141

[74] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020. 22

[75] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018. 41, 161

[76] National Research Council et al. *How people learn: Brain, mind, experience, and school: Expanded edition.* National Academies Press, 2000. 101

[77] Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. IKE-an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 12–17, 2016. 31

[78] Hao Dang and Peter K Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *IROS*, pages 1311–1317. IEEE, 2012. 126

[79] Angel Daruna, Weiyu Liu, Zsolt Kira, and Sonia Chetnova. RoboCSE: Robot common sense embedding. In *ICRA*, pages 9777–9783. IEEE, 2019. 27, 42

[80] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *CVPR*, 2018. 40, 65, 105, 143

[81] Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019. 32, 142

[82] Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers Inc., 1990. ISBN 1-55860-033-7. 14, 141

[83] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103, August 2015. 14, 141

[84] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *NeurIPS*, pages 271–278, 1993. 105

[85] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016. 22

[86] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 25, 36

[87] Misha Denil, Pulkit Agrawal, Tejas D Kulkarni, Tom Erez, Peter Battaglia, and Nando de Freitas. Learning to perform physics experiments via deep reinforcement learning. *arXiv preprint arXiv:1611.01843*, 2016. 32, 142

[88] Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018. 22

[89] Renaud Detry, Jeremie Papon, and Larry Matthies. Task-oriented grasping with semantic and geometric scene understanding. In *IROS*, pages 3266–3273. IEEE, 2017. 126

[90] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 4, 19, 27, 28, 37, 82, 83, 85, 86

[91] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *TPAMI*, 29(11):1944–1957, 2007. 22

[92] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Towards graph pooling by edge contraction. In *ICML Workshops*, 2019. 22

[93] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of Wikipedia: Knowledge-powered conversational agents. In *ICLR*, 2018. 6, 34

[94] Santosh Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 27, 30, 66, 84

[95] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *ICRA*, 2018. 124, 126

[96] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. Technical report, Washington Univ. Seattle Dept. of Computer Science and Engineering, 2006. 30

[97] Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017. 22

[98] Simon S Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *NeurIPS*, 2019. 23

[99] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012. 35, 48

[100] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015. 21, 46, 48

[101] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. 17

[102] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012. URL `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`. 55

[103] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2018. 105

[104] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011. 33, 142

[105] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation

from simulated self-supervision. *RSS*, 2018. 124, 125, 126, 128

[106] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 35, 48

[107] Matthias Fey. Just jump: Dynamic neighborhood aggregation in graph neural networks. *arXiv preprint arXiv:1904.04849*, 2019. 22

[108] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 22, 91

[109] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971. 14, 141

[110] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*, 2017. 105

[111] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, pages 1515–1528, 2018. 105

[112] Maxwell Forbes, Ari Holtzman, and Yejin Choi. Do neural language representations learn physical commonsense? *arXiv preprint arXiv:1908.02899*, 2019. 35

[113] Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017. 105

[114] Andrea Frome, Greg S. Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. DeViSE: A deep visual-semantic embedding model. In *NeurIPS*, 2013. 36, 126

[115] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, pages 457–468, 2016. 65, 81, 83

[116] Cipriano Galindo, Alessandro Saffiotti, Silvia Coradeschi, Pär Buschka, Juan-Antonio Fernandez-Madrigal, and Javier González. Multi-hierarchical semantic maps for mobile robotics. In *IROS*, pages 2278–2283. IEEE, 2005. 42

[117] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *NeurIPS*, 2015. 38, 65

[118] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *CVPR*, pages 6639–6648, 2019. 83

[119] Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012. 141

[120] François Gardères, Maryam Ziaeefard, Baptiste Abeloos, and Freddy Lecue. ConceptBERT: Concept-aware representation for visual question answering. In *EMNLP*, pages 489–498, 2020. 37, 84, 92, 93

[121] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer, 2003. 23

[122] Joseph C Giarratano and Gary Riley. *Expert systems*. PWS publishing co., 1998. 14, 141

[123] Alison Gopnik. Scientific thinking in young children. theoretical advances, empirical research and policy implications. *Science*, 28(337):1623–1627, Sept 2012. 139, 142

[124] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual question answering in interactive environments. In *CVPR*, 2017. 65

[125] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. *IEEE International Joint Conference on Neural Networks*, 2, 2005. 46, 48

[126] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017. 66, 67, 70, 71, 75, 92

[127] Jarosław M Granda, Liva Donina, Vincenza Dragone, De-Liang Long, and Leroy Cronin. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature*, 559(7714):377, 2018. 33, 143

[128] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Western Joint IRE-AIEE-ACM Computer Conference*, pages 219–224, 1961. 14, 16

[129] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016. 22

[130] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *NeurIPS*, 2018. 123

[131] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, pages 5356–5364, 2019. 87

[132] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden

Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019. 161

[133] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *EMNLP*, 2015. 54

[134] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017. 22

[135] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018. 105

[136] Larry Heck and Hongzhao Huang. Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs. *GlobalSIP*, 2014. 22

[137] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016. 105

[138] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 21, 46, 48

[139] Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. Grounded language learning fast and slow. In *ICLR*, 2021. 41

[140] Martin Hjelm, Carl Henrik Ek, Renaud Detry, and Danica Kragic. Learning human priors for task-constrained grasping. In *International Conference on Computer Vision Systems*, pages 207–217. Springer, 2015. 126

[141] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 20

[142] Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. Hierarchical decision making by generating and following natural language instructions. In *NeurIPS*, pages 10025–10034, 2019. 41, 104, 113

[143] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*, 2017. 65

[144] De-An Huang, Minghuang Ma, Wei-Chiu Ma, and Kris Kitani. How do we use our hands? Discovering a diverse set of common grasps. In *CVPR*, 2015. 126

[145] JMcVLevine Hunt. Intrinsic motivation and its role in psychological development. In *Nebraska Symposium on Motivation*, volume 13, pages 189–282. University of Nebraska Press, 1965. 101

[146] Zaeem Hussain, Mingda Zhang, Xiaozhong Zhang, Keren Ye, C. Thomas, Zuha

Agha, Nathan Ong, and Adriana Kovashka. Automatic understanding of image and video advertisements. In *CVPR*, pages 1100–1110, 2017. 39

[147] Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. COMET-ATOMIC 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*, 2021. 27

[148] Stephen James, Z. Ma, David Rovick Arrojo, and A. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5:3019–3026, 2020. 161

[149] Eric Jang, Sudheendra Vijayanarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *Proceedings of Machine Learning Research*, 78:119–132, 2017. 126

[150] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. TGIF-QA: Toward spatio-temporal reasoning in visual question answering. In *CVPR*, 2017. 65

[151] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. In *CVPR*, 2020. 86

[152] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *NeurIPS*, pages 9414–9426, 2019. 40, 101, 104

[153] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *TACL*, 8:423–438, 2020. 4, 19, 27, 31, 85

[154] Daniel D Johnson, Hugo Larochelle, and Daniel Tarlow. Learning graph structure with a finite-state automaton layer. In *NeurIPS*, 2020. 22

[155] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A Shamma, Michael S Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015. 36, 47

[156] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 38, 40, 65, 70

[157] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017. 65

[158] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017. 143

[159] Peter Kaiser, Mike Lewis, Ronald PA Petrick, Tamim Asfour, and Mark Steedman. Extracting common sense knowledge from text for robot planning. In *ICRA*, pages 3749–3756. IEEE, 2014. 42

[160] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, 2018. 123

[161] Immanuel Kant. Critique of pure reason. 1781. *Modern Classical Philosophers, Cambridge, MA: Houghton Mifflin*, pages 370–456, 1908. 13

[162] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, pages 321–328, 2003. 23

[163] Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. In *ICLR*, 2020. 22

[164] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UnifiedQA: Crossing format boundaries with a single QA system. *arXiv preprint arXiv:2005.00700*, 2020. 34, 85

[165] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *ICLR*, 2021. 22

[166] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *NeurIPS*, 2018. 73, 75, 83, 93

[167] Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009. 33, 143

[168] D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. In *ICLR*, 2015. 56

[169] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 22

[170] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 21, 23, 89, 95, 125, 131, 132

[171] Jamie Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 36

[172] Jamie Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NeurIPS*, 2015. 18

[173] Mia Kokic, Johannes A Stork, Joshua A Haustein, and Danica Kragic. Affor-

dance detection for task-specific grasping using deep learning. In *International Conference on Humanoid Robotics (Humanoids)*, pages 91–98. IEEE, 2017. 126

[174] Mia Kokic, Danica Kragic, and Jeannette Bohg. Learning task-oriented grasping from human activity datasets. In *IEEE Robotics and Automation Letters*. IEEE, 2020. 126

[175] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 157, 161

[176] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, 2002. 23

[177] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018. 141

[178] Satwik Kottur, Ramakrishna Vedantam, José MF Moura, and Devi Parikh. Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *CVPR*, pages 4985–4994, 2016. 36

[179] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual Genome: connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 26, 38, 54, 55, 65, 83, 84, 87, 88

[180] Tamar Kushnir and Alison Gopnik. Young children infer causal strength from probabilities and interventions. *Psychological Science*, 16(9):678–683, 2005. 139, 142

[181] Heinrich Küttler, Nantas Nardelli, Alexander H Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack learning environment. In *NeurIPS*, 2020. 161

[182] Safoura Rezapour Lakani, Antonio J Rodríguez-Sánchez, and Justus Piater. Exercising affordances of objects: A part-based approach. *IEEE Robotics and Automation Letters*, 3(4):3465–3472, 2018. 126

[183] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 2014. 35, 48

[184] Ni Lao, Tom Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *NeurIPS*, 2011. 36, 47

[185] Jey Han Lau, Timothy Baldwin, and Trevor Cohn. Topically driven neural language model. In *ACL*, 2017. 28

[186] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé. Hierarchical imitation and reinforcement learning. In *ICML*, pages 2917–2926, 2018. 105

[187] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, pages 3734–3743. PMLR, 2019. 22

[188] Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. *arXiv preprint arXiv:1911.07246*, 2019. 161

[189] Séverin Lemaignan, Mathieu Warnier, E Akin Sisbot, Aurélie Clodic, and Rachid Alami. Artificial cognition for social human–robot interaction: An implementation. *Artificial Intelligence*, 247:45–69, 2017. 27, 42

[190] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, November 1995. 14, 16, 24, 25, 141

[191] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Symposium on Experimental Robotics (ISER)*, 2016. 123

[192] Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. Unicoder-VL: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*, 2019. 37, 83

[193] Guohao Li, Hang Su, and Wenwu Zhu. Incorporating external knowledge to answer open-domain visual questions with dynamic memory networks. *arXiv preprint arXiv:1712.00733*, 2017. 65

[194] Guohao Li, Xin Wang, and Wenwu Zhu. Boosting visual question answering with context-aware knowledge aggregation. In *ACMMM*, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379885. 37, 82, 84, 93

[195] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019. 37, 82, 83, 85

[196] Yujia Li and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016. 20, 23, 46, 48, 89

[197] Gi Hyun Lim, Il Hong Suh, and Hyowon Suh. Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):492–509, 2010. 27, 42

[198] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva

Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 55, 67, 86

[199] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 86

[200] Hugo Liu and Push Singh. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004. 14, 16, 25, 31, 83, 88, 141, 163

[201] Weiyu Liu, Angel Daruna, and Sonia Chernova. A survey of semantic reasoning frameworks for robotic systems. *preprint*, 2020. URL `http://weiyuliu.com/data/A_Survey_of_Semantic_Reasoning_Frameworks_for_Robotic_Systems.pdf`. 41

[202] Weiyu Liu, Angel Daruna, and Sonia Chernova. CAGE: Context-aware grasping engine. In *ICRA*, 2020. 124, 125, 126, 128, 134, 135

[203] 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, 19

[204] John Locke. *An essay concerning human understanding*. Kay & Troutman, 1847. 13

[205] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *ECCV*, pages 852–869. Springer, 2016. 36

[206] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NeurIPS*, 2016. 65

[207] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. VilBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019. 37, 82, 83, 84, 85

[208] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In *CVPR*, pages 10437–10446, 2020. 37, 82

[209] Keting Lu, Shiqi Zhang, Peter Stone, and Xiaoping Chen. Robot representation and reasoning with knowledge from reinforcement learning. *arXiv preprint arXiv:1809.11074*, 2018. 32, 141

[210] Wenhao Lu, Jian Jiao, and Ruofei Zhang. TwinBERT: Distilling knowledge to twin-structured bert models for efficient retrieval. *arXiv preprint arXiv:2002.06275*, 2020. 4, 19, 27

[211] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob

Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, pages 6309–6317, 7 2019. 102

[212] 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 knowledge for commonsense question answering. In *AAAI*, volume 34, pages 8449–8456, 2020. 34, 85

[213] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020. 40, 105

[214] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. In *NeurIPS*, 2020. 22

[215] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *RSS*, 2017. 123

[216] Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *AAAI*, 2020. 34

[217] Mateusz Malinowski and Mario Fritz. Towards a visual turing challenge. *NeurIPS Workshops*, 2014. 81

[218] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NeurIPS*, 2014. 38, 65, 70

[219] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015. 65, 81

[220] Tomasz Malisiewicz and Alyosha Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NeurIPS*, 2009. 36, 47

[221] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019. 141

[222] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. In *CVPR*, 2017. 7

[223] Kenneth Marino, Abhinav Gupta, Rob Fergus, and Arthur Szlam. Hierarchical RL using an ensemble of proprioceptive periodic policies. In *ICLR*, 2019. 105

[224] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi.

OK-VQA: A visual question answering benchmark requiring external knowledge. In *CVPR*, 2019. 7, 81, 82, 84, 92, 93, 99

[225] Kenneth Marino, Rob Fergus, Arthur Szlam, and Abhinav Gupta. Empirically verifying hypotheses using reinforcement learning. *arXiv preprint arXiv:2006.15762*, 2020. 10, 150

[226] Kenneth Marino, Xinlei Chen, Devi Parikh, Abhinav Gupta, and Marcus Rohrbach. KRISP: Integrating implicit and symbolic knowledge for open-domain knowledge-based VQA. In *CVPR*, 2021. 8

[227] Vincenzo Di Massa, G. Monfardini, L. Sarti, F. Scarselli, M. Maggini, and M. Gori. A comparison between recursive neural networks and graph neural networks. *IEEE International Joint Conference on Neural Network Proceedings*, 2006. 46

[228] David Matthews, Sam Kriegman, Collin Cappelle, and Josh Bongard. Word2vec to behavior: morphology facilitates the grounding of language in machines. *IROS*, 2019. 104, 126

[229] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016. 40, 105

[230] Andrew N Meltzoff. Imitation, objects, tools, and the rudiments of language in human ontogeny. *Human evolution*, 3(1-2):45–64, 1988. 101

[231] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 2009. 46, 48

[232] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013. 17, 18

[233] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013. 17

[234] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, pages 746–751, 2013. 17, 28

[235] George A Miller. WordNet: a lexical database for english. *Commun. of the ACM*, 38(11):39–41, 1995. 25, 36, 54, 84, 88, 125, 129, 132

[236] Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Edward Grefenstette, and Sebastian Riedel. Scalable neural theorem proving on knowledge bases and natural language. 2018. 34

[237] Ishan Misra, C. Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. Seeing through the human reporting bias: Visual classifiers from noisy human-centric

labels. In *CVPR*, 2016. 55

[238] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhanava Dalvi, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir P. Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, R. Wang, Derry T. Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, M. Greaves, and Joel S. Welling. Never-ending learning. *Commun. ACM*, 61(5):103–115, April 2018. 30, 143, 157

[239] Bogdan Moldovan, Plinio Moreno, Martijn Van Otterlo, José Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *ICRA*, pages 4373–4378. IEEE, 2012. 42, 126

[240] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, volume 33, pages 4602–4609, 2019. 22

[241] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. In *ICCV*, 2019. 123, 130, 131

[242] Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. MarioQA: answering questions by watching gameplay videos. In *ICCV*, 2017. 65

[243] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. 2019. 128

[244] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *CoRL*, 2020. 10

[245] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-DOF grasping for target-driven object manipulation in clutter. In *ICRA*, 2020. 123

[246] Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. Text-based RL agents with commonsense knowledge: New challenges, environments and baselines. *arXiv preprint arXiv:2010.03790*, 2020. 41

[247] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874, 2018. 104

[248] Medhini Narasimhan and Alexander G. Schwing. Straight to the facts: Learning knowledge base retrieval for factual visual question answering. In *ECCV*, 2018. 37, 63, 65, 84

[249] Medhini Narasimhan, Svetlana Lazebnik, and Alexander G Schwing. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *NeurIPS*, 2018. 37, 63, 65, 82, 84, 87

[250] Arvind Neelakantan, Semih Yavuz, Sharan Narang, Vishaal Prasad, Ben Goodrich, Daniel Duckworth, Chinnadhurai Sankar, and Xifeng Yan. Neural assistant: Joint action prediction, response generation, and latent knowledge reasoning. *arXiv preprint arXiv:1910.14613*, 2019. 42

[251] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982. 14, 15

[252] Allen Newell and Fred M Tonge. An introduction to information processing language v. *Commun. ACM*, 3(4):205–211, 1960. 15

[253] Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *CVPR*, pages 12527–12537, 2019. 40, 105

[254] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016. 21, 46, 48

[255] David Novotny, Diane Larlus, and Andrea Vedaldi. I have seen enough: Transferring parts across categories. In *BMVC*. British Machine Vision Association and Society for Pattern Recognition, 2016. 30

[256] David Novotny, Diane Larlus, and Andrea Vedaldi. Learning the structure of objects from web supervision. In *ECCV*, pages 218–235. Springer, 2016. 30

[257] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, pages 2661–2670. JMLR. org, 2017. 40, 104

[258] Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. *IJCAI*, 2015. 23

[259] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, 2018. 22

[260] Xiaoman Pan, Kai Sun, Dian Yu, Jianshu Chen, Heng Ji, Claire Cardie, and Dong Yu. Improving question answering with external knowledge. In *MRQA@EMNLP*, 2019. 34

[261] J. Park, Chandra Bhagavatula, R. Mottaghi, A. Farhadi, and Yejin Choi. Visualcomet: Reasoning about the dynamic context of a still image. In *ECCV*,

2020. 39

[262] Robert Parker and Linguistic Data Consortium. *English Gigaword fifth edition*. Linguistic Data Consortium, 2011. ISBN 9781585635818. 28

[263] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *NeurIPS*, pages 1043–1049, 1998. 105

[264] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 91

[265] Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Springer, 2000. 32, 140, 142

[266] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014. 17, 27, 28, 29, 90

[267] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014. 22

[268] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018. 29

[269] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *EMNLP*, pages 2463–2473, Hong Kong, China, November 2019. 4, 19, 27, 31, 85

[270] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020. 34

[271] Pinar, Yanardag, and S. V. N. Vishwanathan. Deep graph kernels. *KDD*, 2015. 23

[272] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229, 1987. 20

[273] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*, 2016. 123

[274] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *ICRA*, pages 3515–3522. IEEE, 2012. 42

[275] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*,

pages 5099–5108, 2017. 131

[276] Meng Qu, Jian Tang, and Yoshua Bengio. Weakly-supervised knowledge graph alignment with adversarial learning. *arXiv preprint arXiv:1907.03179*, 2019. 34

[277] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*, 2020. 34

[278] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 4, 19, 28

[279] 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, 19, 29

[280] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *ACL*, 2019. 31

[281] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016. 27, 33, 34, 87

[282] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. *CoRR*, abs/1806.03822, 2018. 34, 143

[283] Justus J Randolph. Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*, 2005. 129

[284] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. ASAP: Adaptive structure aware pooling for learning hierarchical graph representations. In *AAAI*, volume 34, pages 5470–5477, 2020. 22

[285] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, pages 7263–7271, 2017. 36

[286] Mengye Ren, Jamie Kiros, and Richard S. Zemel. Exploring models and data for image question answering. In *NeurIPS*, 2015. 38, 65

[287] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 50, 55, 86

[288] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*, pages 74–84, 2013. 34, 142

[289] Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, pages 23–41, 1965. 15

[290] Marcus Rohrbach, Michael Stark, and Bernt Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011. 36

[291] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020. 22

[292] Sebastian Ruder. Why You Should Do NLP Beyond English. `http://ruder.io/nlp-beyond-english`, 2020. 23

[293] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 17

[294] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. v, 55, 87

[295] Fereshteh Sadeghi, Santosh K Divvala, and Ali Farhadi. VisKE: visual knowledge extraction and question answering by visual verification of relation phrases. In *CVPR*, 2015. 27, 30, 47, 66, 84

[296] Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. ExplaGraphs: An explanation graph generation task for structured commonsense reasoning. *arXiv preprint arXiv:2104.07644*, 2021. 34

[297] Guillaume Salha-Galvan, Romain Hennequin, and M. Vazirgiannis. Simple and effective graph autoencoders with one-hop linear models. *ECML*, abs/2001.07614, 2020. 22

[298] 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 *AAAI*, volume 33, pages 3027–3035, 2019. 26

[299] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula. Robobrain: Large-scale knowledge engine for robots. *arXiv preprint arXiv:1412.0691*, 2014. 27, 42

[300] Franco Scarselli, Marco Gori, Ah Chung Tsoi, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009. 20, 46, 48

[301] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018. 22, 23, 89

[302] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. *arXiv preprint arXiv:2010.00577*, 2020. 22

[303] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 112

[304] Sanket Shah, Anand Mishra, Naganand Yadati, and Partha Pratim Talukdar. KVQA: Knowledge-aware visual question answering. In *AAAI*, volume 33, pages 8876–8884, 2019. 38

[305] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020. 126

[306] Lanbo She, Yu Cheng, Joyce Y Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. Teaching robots new actions through natural language instructions. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873. IEEE, 2014. 41

[307] Nino Shervashidze, S. V. N. Vishwanathan, Tobias H. Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, volume 5, 2009. 23

[308] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 2011. 23

[309] Violetta Shevchenko, Damien Teney, A. Dick, and A. V. Hengel. Reasoning over vision and language: Exploring the benefits of supplemental knowledge. In *LANTERN*, volume abs/2101.06013, 2021. 37

[310] 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 *CVPR*, pages 10740–10749, 2020. 41

[311] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, 2012. 35, 48

[312] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *ICLR*, 2018. 40, 104

[313] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, pages 3693–3702, 2017. 22

[314] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 53

[315] Amanpreet Singh, Vedanuj Goswami, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. MMF: A multimodal framework for vision and language research. `https://github.com/facebookresearch/mmf`, 2020. 91

[316] Amanpreet Singh, Vedanuj Goswami, and Devi Parikh. Are we pretraining it right? digging deeper into visio-linguistic pretraining. *arXiv preprint arXiv:2004.08744*, 2020. 37

[317] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, pages 926–934. Citeseer, 2013. 34

[318] Dan Song, Kai Huebner, Ville Kyrki, and Danica Kragic. Learning task constraints for robot grasping using graphical models. In *IROS*, pages 1579–1585. IEEE, 2010. 125

[319] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. *arXiv preprint arXiv:2009.00563*, 2020. 161

[320] Daniil Sorokin and Iryna Gurevych. Modeling semantics with gated graph neural networks for knowledge base question answering. In *COLING*, 2018. 34

[321] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*, 2017. 16, 17, 132, 134

[322] Elizabeth S Spelke, Karen Breinlinger, Janet Macomber, and Kristen Jacobson. Origins of knowledge. *Psychological review*, 99(4):605, 1992. 142

[323] Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. WIT: Wikipedia-based image text dataset for multimodal multilingual machine learning. *arXiv preprint arXiv:2103.01913*, 2021. 27

[324] Christian Steinruecken, Emma Smith, David Janz, James Lloyd, and Zoubin Ghahramani. *Automated Machine Learning*. Springer Series on Challenges in Machine Learning, 2019. 33, 143

[325] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002. 105

[326] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: Pre-training of generic visual-linguistic representations. In *ICLR*, 2019. 37, 83

[327] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NeurIPS*, 2015. 74

[328] Chen Sun, Chuang Gan, and Ram Nevatia. Automatic concept discovery from

parallel text and visual corpora. In *ICCV*, pages 2596–2604, 2015. 30

[329] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*, 2018. 34

[330] Richard Sutton. The bitter lesson, Mar 2019. URL `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`. 5

[331] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. 105

[332] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021. 161

[333] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*, 2019. 32, 34

[334] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019. 37, 83

[335] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: understanding stories in movies through question-answering. In *CVPR*, 2016. 38, 65, 70

[336] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011. 40, 105, 143

[337] Andreas ten Pas and Robert Platt. Using geometry to detect grasp poses in 3D point clouds. In *Robotics Research*, pages 307–324. Springer, 2018. 128

[338] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *IJRR*, 36(13-14):1455–1473, 2017. 123, 130

[339] Moritz Tenorth and Michael Beetz. Representations for robot knowledge in the KnowRob framework. *Artificial Intelligence*, 247:151–169, 2017. 27, 42

[340] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018. 22

[341] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, 2014. 22

[342] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for

model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 40

[343] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*, 2021. 161

[344] Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. Joint video and text parsing for understanding events and answering queries. *IEEE MultiMedia*, 2014. 65

[345] Joep Vanlier, Christian A Tiemann, Peter AJ Hilbers, and Natal AW van Riel. Optimal experiment design for model selection in biochemical networks. *BMC systems biology*, 8(1):20, 2014. 33, 143

[346] Karthik Mahesh Varadarajan and Markus Vincze. AfRob: The affordance network ontology for robots. In *IROS*, pages 1343–1350. IEEE, 2012. 42

[347] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 19, 23, 83, 147

[348] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018. 22, 89

[349] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019. 22

[350] Oriol Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015. 36

[351] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016. 23

[352] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *JMLR*, 2010. 23

[353] Chenguang Wang, Xiao Liu, and Dawn Song. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*, 2020. 31

[354] David Wang, David Tseng, Pusong Li, Yiding Jiang, Menglong Guo, Michael Danielczuk, Jeffrey Mahler, Jeffrey Ichnowski, and Ken Goldberg. Adversarial grasp objects. In *Conference on Automation Science and Engineering*. IEEE, 2019. 123

[355] Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. Extracting multiple-relations in one-pass with pre-trained transformers. In *ACL*, 2019. 4, 31

[356] Peng Wang, Qi Wu, Chunhua Shen, Anthony R. Dick, and Anton van den

Hengel. Explicit knowledge-based reasoning for visual question answering. In *IJCAI*, 2017. 37, 38, 63, 65, 70, 71, 84

[357] Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. FVQA: fact-based visual question answering. *TPAMI*, 2017. 37, 38, 63, 65, 66, 70, 71, 75, 82, 84

[358] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023*, 2017. 34, 84

[359] Sida I Wang, Percy Liang, and Christopher D Manning. Learning language games through interaction. In *ACL*, 2016. 41

[360] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *CVPR*, 2018. 36

[361] Zhigang Wang, Juanzi Li, Zhiyuan Liu, and Jie Tang. Text-enhanced representation learning for knowledge graph. In *IJCAI*, pages 4–17, 2016. 34

[362] Edward C Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *ICRA*, pages 1–7. IEEE, 2018. 40, 104

[363] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871. PMLR, 2019. 22

[364] Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony R. Dick, and Anton van den Hengel. Visual question answering: A survey of methods and datasets. *arXiv preprint arXiv: 1607.05910*, 2016. 143

[365] Qi Wu, Peng Wang, Chunhua Shen, Anthony R. Dick, and Anton van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016. 65, 82, 84, 143

[366] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 86

[367] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020. 157, 161

[368] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu,

Minghua Liu, H. Jiang, Yifu Yuan, H. Wang, Li Yi, A. Chang, L. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *CVPR*, pages 11094–11104, 2020. 161

[369] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018. 22

[370] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *ICML*, 2016. 65

[371] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. Improving question answering over incomplete kbs with knowledge-aware reader. In *ACL*, 2019. 34

[372] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *ICLR*, 2020. 31

[373] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*, 2016. 65

[374] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5453–5462. PMLR, 2018. 22

[375] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019. 22

[376] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015. 34

[377] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. In *ICLR*, 2018. 42, 126

[378] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with BERTserini. *NAACL*, page 72, 2019. 34, 84

[379] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018, 2015. 34, 84

[380] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48. PMLR, 2016. 22

[381] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 65

[382] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with Freebase. In *ACL*, 2014. 34, 66, 84

[383] Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: open information extraction on the web. In *NAACL-HLT*, pages 25–26, 2007. 30

[384] Mark Yatskar, Vicente Ordonez, and Ali Farhadi. Stating the obvious: Extracting visual common sense knowledge. In *NAACL-HLT*, pages 193–198, 2016. 30

[385] Keren Ye and Adriana Kovashka. ADVISE: Symbolism and external knowledge for decoding advertisements. In *ECCV*, 2018. 39

[386] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *NeurIPS*, pages 1031–1042, 2018. 83

[387] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*, 2015. 34, 66

[388] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018. 22

[389] Haonan Yu, Haichao Zhang, and Wei Xu. Interactive grounded language acquisition and generalization in a 2D world. In *ICLR*, 2018. 40, 105

[390] Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. Visual madlibs: Fill in the blank description generation and question answering. In *ICCV*, 2015. 38, 65, 70, 71

[391] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30. Curran Associates, Inc., 2017. 22

[392] Edward N Zalta, Uri Nodelman, Colin Allen, and R Lanier Anderson. Stanford encyclopedia of philosophy. *Palo Alto CA: Stanford University*, 2005. 12, 13, 14

[393] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *ICRA*, pages 1–8. IEEE, 2018. 123

[394] Zhen Zeng, Adrian Röfer, and Odest Chadwicke Jenkins. Semantic linking maps for active visual object search. In *ICRA*, 2020. 42

[395] Shiqi Zhang and Peter Stone. CORPP: Commonsense reasoning and proba-

bilistic planning as applied to dialog with a mobile robot. In *AAAI*, 2015. 32, 141

[396] Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: Generalising to new environment dynamics via reading. In *ICLR*, 2020. 40, 105

[397] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 87

[398] Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629, 2018. 6, 34

[399] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and VQA. *arXiv preprint arXiv:1909.11059*, 2019. 37, 83

[400] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014. 5, 45, 47, 55

[401] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014. 25, 27, 36, 47, 66, 84, 126

[402] Yuke Zhu, Ce Zhang, Christopher Ré, and Li Fei-Fei. Building a large-scale multimodal knowledge base system for answering visual queries. *arXiv preprint arXiv:1507.05670*, 2015. 27, 30, 36, 47, 66, 84

[403] Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. Visual7W: grounded question answering in images. In *CVPR*, 2016. 38, 65, 70

[404] Yuke Zhu, Joseph J Lim, and Li Fei-Fei. Knowledge acquisition for visual question answering via iterative querying. In *CVPR*, pages 1154–1163, 2017. 27, 30, 66, 84

[405] Yukun Zhu, Jamie Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27, 2015. 28, 85