Doctoral Dissertation Proposal

Using Natural Language to Aid Task Specification in Sequential Decision Making Problems

Prasoon Goyal

Department of Computer Science
The University of Texas at Austin

Abstract

Intelligent agents that can help humans accomplish everyday tasks, such as a personal robot at home or a robot in a work environment, is a long-standing goal of artificial intelligence. One of the requirements for such general-purpose agents is the ability to teach them new tasks or skills relatively easily. Common approaches to teaching agents new skills include reinforcement learning (RL) and imitation learning (IL). However, specifying the task to the learning agent, i.e. designing effective reward functions for reinforcement learning and providing demonstrations for imitation learning, are often cumbersome and time-consuming.

We aim to use natural language as an auxiliary signal to aid task specification, which reduces the burden on the end user. To make reward design easier, we propose a novel framework that is used to generate language-based rewards in addition to the extrinsic rewards from the environment for faster policy training using RL. To ameliorate the problem of providing demonstrations, we propose a new setting that enables an agent to learn a new task without demonstrations in an IL setting, given a demonstration from a related task and a natural language description of the difference between the desired task and the demonstrated task.

The primary contributions of this dissertation will be new frameworks that enable incorporating natural language in RL and IL, which would enable non-expert users to specify new tasks to intelligent agents more conveniently.

Contents

Chapter 1

Introduction

Teaching agents to perform new tasks is one of the central problems in artificial intelligence. One popular paradigm, called sequential decision making, involves an agent interacting with an environment, for instance, a robot in a home environment. Over the past few decades, several approaches have been developed to train agents to perform a variety of tasks in this setting. Broadly, these approaches can be divided into two classes—**reinforcement learning** (RL) and **imitation learning** (IL). In reinforcement learning, the agent receives a reward from the environment for every action, and needs to learn a behavior that maximizes its sum of expected future rewards. Imitation learning is an alternate method for teaching agents—instead of the reward, the agent receives a set of demonstrations from an expert, and needs to learn a behavior.

While these approaches have been successfully applied to domains ranging from game-playing to real robotics tasks, making them useful for large-scale real-world applications is still a distant goal. One of the challenges in scaling up these approaches to real-world problems is communicating all the useful information to complete the task to the learning agent, such as the task objective, hints, constraints, and user preferences. In the standard RL or IL settings, communicating such information would require designing intricate reward functions or providing a large number of demonstrations, respectively, which are often difficult to scale as the complexity of tasks grows. A promising direction to address this challenge is to use natural language as an auxiliary signal in sequential decision making. Language could be used in a variety of ways, for instance, in (1) communicating the task, (2) providing feedback for agent's performance, (3) guiding the agent to focus on the important aspects of the task, and (4) enabling the agent to ask clarification questions. Thus, developing techniques that can effectively use natural language in sequential decision making would ameliorate several limitations of current approaches, by making them more sample efficient and robust to task-irrelevant features. Further, language would enable non-expert users to communicate their intent to learning agents, which is an important requirement for integrating intelligent systems into our everyday lives.

In this dissertation proposal, we focus on using language to communicate the task, where we seek to answer the following question:

How can natural language be used as an auxiliary signal to reduce the burden of task specification on the end user?

Several approaches have been proposed in the past that use natural language to communicate with a learning agent. However, a significant fraction of these approaches focus on *instruction-following*, where language is essentially used to communicate the task goals to the agent (See Section 2.2.2). While useful, we posit that natural language can be used to communicate additional information to the agent, and propose techniques that can use language as an *auxiliary signal*, which can be used in conjunction with the primary learning signal (i.e. reward for RL, and demonstrations for IL).

The contributions proposed in this dissertation proposal are as follows:

1. **Language to Aid Task Specification in Reinforcement Learning**: We consider the setting where the agent is provided with a linguistic description of the task, in addition to the reward function from the environment. Note that this is different from instruction-following, where language is the *only* signal to communicate the task, whereas in the setting we consider, language is an auxiliary signal, provided *in addition* to the environment rewards.

   (a) We develop a framework that learns a relatedness model between the task description and the agent's actions using supervised learning, for a discrete action space. The trained relatedness model is then used to provide additional rewards to the agent in an RL setting [Goyal et al., 2019a].

   (b) We extend the model in [Goyal et al., 2019a] to work with continuous action spaces, and learn a relatedness model between the task description and the states (instead of between the description and actions) [Goyal et al., 2020].

2. **Language to Aid Task Specification in Imitation Learning**: We introduce a new setting, where the agent is provided with the demonstration of a *source* task, and needs to learn a slightly different task, the *target* task, with the difference between the source and target tasks communicated using natural language.

   (a) We present a model that outputs the reward function for the target task, by predicting the goal state for the target task as an intermediate step [Goyal et al., 2021b].

   (b) (Proposed) We propose a neurosymbolic task adaptation model that allows adaptation in non-goal-based tasks.

   (c) (Proposed) We propose a model that enables policy transfer from the source task to the target task, by learning to predict how the policy should change given the linguistic description.

The techniques we propose enable learning a policy using RL from sparse or coarse dense rewards, and learning from demonstrations of related tasks in IL, by using language to communicate additional information. This reduces the burden of task specification on the end user.

Chapter 2

Background and Related Work

2.1 Sequential Decision Making

Several tasks in the real world can be formulated as an agent interacting with an environment, wherein at each step, the agent receives an observation from the environment and takes an action, and the environment transitions to a new state. The final state of the environment is, therefore, the result of a sequence of action decisions made by the agent. As such, these problems are referred to as *sequential decision making* problems. Examples include interacting with objects in the real world to accomplish a desired goal, such as cooking, rearranging a room, driving, or assembling furniture, as well as virtual environments, such as playing video games, or ordering an item from a website.

Over the last several decades, many approaches have been proposed to train learning agents for such problems. Broadly, these approaches can be divided into two categories—reinforcement learning and imitation learning.

2.1.1 Reinforcement Learning

In reinforcement learning, the agent receives a reward from the environment at every timestep, which is the primary learning signal [Sutton and Barto, 2018]. The standard reinforcement learning (RL) setup is often represented using the Markov Decision Process (MDP) formalism. An MDP can be defined by the tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of all states in the environment, A is the set of all actions available to the agent, $T : S \times A \times S \to [0, 1]$ is the transition function, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. At timestep t, the agent observes the current state $s_t \in S$, and takes an action $a_t \in A$. The environment transitions to a new state $s_{t+1} \sim T(s_t, a_t, \cdot)$, and the agent receives a reward $R_t = R(s_t, a_t)$.

The objective is to learn a policy $\pi : S \times A \to [0, 1]$ that maximizes the expected return, defined as the discounted sum of future rewards, $G_t = \sum_{i=t}^{T} \gamma^{i-t} R_i$, where T is the horizon of the episode.

A state-value function $V^\pi : S \to \mathbb{R}$ is defined as the expected return from a state when following policy π. Given a policy π, the state-value function can be computed using the

Bellman equation,

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a) \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi}(s') \right)$$

An action-value function $Q^{\pi} : S \times A \to \mathbb{R}$ is defined as the expected return when action a is taken at state s, and the policy π is followed thereafter. An analogous Bellman equation can be written in terms of the action-value function.

Reinforcement Learning Algorithms

Various approaches have been proposed to learn an optimal policy, which can be classified into the following categories.

Value-based methods. Value-based methods learn the optimal state-value function $V^*(s) = \max_{\pi} V^{\pi}(s)$ or the optimal action-value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$, from which the optimal policy π^* can be recovered [Watkins and Dayan, 1992, Mnih et al., 2015].

Policy-based methods. Policy-gradient methods learn an optimal policy π^* directly, by maximizing the expected total reward. The gradient of the policy may be defined in multiple ways, such as using the reward or the value function [Williams, 1992, Schulman et al., 2015b, Lillicrap et al., 2015].

Actor-critic methods. Finally, actor-critic methods learn both a policy (*actor*), and a value function (*critic*), that are trained jointly [Schulman et al., 2015a, Schulman et al., 2017, Haarnoja et al., 2018].

Challenges in Reinforcement Learning

While reinforcement learning has been successfully applied in multiple domains, several challenges remain that need to be addressed to make RL practical for real-world applications.

First, specifying a reward function that accurately and efficiently encodes the desired task may be non-trivial. A common approach that works for simple tasks consists of using a sparse reward—for instance, the agent receives a non-zero reward on completing the task, and a zero reward in all other cases. To learn from such a reward function, the agent must explore without much feedback from the environment, which could become prohibitively time-consuming as the complexity of tasks grows. One way to address this limitation is to define a dense reward function, which is non-zero at most timesteps, giving the agent additional signal as it is making progress towards the goal [Ng et al., 1999]. However, designing dense rewards is itself challenging, and may lead to reward hacking, where the agent finds an undesired behavior that achieves a high reward [Amodei and Clark, 2016].

Another challenge in RL is sample efficiency—even for hand-designed reward functions, the agent might need a considerable number of interactions with the environment to learn the desired behavior. Approaches to address this involve using expert data [Brys et al., 2015], curiosity-driven exploration in which the agent is rewarded for visiting new regions of the

state space [Burda et al., 2018, Achiam and Sastry, 2017, Schmidhuber, 1991], hierarchical RL in which a low-level policy is trained to reach different subgoals and a high-level policy is trained to predict the next subgoal [Barto and Mahadevan, 2003, Vezhnevets et al., 2017], and transfer learning in which a policy trained for one task is adapted for a different task [Taylor and Stone, 2009].

Finally, when policies are trained or deployed in the real world, there are safety concerns that must be taken into account [Garcıa and Fernández, 2015]. Common approaches to safe RL include adding constraints to prevent the agent from visiting unsafe states, and modifying the reward function such that unsafe states have a large negative value.

In this proposal, we present approaches that use language to address reward design and sample efficiency (Chapters 3 and 4). We show that rewards can be generated from natural language descriptions, which addresses the reward design problem, particularly for non-experts. Further, the rewards generated from natural language help improve the sample efficiency of the learner, both in sparse and dense reward settings.

2.1.2 Imitation Learning

The imitation learning problem can be described using an MDP $\setminus$ R $= \langle S, A, T, \gamma \rangle$, where the symbols denote the state space, the action space, the transition function, and the discount factor, as detailed in Section 2.1.1. Instead of the reward, the agent has access to a set of expert demonstrations $\tau_1, \ldots, \tau_N$. The goal of imitation learning is to infer the demonstrator's intent, and thereby learn a policy $\pi : S \times A \to [0, 1]$ that maximizes the return, under the unknown expert reward function.

Imitation Learning Algorithms

Approaches in imitation learning can be classified into the following categories.

Behavior cloning. Given a set of demonstrations $\tau_1, \ldots, \tau_N$, where $\tau_i = ((s_{ij}, a_{ij}))_{j=1}^{N_i}$, a policy $\pi : S \times A \to [0, 1]$ is learned using the state-action pairs in all the demonstrations, in a supervised learning setting [Pomerleau, 1991]. When used for predictions on unseen states, minor errors in prediction cause the agent to diverge from the data distribution the policy was trained on, leading to compounding errors. Several methods have been proposed that address this limitation, such as [Ross et al., 2011, Brantley et al., 2019].

Inverse Reinforcement Learning. In these approaches, a reward function is inferred from the demonstrations, and a policy is learned using RL on the recovered reward function [Ng et al., 2000, Ziebart et al., 2008a, Ramachandran and Amir, 2007].

Adversarial Imitation Learning. These approaches learn a policy jointly with a discriminator, such that the discriminator tries to distinguish states visited by the policy from the states visited by the expert, and the policy tries to visit states so as to confuse the discriminator [Ho and Ermon, 2016, Torabi et al., 2018].

Challenges in Imitation Learning

Imitation learning is fundamentally an ill-defined problem—given a set of demonstrations, there are multiple reward functions under which the demonstrations are optimal. However, not all these reward functions lead to policies that may be desirable under the demonstrator's true reward function. To address this, several approaches have been proposed that model this uncertainty [Ziebart et al., 2008b, Ramachandran and Amir, 2007], obtain feedback/-corrections from a human [Cui and Niekum, 2018, Niekum et al., 2013], or formulate the problem differently such as learning from pairwise rankings between a set of demonstrations [Brown et al., 2019].

Another major challenge in imitation learning is that of sample efficiency—providing demonstrations is often cumbersome, and therefore, we would like the learning agent to learn from as few demonstrations as possible. Recently, several approaches have been proposed that use meta-learning [Thrun and Pratt, 2012], to enable few-shot imitation learning [Finn et al., 2017, Duan et al., 2017, Pathak et al., 2018].

In this proposal, we take a step towards addressing sample efficiency in imitation learning, using language. We propose a framework which enables reusing demonstrations from related tasks, where the difference between the target task and the demonstrated task is specified using language (Chapters 5 and 6). This allows the agent to learn a new task in a zero-shot setting, that is, without any new demonstrations.

2.2 Language Grounding

Language grounding, or symbol grounding, refers to the problem of mapping linguistic concepts to the agent's perception and actions [Harnad, 1990]. For example, to understand an instruction such as "Place the mug next to the book", a learning agent needs to map *mug* and *book* to objects in the environment, *place* to an action, and *next to* to a spatial relation in the environment.

2.2.1 Grounding Language to Images and Videos

In the last few years, many problems have been introduced that require grounding linguistic concepts to images or videos, such as image captioning [You et al., 2016, Anderson et al., 2018a], video captioning [Venugopalan et al., 2015b], visual question-answering [Antol et al., 2015], and identifying regions of an image referred to by a natural language expression [Kazemzadeh et al., 2014].

2.2.2 Instruction-following

In a large class of problems, which can broadly be termed as *instruction-following*, language is used to communicate the task to a learning agent. In this setting, the agent is given a natural language command for a task, and is trained to take a sequence of actions that complete the task. A popular approach for instruction-following involves using graphical models that are instantiated based on the structure of the natural language command, from which the most likely plan is inferred [Tellex et al., 2011, Howard et al., 2014b, Howard et al., 2014a]. Other

approaches involve parsing the input command into a rich grammar formalism [Artzi and Zettlemoyer, 2013], using specialized neural network architectures to map instructions to visual goals [Misra et al., 2018] position visitation [Blukis et al., 2018b, Blukis et al., 2019], intermediate goals [Paxton et al., 2019], or actions [Bisk et al., 2016, Stepputtis et al., 2020a].

A well-studied problem in this setting is Vision-and-language Navigation, where the tasks consist of navigating to a desired location in an environment, given a natural language command and an egocentric view from the agent's current position [Anderson et al., 2018b, Fried et al., 2018, Wang et al., 2019]. Another subcategory of instruction-following involves instructing an embodied agent using natural language [Tellex et al., 2011, Hemachandra et al., 2015, Arkin et al., 2017, Shridhar et al., 2020, Stepputtis et al., 2020b, Misra et al., 2016, Sung et al., 2018a, Blukis et al., 2018a, Blukis et al., 2019, Shao et al., 2020].

The approaches that we propose are different from instruction-following in that we use language as an *auxiliary signal* in addition to the main supervisory signal for the setting (i.e. reward for RL and demonstrations for IL).

2.2.3 Language to Aid Learning

A number of approaches have been proposed that use language to aid a learning agent.

Some of these approaches use language to generate plans. [Branavan et al., 2012a] extract preconditions from text to generate plans for games. [Sung et al., 2018b] use language in conjunction with trajectories and point clouds to plan manipulation trajectories to interact with unseen objects. [Nyga et al., 2018] use language in a dialog setting to fill incomplete plans. These approaches are different from our framework presented in Chapters 3 and 4, in that we incorporate language into reinforcement learning, instead of planning.

Among the approaches that use language in an RL setting, [Narasimhan et al., 2017, Narasimhan et al., 2018] use language to transfer concepts from one task to another to speed up RL. [Kuhlmann et al., 2004] use language to modify the Q-function of an RL agent. [Branavan et al., 2012b] use language to generate additional features for speeding up learning. [Kaplan et al., 2017] use a predefined set of natural language commands which are used to generate additional rewards to train an RL agent. [Hutsebaut-Buysse et al., 2020b] use pretrained word embeddings to make goal-conditioned RL sample efficient. [Abolghasemi et al., 2018] use natural language to attend to specific parts of the state to learn more robust visuomotor policies. [Wang and Narasimhan, 2021] learn to ground language to entities and dynamics, simultaneously with the policy. [Tambwekar et al., 2021] propose initializing decision-tree based policies using natural language.

Language has also been used to incorporate human feedback. [Broad et al., 2017] propose using natural language corrections for robotic manipulators. [Co-Reyes et al., 2018] propose iterative natural language corrections to communicate the goal to a learning agent. [Mehta and Goldwasser, 2019] propose using natural language for advice, in an instruction-following setting. [Sumers et al., 2020] use linguistic feedback to learn rewards by mapping language to features and sentiment. The framework we present in Chapters 3 and 4 is orthogonal to most of these approaches, and can be combined with them relatively easily.

[Andreas et al., 2017] propose to exploit compositionality in natural language to aid generalization, by mapping tasks into a latent space of language. Here, language is used to describe [Hutsebaut-Buysse et al., 2020a] propose a transfer learning framework, wherein a

new task can be learned from a set of previously learned tasks, each of which is described using language. While related to our setting introduced in Chapter 5, in these methods, language is provided for each task independently, and tasks are deemed similar if their linguistic descriptions are related. In our setting, however, language is used to explicitly describe the difference between two tasks. [Kamlish et al., 2019] use natural language commentary in the domain of chess to learn an evaluation function for moves, but is different from our approaches, as in this approach, language is only used at training time whereas for the frameworks we propose, a key element is using language at test time to aid learning a *new* task not seen during training.

Chapter 3

Using Natural Language for Reward Shaping in Reinforcement Learning

3.1 Introduction

One of the key challenges in applying reinforcement learning to a problem is designing reward functions that accurately and efficiently describe tasks. For the sake of simplicity, a common strategy is to provide the agent with sparse rewards—for example, positive reward upon reaching a goal state, and zero reward otherwise. However, it is well-known that learning is often difficult and slow in sparse reward settings [Večerík et al., 2017]. By contrast, dense rewards can be easier to learn from, but are significantly more difficult to specify. In this work, we address this issue by using natural language to provide dense rewards to RL agents in a manner that is easy to specify.

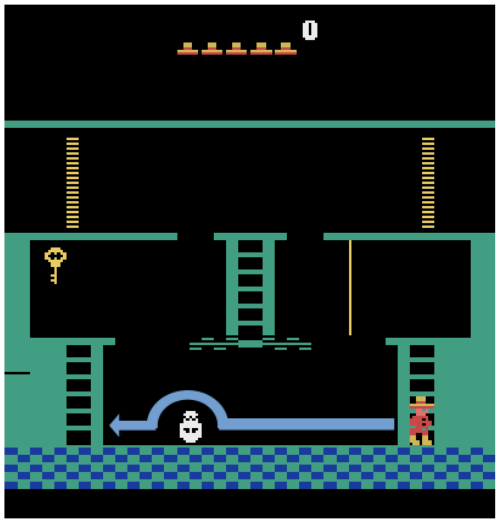


Figure 3.1: An agent exploring randomly to complete the task described by the blue trajectory may need considerable amount of time to learn the behavior. By giving natural language instructions like "Jump over the skull while going to the left", we can give intermediate signals to the agent for faster learning.

Consider the scenario in Figure 3.1 from the Atari game Montezuma's Revenge. Suppose we want the agent to go to the left while jumping over the skull (as shown in the blue trajectory). If the agent is given a positive reward only when it reaches the end of the desired trajectory, it may need to spend a significant amount of time exploring the environment to learn that behavior. Giving the agent intermediate rewards for progress towards the goal can help, a technique known as "reward shaping" [Ng et al., 1999]. However, designing intermediate rewards is hard, particularly for non-experts.

Instead, we propose giving the agent intermediate rewards using instructions in natural language. For instance, the agent can be given the following instruction:"Jump over the skull while going to the left" to provide intermediate rewards that accelerate learning. Since natural language instructions can easily be provided even by non-experts, it will enable them to teach RL agents new skills more conveniently.

The main contribution of this work is a new framework which takes an arbitrary natural language instruction and the trajectory executed by the agent so far, and makes a prediction whether the agent is following the instruction, which can then be used as an intermediate reward. Our experiments show that by using such reward functions, we can speed up learning in sparse reward settings by guiding the exploration of the agent. We describe the approach and the main experimental results below; for full details, see [Goyal et al., 2019b].

3.2 Overview of the Approach

In this work, we consider an extension of the MDP framework, defined by $\langle S, A, R, T, \gamma, l \rangle$, where $l \in L$ is a language command describing the intended behavior (with L defined as the set of all possible language commands). We denote this language-augmented MDP as MDP+L. Given an MDP(+L), reinforcement learning can be used to learn an optimal policy $\pi^* : S \to A$ that maximizes expected sum of rewards. We use R_{ext} ("extrinsic") to denote the MDP reward function above, to avoid confusion with language-based rewards that we define in Section 3.4.

In order to find an optimal policy in an MDP+L, we use a two-phase approach (Figure 3.2):

LanguagE-Action Reward Network (LEARN). In this step, we train a neural network that takes paired (trajectory, language) data from the environment and predicts if the language describes the actions within the trajectory. To train the network, we collect natural language instructions for trajectories in the environment (Section 3.3).

Language-aided RL. This step involves using RL to learn a policy for the given MDP+L. Given the trajectory executed by the agent so far and the language instruction, we use LEARN to predict whether the agent is making progress and use that prediction as a shaping reward (Section 3.4). Note that since we are only modifying the reward function, this step is agnostic to the particular choice of RL algorithm.

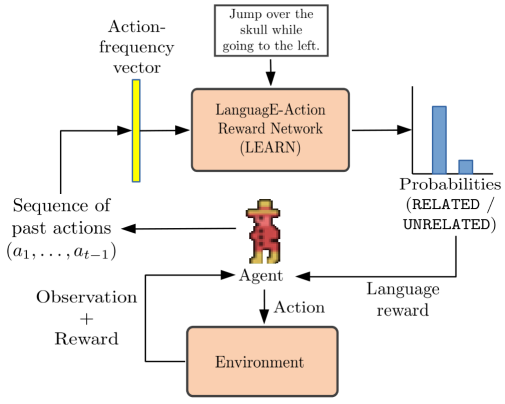


Figure 3.2: Our framework consists of the standard RL module containing the agent-environment loop, augmented with a LanguagE-Action Reward Network (LEARN) module.

3.3 LanguagE-Action Reward Network

LEARN takes in a trajectory and a language description and predicts whether the language describes the actions in the trajectory. More formally, given a trajectory τ, we create *action-frequency vectors* from it as follows:
1. Sample two distinct timesteps i and j (such that $i < j$) from the set $\{1, \ldots, |\tau|\}$, where $|\tau|$ denotes the number of timesteps in τ. Let $\tau[i:j]$ denote the segment of τ between timesteps i and j.
2. Create an *action-frequency vector* f from the actions in $\tau[i:j]$, where the dimensionality of f is equal to the number of actions in the MDP+L, and the k^{th} component of f is the fraction of timesteps action k appears in $\tau[i:j]$.

Using the above process, we create a dataset of (f, l) pairs from a given set of (τ, l) pairs. Positive examples are created by sampling f from a given trajectory τ and using the language description l associated with τ. Negative examples are created by (1) sampling an action-frequency vector f from a given trajectory τ, but choosing an alternate language description l' sampled uniformly at random from the data excluding l, or (2) creating a random action-frequency vector f' and pairing it with the language description l. These examples are used to train a neural network, as described below. Thus, given a pair (f, l), the network learns to predict whether the action-frequency vector f is related to the language description l or not.

Neural Network Details

The action-frequency vector is passed through a sequence of fully-connected layers to get an encoded action vector with dimension D_1. To embed the natural language instruction into a D_2-dimensional vector, we experimented with three models:

1. **InferSent** : In this model, we used a pretrained sentence embedding model [Conneau et al., 2017], which embeds sentences into a 4096-dimensional vector space. The 4096-dimensional vectors were projected to D_2-dimensional vectors using a fully-connected layer. We train only the projection layer during training, keeping the original sentence embedding model fixed.

2. **GloVe+RNN** : In this model, we represent the sentence using pretrained 50-dimensional GloVe word embeddings [Pennington et al., 2014], and train a two-layer GRU [Cho et al., 2014] encoder on top of it, while keeping the word embeddings fixed. We used the mean of the output vectors from the top layer as the encoding of the sentence. The hidden state size of the GRUs was set to D_2.

3. **RNNOnly** : This model is identical to Glove+RNN, except instead of starting with pretrained GloVe vectors, we randomly initialize the word vectors and train both the word embeddings and the two-layer GRU encoder.

These three models trade-off prior domain knowledge with flexibility – InferSent model starts with the knowledge of sentence similarity and is least flexible, GloVe+RNN model starts with word vectors and is more flexible in combining them to generate sentence embeddings,

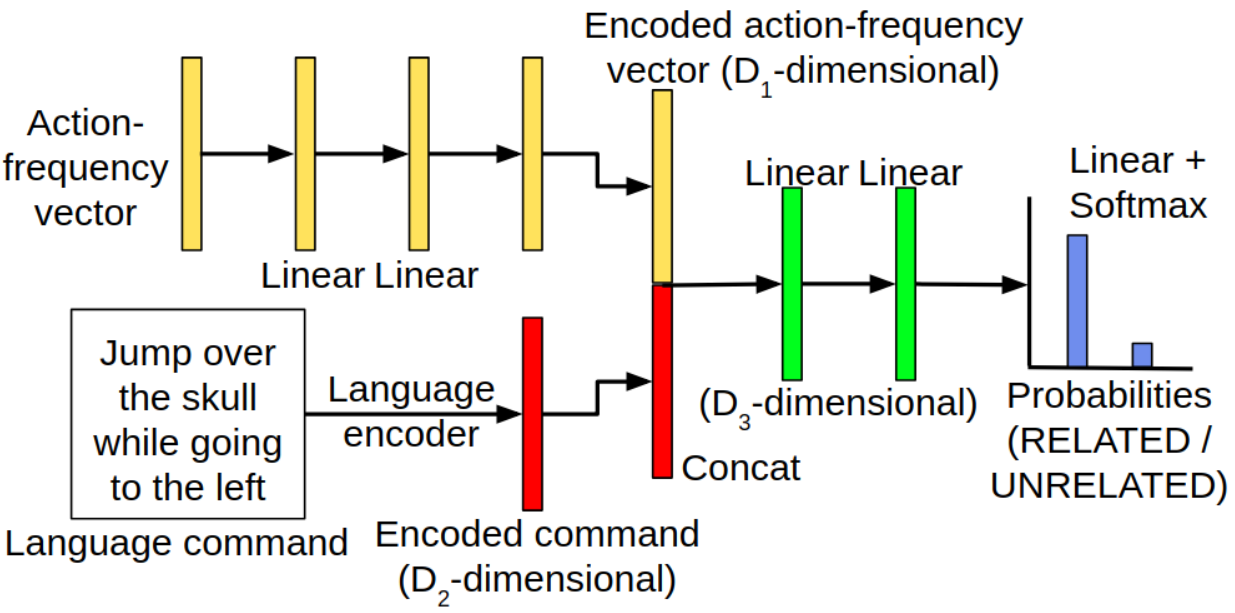


Figure 3.3: Neural network architecture for LEARN (Section 3.3)

while RNNOnly starts with no linguistic knowledge and is completely flexible while learning word and sentence representations.

The encoded action-frequency vector and language vector are then concatenated, and further passed through another sequence of fully-connected layers, each of dimension D_3, followed by a softmax layer. The final output of the network is a probability distribution over two classes – `RELATED` and `UNRELATED`, corresponding to whether the action-frequency vector f can be explained by the language instruction l. Our complete neural network architecture is shown in Figure 3.3. D_1, D_2 and D_3 were tuned using validation data.

We used backpropagation with an Adam optimizer [Kingma and Ba, 2014] to train the above neural network for 50 epochs to minimize cross-entropy loss.

To collect data for training LEARN, we used trajectories of human gameplay from the Atari Grand Challenge dataset [Kurin et al., 2017], and obtained linguistic descriptions for short segments from these trajectories using Amazon Mechanical Turk. The final dataset consisted of 6,870 linguistic descriptions paired with the corresponding trajectories.

3.4 Using Language-based Rewards in RL

To incorporate language information into RL, we use LEARN's predictions to generate intermediate rewards. Given the sequence of actions $a_1, \ldots, a_{t-1}$ executed by the agent until timestep t and the language instruction l associated with the given MDP+L, we create an action-frequency vector f_t, by setting the k^{th} component of f equal to the fraction of timesteps action k appears in $a_1, \ldots, a_{t-1}$. The resulting action-frequency vector f and the language instruction l are passed to LEARN. Let the output probabilities corresponding to classes `RELATED` and `UNRELATED` be denoted as $p_R(f_t)$ and $p_U(f_t)$. Note that since l is fixed for a given MDP+L, $p_R(f_t)$ and $p_U(f_t)$ are functions of only the current action-frequency

vector f_t.

Intuitively, trajectories that contain actions described by the language instruction more often will have higher values of $p_R(f_t)$, compared to other trajectories. For instance, if the language instruction is "Jump over the skull while going to the left", then trajectories with high frequencies corresponding to the "jump" and "left" actions will be considered more related to the language by LEARN. Therefore, we can use these probabilities to define intermediate language-based rewards. These intermediate rewards will enable the agent to explore more systematically, by choosing relevant actions more often than irrelevant actions.

To map the probabilities to language-based shaping rewards, we define a potential function for the current timestep as $\phi(f_t) = p_R(f_t) - p_U(f_t)$. The intermediate language-based reward is then defined as $R_{lang}(f_t) = \gamma \cdot \phi(f_t) - \phi(f_{t-1})$, where γ is the discount factor for the MDP+L. We show in the supplementary material that a policy that is optimal under the original reward function (R_{ext}) is also optimal under the new reward function ($R_{ext} + R_{lang}$).

3.5 Experimental Evaluation

To validate the effectiveness of our approach, we conducted experiments on the Atari game Montezuma's Revenge. The game involves controlling an agent to navigate around multiple rooms. There are several types of objects within the rooms – (1) ladders, ropes, doors, etc. that can be used to navigate within a room, (2) enemy objects (such as skulls and crabs) that the agent needs to escape from, (3) keys, daggers, etc. that can be collected. A screenshot from the game is included in Figure 3.1. We selected this game because the rich set of objects and interactions allows for a wide variety of natural language descriptions.

We define a set of 15 diverse tasks in multiple rooms, each of which requires the agent to go from a fixed start position to a fixed goal position while interacting with some of the objects present in the path.[1] For each task, the agent gets an extrinsic reward of +1 from the environment for reaching the goal, and an extrinsic reward of zero in all other cases.

For each of the tasks, we generate a reference trajectory, and use Amazon Mechanical Turk to obtain 3 descriptions for the trajectory. We use each of these descriptions as language commands in our MDP+L experiments, as described below. Note that we do not use the reference trajectories to aid learning the policy in MDP+L; they are only used to collect language commands to be used in our experiments.

For all experiments, the policy was trained using Proximal Policy Optimization, a popular on-policy RL algorithm [Schulman et al., 2017], for 500,000 timesteps.

We experiment with 2 different RL setups to evaluate how much using language-based rewards help:

1. **ExtOnly**: In this setup, we use the original environment reward, without using language-based reward. This is the standard MDP setup, and serves as our baseline.

2. **Ext+Lang**: In this setup, in addition to the original environment reward that the agent gets on completing the task successfully, we also provide the agent potential-based

[1] Although the tasks (and corresponding descriptions) involve interactions with objects, we observe that just using actions, as we do in our approach, already gives improvements over the baseline, likely because most objects can be interacted with only in one way.

language reward R_{lang} at each step, as described in Section 3.4.

We use the following metrics:

1. **AUC**: From each policy training run, we plot a graph with the number of timesteps on the x-axis and the number of successful episodes on the y-axis. The area under this curve is a measure of how quickly the agent learns, and is the metric we use to compare two policy training runs.

2. **Final Policy**: To compare the final learned policy with ExtOnly and Ext+Lang, we perform policy evaluation at the end of 500,000 training steps. For each policy training run, we use the learned policy for an additional 10,000 timesteps without updating it, and record the number of successful episodes.

For the Ext+Lang setup, we perform validation over the three types of language encoders described in Section 3.4 (InferSent / GloVe+RNN / RNNOnly). For each type of language encoder, we use the LEARN model with the best accuracy on the validation data. Further, we define the joint reward function as $R_{total} = R_{ext} + \lambda R_{lang}$. The type of language encoder and the hyperparameter λ are selected using validation.

Results

At test time, we performed 10 policy learning runs with different initializations for each task and each description. The results, averaged across all tasks and descriptions, are summarized in Figure 3.4, from which we can conclude that Ext+Lang learns much faster than ExtOnly, demonstrating that using natural language instructions for reward shaping is effective. In particular, the average number of successful episodes for ExtOnly after 500,000 timesteps is 903.12, while Ext+Lang achieves that score only after 358,464 timesteps, which amounts to a 30% speed-up. Alternately, after 500,000 timesteps, Ext+Lang completes 1529.43 episodes on average, compared to 903.12 for ExtOnly, thereby giving a 60% relative improvement.

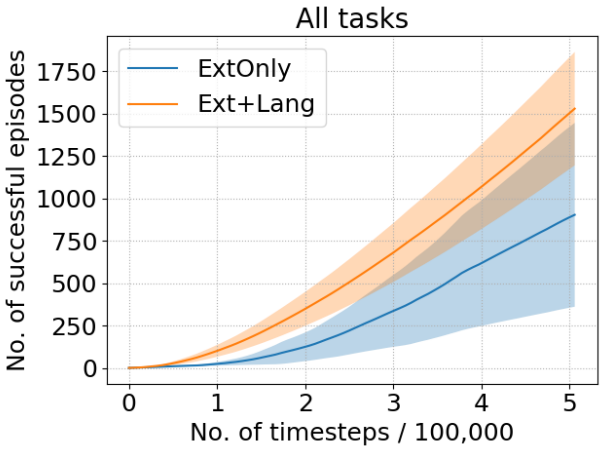


Figure 3.4: Comparison of different reward functions: The solid lines represent the mean successful episodes averaged over all tasks, and the shaded regions represent 95% confidence intervals.

Further, we found that Ext+Lang results in statistically significant improvement for 11 out of 15 tasks in AUC, and for 8 out of 15 tasks in the final policy, demonstrating that using language-based rewards lead to both faster and more robust policies on average.

Chapter 4

Guiding Reinforcement Learning Using Natural Language by Mapping Pixels to Rewards

4.1 Introduction

The approach presented in Chapter 3 is a simple and effective way to generate intermediate rewards for sample-efficient learning in sparse reward settings. However, it suffers from several limitations—(1) the action frequency vector can only be defined for discrete action spaces, (2) by only looking at the frequency of actions, the temporal information in the trajectories is discarded, and most importantly, (3) the model only uses the actions, ignoring the information in the states. In this work, we propose an improved model that addresses these limitations, and apply it to a simulated robot manipulation domain.

Our experiments on a diverse set of tasks in the Meta-World domain [Yu et al., 2019] demonstrate that the proposed approach results in improved sample efficiency during policy learning, both in sparse and hand-designed dense reward settings. This motivates a new paradigm where language could be used to improve over hand-designed rewards, which may be suboptimal owing to the difficulty of designing rewards by hand.

We describe the approach and the main experimental results below; for full details, see [Goyal et al., 2020].

4.2 Approach

We use the MDP+L framework introduced in Section 3.2, defined as $M' = \langle S, A, T, R, \gamma, L \rangle$, where L is an instruction describing the task using natural language, and the other quantities are as defined as in a standard MDP. Further, we also use the same two-phase framework for learning in an MDP+L described in Chapter 3, which we describe again below:

Phase 1: A neural network (PixL2R) is trained to predict whether a given trajectory and language are related or not. This requires paired ⟨trajectory, language⟩ data in the environment. We describe this phase in detail in Section 4.2.1.

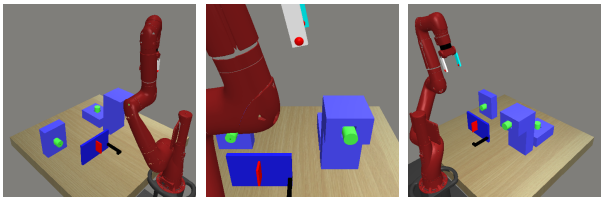


Figure 4.1: Viewpoints used for data collection and experiments.

Phase 2: Next, a policy is trained for a new task – in addition to the extrinsic reward from the environment, the agent additionally gets a language command describing the task. At every step, the agent's trajectory so far is compared against the description of the task using the trained PixL2R model and the relatedness scores predicted by the model are used to generate intermediate rewards for reward shaping [Ng et al., 1999]. Section 4.2.2 describes this phase.

Note that the trained PixL2R model can be used during policy learning for a wide variety of downstream tasks, insofar as the objects and linguistic vocabulary in these tasks closely match the data used to train the PixL2R model. Thus, the cost of training PixL2R is amortized across all the downstream tasks.

4.2.1 PixL2R: Pixels and Language to Reward

First, a relatedness model – PixL2R – between a trajectory and a language is trained given paired data using supervised learning.

Network Architecture

The inputs to the network consist of a trajectory and a natural language description. Representing the trajectory using a single sequence of frames may be prone to perceptual aliasing and occlusion. Thus, our network architecture is designed to take multiple views as inputs. We use three different viewpoints in our experiments (see Figure 4.1), but it is straightforward to generalize to more or fewer viewpoints. In our ablation experiments, we compare the model described here with a model that takes a single viewpoint as input.

An independent CNN is used for encoding the sequence of frames from each viewpoint to generate a fixed size representation for each frame. These sequence of vectors are concatenated across the views to generate a single sequence of fixed size vectors, which is then passed through a two-layer LSTM to get an encoding of the entire trajectory.

The language description is converted to a one-hot representation, and passed through an embedding layer, followed by a two-layer LSTM. The outputs of the LSTMs encoding the trajectory and the language are then concatenated, and passed through a sequence of fully-connected layers to generate a relatedness score. See Figure 4.2 for a diagram of the neural network.

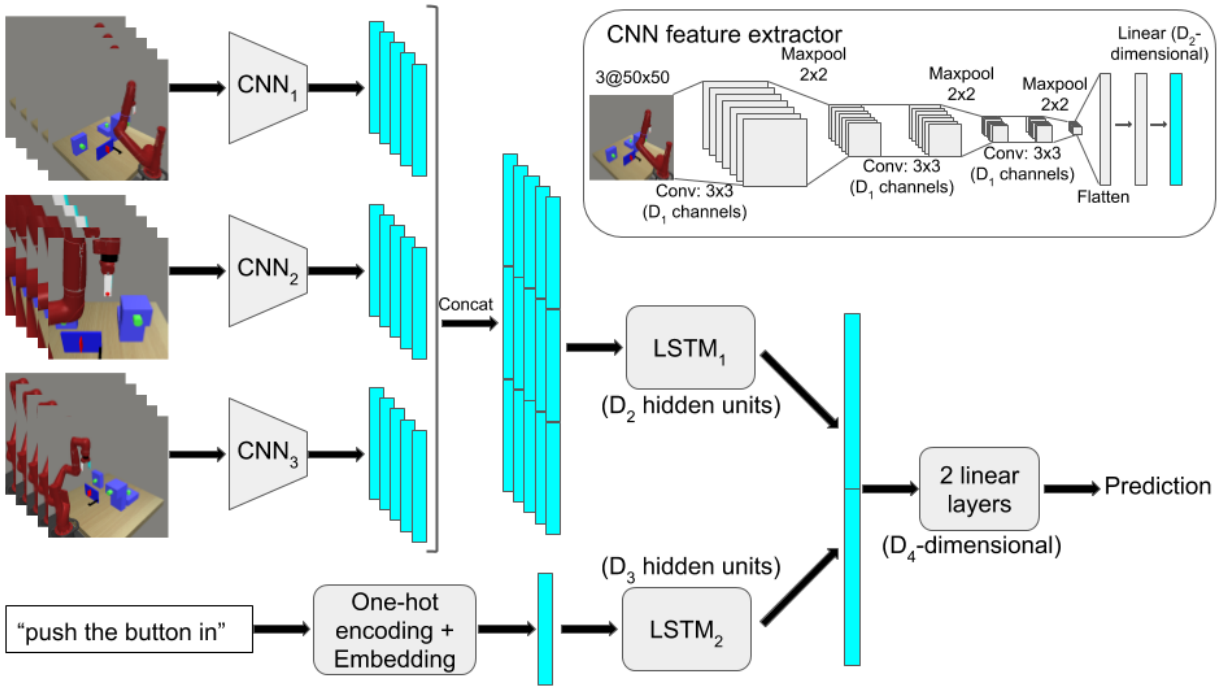


Figure 4.2: Neural network architecture: The sequence of frames from the three viewpoints are passed through three separate CNN feature extractors, and then concatenated across views. The sequence is then passed through an LSTM to obtain an encoding of the trajectory. The given linguistic description is encoded using a randomly initialized embedding layer followed by an LSTM. The outputs of the two LSTMs is concatenated and passed through a sequence of 2 linear layers to generate the final prediction.

Data Augmentation

Frame dropping. After sampling a trajectory, each frame is independently selected with a probability of 0.1. The resulting sequence of frames is passed through the network. This makes the training faster by reducing the input size, as well as making the network robust to minor variations in trajectories. During policy training, the trajectories are subsampled to keep 1 frame in every 10.

Partial trajectories. Since during policy training the model will have to make predictions for partial trajectories, we use partial trajectories during supervised training as well. Given a trajectory of length L, we sample $l \sim Uniform\{1, \ldots, L\}$, and use the first l frames of the trajectory.

Training Objectives

Classification. First, we trained the neural network using binary classification. The final output of the network is a two-dimensional vector, corresponding to the logits for the two classes – `RELATED` and `UNRELATED`. The network is trained to minimize the cross-entropy loss.

As mentioned above, we train the model with partial trajectories of different lengths to better match the distribution of trajectories that will be seen during policy learning. However, partial trajectories might sometimes be hard to classify as related or unrelated to the description, since it requires extrapolating the complete path the agent will follow. Our preliminary experiments suggest that these harder to classify examples affect learning—on unseen complete trajectories, a model trained with only complete trajectories has a lower error compared to a model trained on both complete and partial trajectories. This motivated us to experiment with an alternative regression setting described next.

Regression. In this setting, the model predicts a single relatedness score between the given trajectory and language, which is mapped to $[-1, 1]$ using the tanh() function. The ground truth score is defined as $s \cdot \frac{l}{L}$, where $s = 1$ for `RELATED` and $s = -1$ for `UNRELATED` pairs, l is the length of the incomplete trajectory and L is the length of the complete trajectory as described above. Thus, given a description, a complete related trajectory has a ground truth score of 1, while a complete unrelated trajectory has a score of -1. Shorter trajectories smoothly interpolate between these values, with very small trajectories having a score close to 0. The network is trained to minimize the mean squared error. Intuitively, this results in a small loss when the model predicts the incorrect sign on short trajectories. As the trajectories become longer, incorrect sign predictions result in higher losses.

The network is trained end-to-end using an Adam optimizer [Kingma and Ba, 2014].

4.2.2 Policy Learning Phase

Having learned a PixL2R model as described above, the relatedness scores from the model can be used to generate language-based intermediate rewards during policy learning on new scenarios. During policy training, the agent receives a natural language description of the

goal, in addition to the extrinsic reward from the environment. At every timestep, the PixL2R model is used to score trajectories executed by the agent against the given natural language description, to generate intermediate rewards. We used potential-based shaping rewards [Ng et al., 1999], which are of the form $F(s_t) = \gamma \cdot \phi(s_t) - \phi(s_{t-1})$, where s_t is the state at timestep t and $\phi : S \to \mathbb{R}$ is a potential function. In our case, s_t is the sequence of states encountered by the agent up to timestep t in the current episode.

For the classification setting, we used the potential function $\phi(s_t) = p_R(s_t) - p_U(s_t)$, where p_R and p_U are the probabilities assigned by the model to the classes `RELATED` and `UNRELATED` respectively. For the regression setting, the relatedness score predicted by the model is directly used as the potential for the state. Note that for both the settings, the potential of any state lies in $[-1, 1]$.

4.3 Domain and Dataset

4.3.1 Description of the Domain

We use Meta-World [Yu et al., 2019], a recently proposed benchmark for meta-reinforcement learning, which consists of a simulated Sawyer robot and everyday objects such as a faucet, windows, coffee machine, etc. Tasks in this domain involve the robot interacting with these objects, such as turning the faucet clockwise, opening the window, pressing the button on the coffee machine, etc. Completing these tasks requires learning a policy for continuous control in a 4-dimensional space (3 dimensions for the end-effector position, and the fourth dimension for the force on the gripper). While the original task suite consists of only one object in every task, we create new environments which contain one or more objects in the scene, and the robot needs to interact with a pre-selected object amongst those. In a sparse reward setting, the agent is given a non-zero reward only on successfully interacting with the pre-selected object. In the absence of any other learning signal, the agent might have to learn to approach and interact with multiple objects in the scene in order to figure out the correct object. Using natural language to describe the task in addition to the sparse reward helps alleviate this issue.

4.3.2 Data Collection

First, 13 tasks were selected from the Meta-World task suite. This gave us a total of 9 objects to interact with (for 4 objects, multiple tasks can be defined, e.g. turning a faucet clockwise or counter-clockwise). We then created 100 scenarios for each task as follows: In each scenario, the task-relevant object is placed at a random location on the table. Then, a new random location is sampled, and one of the remaining objects is placed at this position. This process is repeated until the new random location is close to an already placed object. This results in 1300 scenarios in total, with a variable number of objects in each scenario.

Linguistic descriptions were collected for each of these tasks using Amazon Mechanical Turk, with a total of 520 descriptions, which gives us 40 descriptions per task on average.

Given pairs of related ⟨trajectory, language⟩, positive examples were generated by pairing a scenario for one of the 13 tasks with a randomly sampled description of the corresponding

task. Negative examples were generated by pairing trajectories from a task with descriptions from unrelated tasks.

4.4 Experiments

4.4.1 Policy Training with Language-based Rewards

To empirically evaluate the effectiveness of PixL2R, the proposed framework was tested on 16 scenarios. For each scenario, a policy was trained using the PPO algorithm for 500,000 timesteps.

For each of the scenarios, a policy was trained using 4 different reward settings:

1. `Sparse`: The agent is given a reward of 1 for reaching the goal, and 0 in all other cases.

2. `Sparse+RGR`: The agent is given language-based rewards generated by PixL2R trained using the regression objective, in addition to the sparse rewards.

3. `Dense`: The agent is given hand-designed dense rewards defined in the original Meta-World benchmark.

4. `Dense+RGR`: The agent is given language-based rewards generated by PixL2R trained using the regression objective, in addition to the hand-designed dense rewards.

The resulting policy training curves are shown in Figure 4.3. Each curve is obtained by averaging over all runs (16 scenarios $\times$ 15 runs per scenario with different random seeds) for that reward type. The results verify that using language-based rewards in addition to sparse rewards result in higher performance on average than using only sparse ones. More interestingly, we find that using language-based rewards in conjunction with hand-designed rewards also results in an improvement. A plausible explanation is that the hand-designed dense rewards in Meta-World are suboptimal, since the reward function for each task consists of parameters that require tuning, highlighting the complexity of reward design mentioned earlier. This result motivates a novel paradigm wherein coarse dense rewards could be designed by hand, and the proposed framework can be used to get a further improvement in policy training efficiency by using natural language.

4.4.2 Ablations

Having established that policy learning works better with the language-based rewards, we ran ablation experiments to better understand our design choices and to inspect what factors most affect the efficiency of policy learning.

All the ablation experiments were performed with language-based rewards added to dense rewards, since most applications of RL in robotics currently use dense hand-designed rewards (which could be suboptimal for complex tasks).

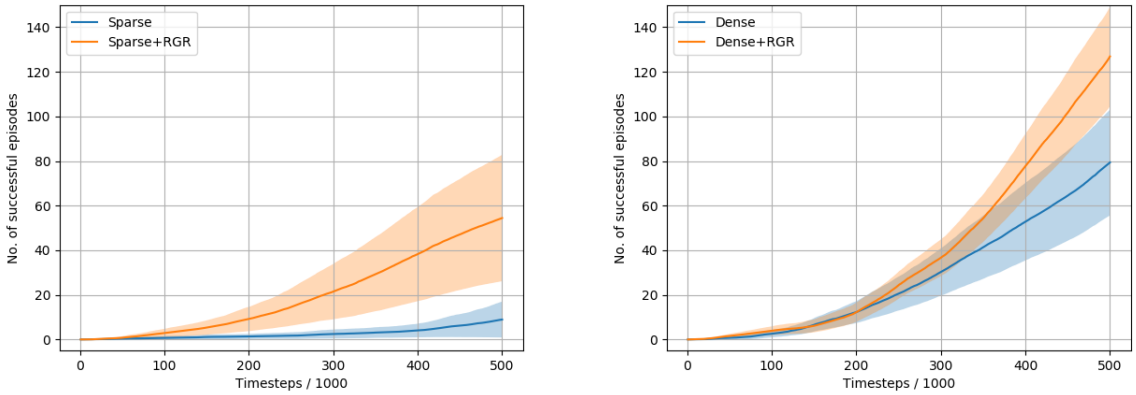


Figure 4.3: A comparison of policy training curves for different reward models. The shaded regions denote 95% confidence intervals.

1. **LastFrame**: To analyze whether using the full sequence of frames contains more information than the last frame, instead of using the sequence of frames in the trajectory, only the last frame of the trajectory was used, both for training the PixL2R model, as well as for policy training.

2. **MeanpoolLang**: To study if the temporal ordering of the words in the description is useful, the LSTM used to encode the language was replaced with the mean-pooling operation.

3. **MeanpoolTraj**: To study if the temporal ordering of the frames in the trajectory was useful, the LSTM used to encode the sequence of frames was replaced with the mean-pooling operation.

4. **SingleView**: To study the impact of perceptual aliasing and/or occlusion when using a single viewpoint, instead of using 3 viewpoints for the trajectory, only 1 viewpoint was used. A model was trained with *each* of the three viewpoints in the supervised learning phase, and the model with the best validation score was used for policy learning.

5. **Dense+CLS**: Instead of the regression loss, classification loss was used, to understand the benefit of using regression loss when working with partial trajectories.

For each ablation, the same setup was used as for `Dense+RGR`. The mean successful episodes across all runs are reported in Table 4.1 for each setting. Further, the p-values for Wilcoxon tests between each ablation and the `Dense` rewards is reported, from which we can make the following observations:

- Using only the last frame (`LastFrame`), or using mean-pooling instead of an LSTM to encode the language (`MeanpoolLang`) does not substantially affect policy learning

Setting	Mean Successful Episodes	p-value w.r.t. Dense
Dense	79.4	-
Dense+RGR	126.9	0.0340
LastFrame	133.5	0.0114
MeanpoolLang	138.3	0.0004
MeanpoolTraj	78.4	0.9601
SingleView	100.4	0.3789
Dense+CLS	102.0	0.6384

Table 4.1: Comparison of various ablations to the Dense+RGR model.

efficiency. In both these cases, the resulting model is still statistically significantly better than `Dense` rewards. Both of these results agree with intuition, since the progress in the task can be predicted using the last frame alone, and since the linguistic descriptions are not particularly complex in the given domain, simply looking at which words are present or absent is often sufficient to identify the task without using the ordering information between the words.

- Using mean-pooling instead of an LSTM to encode the sequence of frames (`Meanpool-Traj`) drastically reduces the number of successful episodes, and results in no statistically significant improvement over `Dense`. Again, this agrees with intuition, since it is not possible to infer the direction of movement of the robot from an unordered set of frames.

- Using a single view instead of multiple views (`SingleView`) results in a smaller increase in the number of successful episodes, which is no longer statistically significant over `Dense`. As mentioned earlier, using frames to represent trajectories requires addressing challenges such as perceptual aliasing and occlusion, and these ablation results suggest that using multiple viewpoints alleviates these issues.

- Using classification loss instead of regression (`Dense+CLS`) also leads to a drop in performance, again making the resulting improvements no longer statistically significant. This is consistent with our initial observation (Section 4.2.1), wherein, the learning problem becomes more difficult due to partial trajectories when the classification loss is used.

It is worth noting that while these ablations agree with intuition, and therefore suggest that the model is extracting meaningful information from trajectories and language descriptions, the performance of these variants depends crucially on the domain. For instance, an environment that is not fully observable in the last frame might show a significant drop in performance when using only the last frame instead of the full trajectory.

Chapter 5

Zero-shot Task Adaptation Using Natural Language

5.1 Introduction

In Chapters 3 and 4, we proposed approaches that use natural language to generate intermediate rewards for reinforcement learning, resulting in more efficient policy learning. As described in Chapter 2, an alternate paradigm to teach new tasks to learning agent in sequential decision making problems is using imitation learning [Argall et al., 2009]. This involves showing demonstration(s) of the desired task to the agent, which can then used by the agent to infer the demonstrator's intent, and hence, learn a policy for the task. However, for each new task, the agent must be given a new set of demonstrations, which is not scalable as the number of tasks grow, particularly because providing demonstrations is often a cumbersome process. We propose using language to reduce the burden of providing demonstrations.

To this end, we propose a novel setting—given a demonstration of a task (the *source task*), we want an agent to complete a somewhat different task (the *target task*) in a **zero-shot** setting, that is, without access to *any* demonstrations for the target task. The difference between the source task and the target task is communicated using language.

For example, consider an environment consisting of objects on different shelves of an organizer, as shown in Figure 5.1. Suppose the **source task** (top row) requires moving the green flat block from bottom-right to bottom-left, the blue flat block from middle-left to bottom-right, and then the green flat block from bottom-left to middle-left. The **target task** (bottom row) is similar, but in the final step, the green flat block should be moved to top-left instead. We posit that given a demonstration for the source task, and a free-form natural language description of the difference between the source and the target tasks, such as "In the third step, move the green flat block from bottom left to top left", an agent should be able to infer the goal for the target task. We propose a framework that can handle a diverse set of adaptations between the source and the target tasks, such as a missing step, an extra step, and swapping the final positions of two objects.

The environment has a similar structure to several real-world applications, where task adaptation using language could be particularly beneficial. For instance, consider the goal of

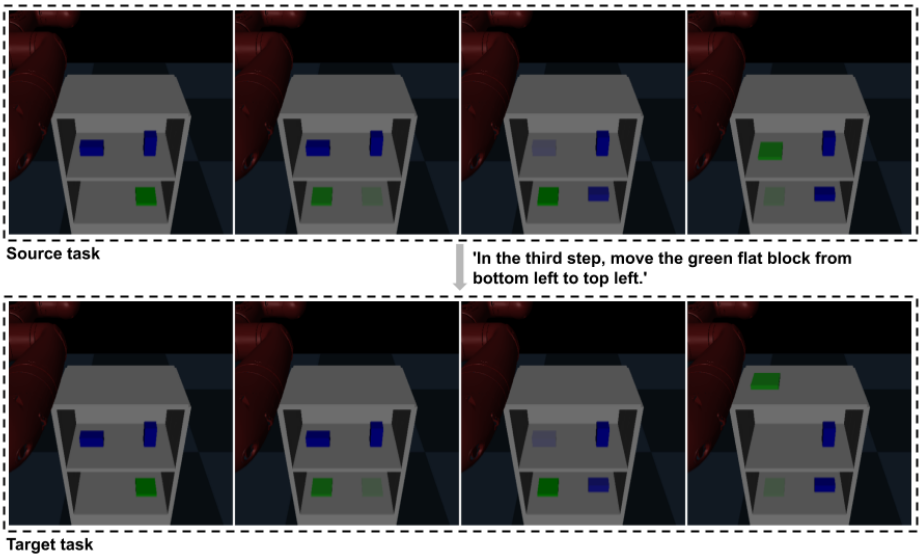


Figure 5.1: Example of the setting: The top row shows the *source task*, while the bottom shows the *target task*. Given a demonstration of the source task, and a natural language description of the difference between the two tasks such as "In the third step, move the green flat block from bottom left to top left.", our goal is to train an agent to perform the target task *without* any demonstrations of the target task.

building service robots for home environments. These robots must be able to learn a wide variety of tasks from non-expert users. Many tasks, such as cooking or assembly, involve a sequence of discrete steps, and such tasks could have several variations, like different cooking recipes or assembling different kinds of furniture. Being able to demonstrate one (or a few) of these tasks, and then communicating the difference between the demonstrated task(s) and other similar tasks could significantly reduce the burden of teaching new skills for the users.

These problems involve planning/control at 2 levels—high-level planning over the steps, and low-level control for executing each step. Since our proposed algorithm focuses on the high-level planning, we illustrate our approach on the simple environment shown in Figure 5.1, where the low-level control is abstracted away. However, our framework is general, and can be combined with approaches that perform low-level control.

The proposed setting is challenging for several reasons. First, most existing approaches in imitation learning and instruction-following infer the goal for a target task from a demonstration or an instruction, respectively. However, in our setting, neither of these modalities is sufficient by itself, and the agent must be able to combine complementary information from the source demonstration(s) and the natural language descriptions, in order to infer the goal for the target task. Second, in order to understand the natural language description, the agent must be able to map concepts in the description to objects and actions, a problem known as symbol grounding [Harnad, 1990]. Finally, in order to be scalable, we intend to learn a purely data-driven model that can does not require engineering features for the language or the environment, and can learn to infer the goal for the target task directly from data.

Further, since the proposed setting requires combining information from both the demon-

stration and the language, it can therefore serve as an important step towards building systems for more complex tasks which are difficult to communicate using demonstrations or language alone.

We introduce the Language-Aided Reward and Value Adaptation (LARVA) model that takes in a dataset of source demonstrations, linguistic descriptions, and either the reward or optimal value function for the target task, and is trained to predict the reward or optimal value function of the target task given a source demonstration and a linguistic description. The choice between reward and value prediction could be problem-dependent—for domains with complex transition dynamics, learning a value function requires reasoning about these dynamics, and therefore, it might be better to use LARVA for reward prediction, with a separate policy learning phase using the predicted rewards; for domains with simpler dynamics, a value function could be directly predicted using LARVA, thereby removing the need for a separate policy learning phase.

We describe the approach and the main experimental results below; for full details, see [Goyal et al., 2021b].

5.2 Problem Definition

Consider a **goal-based task**, which can be defined as a task where the objective is to reach a designated goal state in as few steps as possible. It can be expressed using the standard Markov Decision Process (MDP) formalism, as $M = \langle S, A, P, R, \gamma, g \rangle$, where S is the set of all states, A is the set of all actions, $P : S \times A \times S \to [0, 1]$ is the transition function, $R : S \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor, and $g \in S$ is the unique goal state.

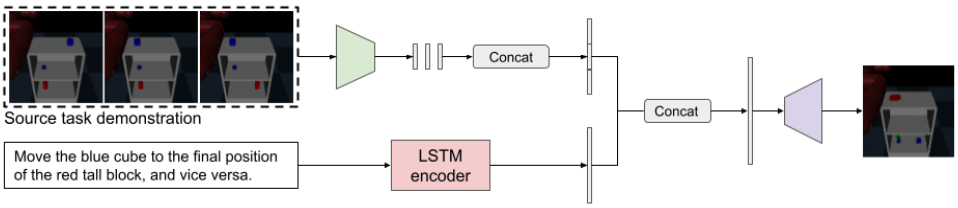
Further, $V^*_R : S \to \mathbb{R}$ denotes the optimal value function under the reward function R, and can be used to act optimally.

The reward function for a goal-based task can be defined as $R(s) = \mathbb{1}[s = g]$, where $\mathbb{1}[\cdot]$ is the indicator function. Thus, for $\gamma < 1$, an optimal policy for a goal-based task must reach the goal state g from any state $s \in S$ in as few steps as possible.

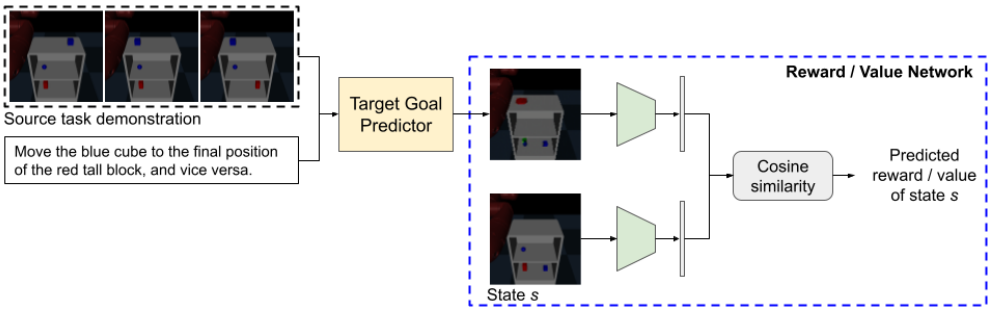
Let $\mathcal{T} = \{M_i\}_{i=1}^N$ be a **family** of goal-based tasks M_i, each with a distinct goal g_i, and the reward function R_i defined as above. The set of states S_i and actions A_i, the transition functions P_i, and the discount factors γ_i across different tasks may be related or unrelated [Kroemer et al., 2019].

For instance, in the environment shown in Figure 5.1, a goal-based task consists of arranging the objects in a specific configuration defined by a goal state g, while $\mathcal{T}$ is the set of all multi-step rearrangement tasks in the environment.

Let T_{src}, $T_{tgt} \in \mathcal{T}$ be two tasks, and L be a natural language description of the difference between the tasks. Given a demonstration for the source task T_{src}, and the natural language description l, our objective is to train an agent to complete the target task T_{tgt} in a **zero-shot setting**, i.e., without access to the reward function or demonstrations for the target task.



(a) The target goal predictor takes in a source demonstration and a description to predict the goal state for the target task.



(b) The reward / value network takes the predicted goal state from the target goal predictor, and another state s from the target task to predict the reward or value of the state s under the target reward function.

Figure 5.2: Neural network architecture for LARVA

5.3 LARVA: Language-Aided Reward and Value Adaptation

We propose Language-Aided Reward and Value Adaptation (LARVA), a neural network that takes in a source demonstration, τ_{src}, the difference between the source and target tasks described using natural language, l, and a state from the target task, $s \in S_{tgt}$, and is trained to predict either $R_{tgt}(s)$, the reward for the state s in the target task, or $V^*_{R_{tgt}}(s)$, the optimal value of the state s under the target reward function R_{tgt}.

We assume access to a training set $\mathcal{D} = \{(\tau^i_{src}, l^i, g^i_{tgt})\}_{i=1}^N$, where τ^i_{src} is a demonstration for the source task of the i^{th} datapoint, l^i is the natural language description of the difference between the source task and the target task for the i^{th} datapoint, and g^i_{tgt} is the goal state for the target task. The details of the dataset and the data collection process are described in Section 5.4.

5.3.1 Network Architecture

We decompose the learning problem into two subproblems: (1) predicting the goal state for the target task given the source demonstration and the language, and (2) predicting the reward / value of state s given the goal state of the target task. As such, we propose a neural network architecture that consists of two modules: (1) Target Goal Predictor, and (2) Reward / Value Network (see Figure 5.2). This decomposition allows for additional supervision of the target goal predictor, using the ground-truth goal state for the target task.

Target Goal Predictor

Given a sequence of images representing a source demonstration (τ_{src}), and a natural language description of the difference between the source and the target task (l), the target goal predictor module is trained to predict the goal state of the target task (g_{tgt}).

Demonstration Encoder. Each image in the source demonstration is first passed through a convolutional neural network to obtain a D_1-dimensional vector representation, where D_1 is a hyperparameter tuned using the validation set. The resulting sequence of vectors is padded to the maximum demonstration length (T_{max}) in the training data, and the vectors are then concatenated to obtain a single $T_{max} \cdot D_1$-dimensional vector. [1]

Language Encoder. The natural language description is first converted into a one-hot representation using a vocabulary (see Sections 5.4.2 for details about the vocabulary), which is then passed through a two-layer LSTM module to obtain a vector representation of the description. The hidden size of the LSTM is set to D_2, which is tuned using the validation set.

Target Goal Decoder. The vector representations of the source demonstration and the natural language description are concatenated, and the resulting vector is passed through a deconvolution network to obtain an image representation $\hat{g}$ of the target goal state.

Reward / Value Network

The reward or value network takes the predicted target goal state $\hat{g}$ and another state s from the target task as inputs, and is trained to predict the reward or the value respectively of state s under the target reward function. The predicted goal state $\hat{g}$ and the state s are encoded using the same CNN encoder (i.e. shared weights) used for encoding the demonstration frames in the target goal predictor, to obtain D_1-dimensional vector representations. The reward or value of state s is computed as the cosine similarity between the vector representations of $\hat{g}$ and the state s. We represent the ground-truth reward or value as $f(s)$, while the network prediction as $\hat{f}(s)$.

5.3.2 Training

To train the model, we assume access to a dataset $\mathcal{D} = \{(\tau^i_{src}, l^i, g^i_{tgt})\}_{i=1}^N$. Using the goal state for the i^{th} target task, the reward function and the optimal value function for the target task can be computed, which is used to supervise the model training as described below.

[1] We also experimented with an LSTM and a transformer to encode the sequence of vectors, but obtained significantly worse performance compared to a simple concatenation. A possible explanation for this behavior is that encoding the frames into a single vector independently of the language may make it harder to associate information in language with that in individual frames, suggesting that cross-attention between language and the frames might be required. Our preliminary experiments with attention-based models did not work well, but a more thorough anaysis is needed.

Training Objectives

Mean Squared Error. Since we want the model to regress to the true reward or value $f(s)$ for states $s \in S^i_{tgt}$, a natural choice for the loss function is the mean squared error (MSE), $\mathcal{L}_{\text{f}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{s\in S^i_{tgt}}(f(s) - \hat{f}(s))^2$.

Target Goal Supervision. Further, we additionally supervise the Target Goal Predictor using the true goal state g^i_{tgt} for the i^{th} target task, using an MSE loss, $\mathcal{L}_{\text{goal}} = \frac{1}{N}\sum_{i=1}^{N}(g^i_{tgt} - \hat{g}^i_{tgt})^2$.

Thus, the overall training loss is given by $\mathcal{L} = \mathcal{L}_{\text{f}} + \lambda\mathcal{L}_{\text{goal}}$, where λ is an hyperparameter tuned using a validation set.

5.4 Environment and Dataset

In this section, we describe the environment we use in our experiments. While the framework described above is fairly general, in this work, we focus on a simpler setting that is more amenable to analysis. Specifically, we assume discrete state and action spaces S and A, and deterministic transitions, i.e., $P(s, a, s') \in \{0, 1\}, \forall(s, a, s') \in S \times A \times S$.

5.4.1 The Organizer Environment

We propose the Organizer Environment, which consists of an organizer with 3 shelves. There are 8 distinct objects, and each object can take one of 3 colors (red, blue, or green), giving a total of 24 distinct colored objects. Objects can be placed in each shelf, either to the left or the right, resulting in a total of 6 distinct positions, `POSITIONS` = {`Top-Left`, `Top-Right`, `Middle-Left`, `Middle-Right`, `Bottom-Left`, `Bottom-Right`}.

Objects can be placed in different configurations to create different states. In our experiments, we use tasks with 2 or 3 objects. The total number of states with 2 or 3 objects (i.e. $|\bigcup_{T\in\mathcal{T}} S_T|$) is 285,120. The action space A is common across all tasks, and consists of 30 move actions, $\texttt{MOVE}(p_i, p_j), p_i, p_j \in \texttt{POSITIONS}, p_i \neq p_j$. Finally, there is a `STOP` action that indicates the termination of an episode.

5.4.2 Language Data

In this work, we experiment with 6 types of adaptations: (1) moving the same object in the source and target tasks, but to different final positions; (2) moving a different object in the source and target tasks, but to the same final position; (3) moving two objects in the source and target tasks, with their final positions swapped in the target task; (4) deleting a step from the source task; (5) inserting a step to the source task; and (6) modifying a step in the source task.

For each pair of source and target tasks in the dataset, we need a linguistic description of the difference between the tasks. We start by generating linguistic descriptions using a set of templates, such as, "Move `obj1` instead of `obj2` to the same position." We ensure that for most of these templates, the target task cannot be inferred from the description alone,

Table 5.1: Examples of template-generated and natural language descriptions collected using AMT.

	Template	**Natural language paraphrase**
1.	Move the cylinder to middle left.	Place the cylinder in the middle left
2.	Move the red tall block to the final position of green long block.	Place the red tall block in the green longs blocks final position
3.	Skip the third step.	do not do the third step
4.	In the first step, move the green tall cylinder from bottom right to bottom left.	for the first step, put the green tall cylinder in the bottom left position
5.	Move blue tall cylinder from bottom left to middle left after the first step.	For the second step move the blue tall cylinder to the middle left
6.	Move the blue cube to the final position of blue tall cylinder.	Swap the blue cube with the red cube on bottom shelf.
7.	Move blue tall block from top right to bottom left after the second step.	Move blue tall square from upper option to base left after the subsequent advance.

and thus, the model must use both the demonstration of the source task and the linguistic description to infer the goal for the target task.

Next, we collected natural language for a subset of these synthetic (i.e. template-generated) descriptions using Amazon Mechanical Turk (AMT). Workers were provided with the template-generated descriptions, and were asked to paraphrase these descriptions.

We applied a basic filtering process to remove clearly bad paraphrases, such as those with 2 words or less, and those that were identical to the given descriptions. We did not make any edits to the paraphrases, like correcting for grammatical or spelling errors. Some examples of template-generated and natural language descriptions obtained using AMT are shown in Table 5.1.

It can be observed that while the first four rows in the table are valid paraphrases, the remaining three paraphrases could be ambiguous depending on the source and target tasks. For instance, in row 5, the target task involves an *extra* step after the first step, while the natural language paraphrase could be interpreted as *modifying* the second step. In row 6, the natural language description is not a valid paraphrase, while in row 7, the paraphrase is difficult to comprehend. We manually analysed a small subset of the collected paraphrases, and found that about 15% of the annotations were ambiguous / non-informative. Some of this noise could be addressed by modifying the data-collection pipeline, for example, by providing more information about the source and target tasks to humans, and filtering out non-informative / difficult to comprehend paraphrases.

Table 5.2: Success rates of different models

	Experiment	Success rate (%)	
		Synthetic	**Natural**
1.	LARVA; reward prediction	97.8	75.7
2.	LARVA; value prediction	97.7	73.3
3.	LARVA; no target goal supervision	20.0	2.7
4.	LARVA; no language	20.7	22.3
5.	LARVA; no source demonstration	4.2	3.3
6.	NN without decomposition	1.8	1.0
7.	LARVA: Compostionality – red box	87.6	62.4
8.	LARVA: Compostionality – blue cylinder	89.4	65.9

5.5 Experiments

5.5.1 Details about the setup

Dataset. For each adaptation, 6,600 pairs of source and target tasks were generated along with the template-based descriptions. Of these, 600 templates were used for each adaptation to collect natural language descriptions using Amazon Mechanical Turk. Thus, our dataset consisted of 6,600 examples in total for each adaptation, of which 6,000 examples consisted of synthetic language, and 600 consisted of natural language. Of the 6,000 synthetic examples per adaptation, 5,000 were used for training, 500 for validation, and the remaining 500 for testing. Similarly, of the 600 natural language examples per adaptation, 500 were used for training, 50 for validation, and 50 for testing. This gave us a training dataset with 33,000 examples, and validation and test datasets with 3,300 examples each, across all adaptation types.

Evaluation Metrics. For each experiment, the trained model predicts the reward or value of the given state s. When using the value function, the trained network is used to predict the values for all states $s \in S_{tgt}$. When using the reward function, the trained network is used to predict the rewards for all states $s \in S_{tgt}$, from which the optimal value function is computed using dynamic programming. In both cases, if the state with the maximum value matches the goal state for the target task, g_{tgt}, the task is deemed to be successful. We train the models using the entire training set (i.e. both synthetic and natural language examples across all adaptations), and report the percentage of successfully completed target tasks for both synthetic and natural language descriptions. For each experiment, we tune the hyperparameters on the validation set, and report the success rate on the test set corresponding to the setting yielding the maximum success rate on the validation set.

5.5.2 Results

In this section, we describe the performance of our full model, along with various ablations. Our results are summarized in Table 5.2.

First, we evaluate our full LARVA model, with both reward and value predictions (rows 1 and 2 in Table 5.2). In both cases, the models result in successful completion of the target task more than 97% of the time with synthetic language, and more than 73% of the time with natural language. The drop in performance when using natural language can partly be attributed to the 15% of paraphrases that are potentially ambiguous or uninformative, as discussed in Section 5.4.2, while the remaining performance gap is likely because natural language has more variation than synthetic language. Better data collection and more complex models could be explored to bridge this gap further.

The similar performance when predicting rewards and values is not unexpected—we observed in our experiments that training the target goal prediction module was more challenging than training the the reward or value networks. Since the target goal prediction module is identical for both reward and value predictions, the performance in both cases is upper-bounded by the performance of the target goal prediction module. For domains with complex dynamics, reward and value prediction might result in significantly different sucess rates.

Next, we discuss ablations of our model. We present the results only with value prediction, since as noted, both reward and value predictions perform similarly.

1. To study the effect of target goal supervision for training the target goal predictor, we remove $\mathcal{L}_{goal}$, optimizing the network using the ground-truth values only. Row 3 in Table 5.2 shows that this drastically degrades performance, confirming the efficacy of target goal supervision.

2. To ensure that most tasks require using information from both the source demonstration and the natural language description, we run unimodal baselines, wherein the network is provided with only the source demonstration (row 4) or only the language (row 5). As expected, both the settings result in a substantial drop in performance. Interestingly, using only the source demonstration results in over 20% successful completions. This is because given the set of adaptations, the source demonstration constrains the space of target configurations more effectively (e.g. if the source task consists of three steps, the target task must contain at least two of those steps, since source and target tasks differ by only one step for all adaptations).

3. Finally, we experiment with an alternate neural network architecture, that does not decompose the learning problem into target goal prediction and value prediction. The source demonstration, the language, and the target state s are all encoded independently, and concatenated, from which the value for state s is predicted. Row 6 shows that the resulting model achieves a very low success on target tasks, demonstrating the importance of decomposing the learning problem as in LARVA.

In the experiments so far, the data were randomly partitioned into training, validation, and test splits. However, a key aspect of using language is the ability to *compose* concepts. For instance, humans can learn concepts like "blue box" and "red cylinder" from independent examples, and can recognize a "blue cylinder" by composing these concepts without ever having seen examples of the new concept.

Table 5.3: Success rates (%) when using varying amounts of synthetic and natural language data to train LARVA. The row labels show the number of natural language examples used while the column labels show the number of synthetic language examples used.

(a) Synthetic language test set

# natural	# synthetic			
	0	**7,500**	**15,000**	**30,000**
0	-	83.8	93.2	97.3
750	3.0	85.8	93.4	97.2
1,500	5.9	85.8	91.9	96.8
3,000	30.1	88.1	93.6	97.7

(b) Natural language test set

# natural	# synthetic			
	0	**7,500**	**15,000**	**30,000**
0	-	48.7	46.3	51.0
750	1.7	60.7	58.7	65.7
1,500	4.3	63.0	64.0	73.3
3,000	29.3	68.7	72.0	73.3

To evaluate whether our proposed model can exhibit the ability to compose concepts seen during training, we create 2 new splits of our data—in both the splits, the training data consists of all datapoints that do not contain any blue cylinders or red boxes. In the first split, the validation set consists of all datapoints containing blue cylinders, while the test set consists of all datapoints containing red boxes. In the second split, the validation and test sets are swapped.[2]

We train LARVA on these new splits (using value prediction), and report the success rate on the test set in Table 5.2, rows 7 and 8. As can be observed, our model is able to successfully complete a large fraction of the target tasks, by composing concepts seen during training, however, there is room for further improvement by using richer models.

In order to better understand the amount of data needed, we trained LARVA with varying amounts of synthetic and natural language training examples (using value prediction). The results are summarized in Table 5.3.

Unsurprisingly, on the synthetic language test set, the number of natural language examples only makes a difference when the number of synthetic language examples in the training set is small. The results on the natural language test set are more informative. In particular, if no natural language examples are used for training, the model is only able to successfully complete about 50% of the tasks, even as the amount of synthetic language data is increased. Furthermore, using 1,500 natural language examples instead of 3,000 with 30,000 synthetic language examples results in a comparable performance as the full set. Similarly, halving the amount of synthetic language data (i.e. 15,000 examples instead of 30,000) when using the full natural language set results in only a small reduction in performance. However, it is clear that some amount of natural training data is needed to successfully generalize to natural language test cases.

These results suggest that using additional synthetic language or natural language data compared to our full set will likely not result in a significant performance improvement, and thus, improving the performance when using natural language requires filtering out low quality natural language data, and using richer models.

[2]Datapoints containing both a blue cylinder and a red box are discarded.

Chapter 6

Proposed Work

6.1 Task Adaptation

6.1.1 Neurosymbolic Model

The approach presented in Chapter 5 assumes that the tasks are goal-based, and is therefore limited in its applicability. Further, the interaction between the encoded source demonstration and the encoded linguistic description is achieved using a simple concatenation operation. Finally, the predicted rewards or values were not used to learn a policy in the experiments.

More importantly, the proposed model uses a generic neural network architecture, that is not designed with problem-specific inductive biases. For instance, most tasks can be modeled as performing a sequence of operations on a set of entities, and language can be seen as communicating which operations need to be applied on which entities. To this end, we propose using a neurosymbolic model, inspired by [Goyal et al., 2021a], which we describe next. We first describe the approach for discrete domains, and then discuss how the approach can be extended to handle continuous domains.

Consider a dataset consisting of $(\tau_{src}, l, \tau_{tgt})$, where τ_{src} is the source demonstration, τ_{tgt} is the target demonstration, and l is the description of how the source and the target tasks differ, as described in Chapter 5. The first step involves learning M *entities* and N *production rules*, such that states in the demonstration data can be represented using the entities, and state transitions using the production rules. A production rule operates on an entity, thereby changing its state. Thus, actions and environment dynamics can be represented using a set of production rules. In our proposed model, each entity is a learnable feature vector, and each production rule consists of a learnable vector to decide when the rule is applied, and a neural network that encodes the operation performed by the rule on an entity.

Next, given the source task represented using the entities and sequence of production rules, the linguistic description, and a state from the target task, the objective is to predict the production rule to apply, and the entity (or entities) to apply it on, which can be formulated as N-way and M-way classification problems respectively to learn an *adaptation model*.

The learned entities, production rules, and adaptation model can be used at test time, to predict the sequence of states in the target task, which can be considered as a state-only demonstration for the target task. An imitation-from-observation approach, like GAIfO [Torabi et al., 2018], can be used to train a policy from this demonstration.

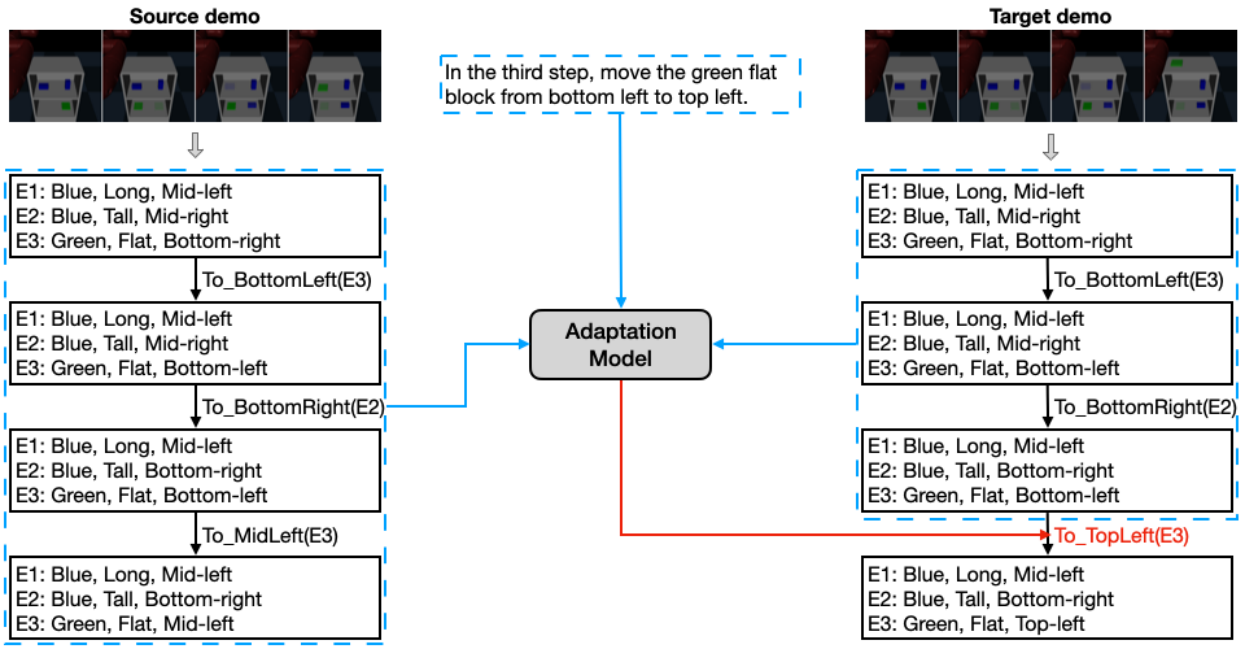


Figure 6.1: Neurosymbolic model for task adaptation: The source and target demonstrations are first represented using a set of entities, and production rules to transition between the states. An *Adaptation Model* is then trained to predict the production rule to apply given the source demonstration, the language, and partial target demonstration. The trained Adaptation Model can be used to predict the target demonstration step-by-step at test time. For ease of exposition, we show the entities and relations in the figure symbolically; in the model, they will be represented using vectors and neural networks (Section 6.1.1)

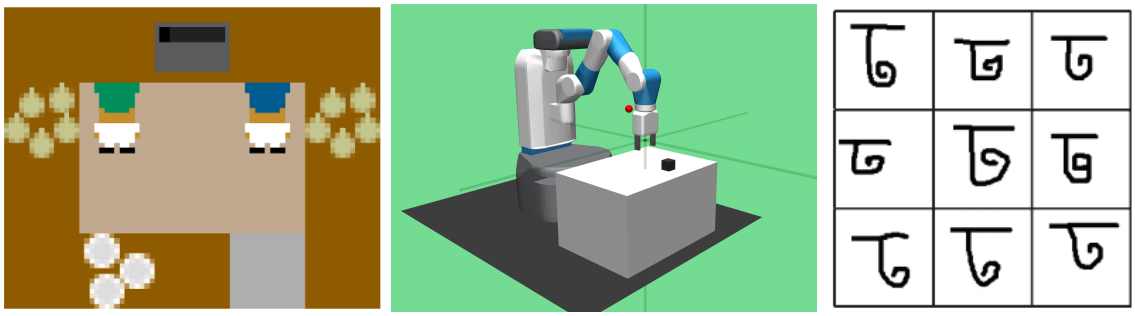


Figure 6.2: Domains for task adaptation. Left: GridWorld Cooking, Center: Block Pushing, Right: Drawing.

To extend this approach to continuous domains, we propose the following modifications. First, the demonstrations in the training data can be segmented, using an approach like [Niekum et al., 2012], to get a sequence of discrete states, which can be used in the framework above. Each segment can then be represented using a production rule. Second, to encode motion properties, each production rule can be augmented with an additional vector. This vector is then learned with jointly with the other model parameters.

6.1.2 Policy Adaptation

The approach proposed in Chapter 5 and Section 6.1.1 learn a policy for the target task purely from the predicted demonstration for the target task. However, in many real-world scenarios, an agent might already have a policy to solve the source task. In such cases, it would likely be beneficial to use the information in the source policy to learn the target policy. We intend to explore this direction as follows.

Assuming access to a dataset consisting of $(\tau_{src}, l, \tau_{tgt})$ as before, we can train policies π_{src} and π_{tgt} for the source and target tasks using inverse reinforcement learning. Next, we train a neural network T that takes in a state s as input, a linguistic description l of how the source and the target tasks differ, and outputs the difference between the action distributions of the source and the target tasks, $\pi_{tgt}(a|s) - \pi_{src}(a|s)$. Finally, given the policy network of a new source task and a linguistic description at test time, we can use the outputs from T to provide additional supervision to train a policy network for the target task, augmenting the approach described in Section 6.1.1.

We will test the following hypotheses:
(1) Does the additional supervision allow learning a better policy for the target task?
(2) Does the additional supervision allow learning a policy using fewer interactions with the environment?

Note that transferring policies from source to target tasks assumes that the dynamics for the tasks are identical. While this may not always hold, in a lot of real-world applications, such as a service robot in a home environment, the dynamics are largely determined by the robot's architecture and real-world physics, which are invariant across tasks.

6.1.3 Evaluation

In addition to the Organizer environment introduced in Chapter 5, we plan to test the aforementioned ideas on the following domains. See Figure 6.2 for illustrations.

- **GridWorld Cooking**: Inspired by [Carroll et al., 2019], we plan to create a cooking-based gridworld domain, with food and appliances randomly placed in the environment. This domain has semantically meaningful steps compared to the Organizer environment, and enables fine-grained state changes (such as heating a food item for different lengths of time can result in different outcomes), which would allow testing our approach on a discrete domain with richer set of natural language descriptions.

- **Block Pushing**: In this domain, a robot is tasked with moving objects on a table-top. Different initial and goal configurations can be created by adding multiple blocks in various positions and orientations. Language can be used to communicate modifications

in the goal state (e.g. different position or orientation of some blocks), constraints (e.g. "Don't move the red block"), or hints (e.g. "Move the blue block to uncover the red block"). We plan to build on the Fetch MuJoCo domain [Plappert et al., 2018]. This domain will allow us to test the model in a continuous control setting, with a relatively limited linguistic variation.

- **Drawing**: While block pushing allows testing the proposed ideas on continuous control, the linguistic variation in describing the source and target tasks is somewhat limited. As such, we will use the Omniglot dataset [Lake et al., 2015], which consists of a wide variety of handwritten characters from different languages, wherein two instances of the same character can be treated as source and target tasks respectively. Alternatively, two distinct but visually similar characters can be treated as source and target tasks. An agent that can move in the 2D plane is tasked with drawing the target instance given the source instance and a linguistic description of the difference between the two. We expect the linguistic descriptions in this domain to be significantly richer, to express low-level trajectory shapes. This will allow us to test our approach on both continuous control, and rich natural language. To define the reward for a character drawn by the agent, we will use the Sinkhorn distance [Cuturi, 2013].

Evaluating the proposed methods on these domains would demonstrate their applicability to both discrete and continuous action spaces, as well as their ability to handle both simple and richer natural language.

Having completed the proposed work in this section, the dissertation would make the following contributions: (1) frameworks that allow using language as an auxiliary signal by generating additional rewards in an RL setting, (2) frameworks that allow learning a new task given the demonstration of a related task, by using language to communicate the difference. These frameworks would be demonstrated on discrete and continuous domains, and would allow using natural language as an auxiliary signal for task specification in RL and IL in ways that were largely underexplored so far.

6.2 Long-term Directions

In this section, we describe some long-term directions which we might explore, time permitting.

6.2.1 Policy Regularization

Consider the two frames from the Atari game Montezuma's Revenge shown in Figure 6.3. While these frames look very different visually, the agent should take the same action in both states, namely jumping to collect the key or the orb, respectively. Thus, a policy network that is trained to play the game must output similar action distributions for both these states. This motivates the following research question—given linguistic instructions for how to act in different states, can we use the information to train a policy network more efficiently.

Since the policy network must output similar action distributions for states that require similar actions, we can constrain the network to produce similar intermediate representations

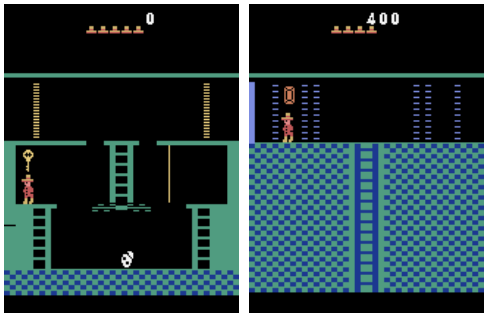


Figure 6.3: Frames from Montezuma's Revenge

for such states. This can be achieved by having an auxiliary loss term that penalizes the network for producing different representations for states with similar linguistic instructions.

More formally, consider a dataset $\{(s_i, l_i)\}_{i=1}^N$, with states and corresponding linguistic descriptions. A standard RL algorithm finds the optimal policy $\pi_\theta^*(s)$ by solving

$$\pi^* = \arg\min_\pi J(\pi)$$

where $J(\pi)$ is the expected sum of future rewards. The policy network $\pi(s)$ can be viewed as the composition of a state encoder $f(\cdot)$, and an action predictor $g(\cdot)$, such that $\pi(s) = g(f(s))$. We can regularize the state encoder $f(s)$ using the language data by requiring that $f(s_1)$ and $f(s_2)$ be similar if l_1 and l_2 are close, for datapoints (s_1, l_1) and (s_2, l_2).

Intuitively, this would encourage the network to encode only those features of the state that are relevant for action prediction.

There are several questions worth studying in this setting:

1. What are the different choices for the regularization term, and how do they compare with each other?

2. How much language data is needed to obtain statistically significant gains?

3. How to interleave the regularization in the original RL optimization problem?

6.2.2 Bayesian Inference

Bayesian Inverse Reinforcement Learning (BIRL) is an approach that uses Bayesian inference to model the posterior distribution of rewards, given demonstrations [Ramachandran and Amir, 2007]:

$$p(R|\mathcal{D}) = p(\mathcal{D}|R)p(R)$$

where $\mathcal{D}$ is the set of demonstrations, and R is the reward function. The likelihood function $p(\mathcal{D}|R)$ is modelled as a Boltzmann distribution, and the prior $p(R)$ over the reward functions can be defined as desired.

Separately, [MacGlashan et al., 2015] have developed an analogous approach that uses Bayesian inference to model the posterior distribution of rewards, given language. Their likelihood function $p(\mathcal{L}|R)$ (where $\mathcal{L}$ is the language command) is based on a statistical machine translation approach, where the reward (i.e. goal configuration) is represented using

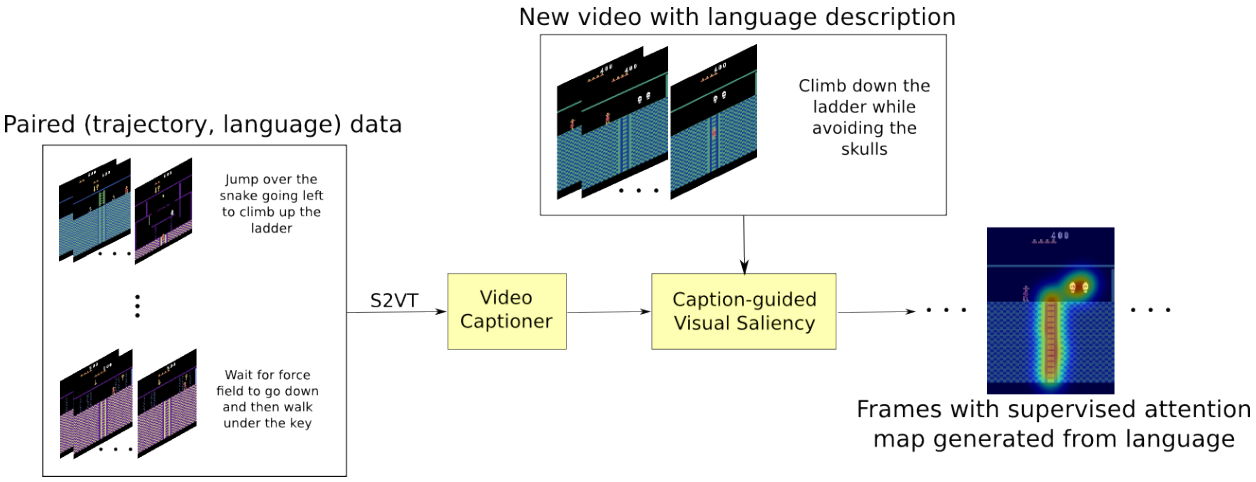


Figure 6.4: A schematic diagram of language-based supervised attention. The paired (trajectory, language) data is used to first train a video captioning model using the S2VT approach. Then, given a new video with language description, Caption-guided Visual Saliency can be used, along with the trained video captioning model, to generate a language-based saliency map over the video frames, which can be used as supervised attention.

a formal language, and a translation model is learned to estimate the probability of natural language given a formal language:

$$p(R|\mathcal{L}) = p(\mathcal{L}|R)p(R)$$

These approaches can be combined to model the posterior distribution of rewards, given both language and demonstrations. Different independence assumptions would result in different expressions for the posterior distributions. For instance, assuming $p(\mathcal{D}, \mathcal{L}|R) = p(\mathcal{D}|R)p(\mathcal{L}|R)$ results in

$$p(R|\mathcal{D}, \mathcal{L}) = p(\mathcal{D}, \mathcal{L}|R)p(R) = p(\mathcal{D}|R)p(\mathcal{L}|R)p(R)$$

6.2.3 Supervised Attention

We propose to use language as supervision for attention [Zhang et al., 2016], that is, attending to parts of the state and/or the demonstrations to which the language explicitly refers. Consider a scenario where the demonstrator shows pouring the contents of a red cup into a blue cup, in the presence of other background objects, like a coffee machine. Learning only from demonstrations would require the agent to extract features from the state (i.e. the image from the agent's camera) that capture the positions of the cups but ignore the coffee machine. A common approach allowing neural network models to focus on relevant state features is an "attention mechanism" , which is typically learned jointly with the end task in an unsupervised manner [Bahdanau et al., 2014, Xu et al., 2015, Vaswani et al., 2017, Yang et al., 2016]. As such, these techniques have a high sample complexity.

Using language to learn the attention model in a supervised manner can enable relevant state features to be inferred more accurately and efficiently by focusing attention on aspects of the world referred to by the language. The learned attention model can then be used to help learn the final task. We briefly describe our proposed framework below.

Language Grounding. In order to use language to generate attention maps over the states, we need to learn a model that grounds the language in the perception of the environment. We plan to use a video captioning model like Sequence-to-Sequence Video to Text (S2VT) [Venugopalan et al., 2015a]. The video captioning model can then be used with techniques for generating visual explanations for neural networks, such as *caption-guided visual saliency* [Ramanishka et al., 2017], which take a video captioning neural network and a given input/output pair, and produce heat maps over the input frames denoting which parts most influenced the computed output. See Figure 6.4 for an illustration.

Attention-based Auxiliary Loss. Given a new demonstration with a paired language description, we propose to generate heat maps using the above approach, which can then be used as an additional supervision signal for the attention model. Specifically, to encourage the agent to attend to the parts of the state referred to by the language, we will use an auxiliary loss to match the feature maps generated by intermediate layers of a convolutional policy network with the heat maps generated from language. It has been shown that such an auxiliary loss, when used with heat maps generated from gaze (instead of language), significantly improves the sample efficiency of policy training [Saran et al., 2020].

By using language to provide this additional supervision, we expect the feature-extraction layers to converge to an effective solution more quickly, allowing the downstream network to focus on relevant features of the state and the trajectory even during initial phases of learning, thereby reducing the sample complexity of learning the end task.

Chapter 7

Conclusion

Having general-purpose intelligent robots that could work alongside humans requires specifying new tasks to these learning agents. In reinforcement learning and imitation learning, which are two common paradigms for teaching new skills to agents, tasks are specified using a reward function or demonstrations respectively. However, these modalities are difficult to work with, as the complexity of tasks grows.

We present techniques that use natural language as an auxiliary signal, in conjunction with rewards or demonstrations, to specify the task more easily. For reinforcement learning, we introduce frameworks that generate rewards from a linguistic description of a task, which can be used in addition to sparse or coarse dense rewards, that are much easier to provide. For imitation learning, we introduce a new setting and propose approaches that allow specifying the desired task using the demonstration of a related task, by using language to describe the difference between the demonstrated task and the desired task, thereby reducing the need for a new demonstration for the desired task.

The approaches presented in this proposal are a step towards building agents that can be communicated with more easily, particularly by non-experts.

Bibliography

[Abolghasemi et al., 2018] Abolghasemi, P., Mazaheri, A., Shah, M., and Bölöni, L. (2018). Pay attention! - Robustifying a Deep Visuomotor Policy through Task-Focused Attention. *arXiv:1809.10093 [cs]*. Reporter: arXiv:1809.10093 [cs] arXiv: 1809.10093.

[Achiam and Sastry, 2017] Achiam, J. and Sastry, S. (2017). Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*.

[Amodei and Clark, 2016] Amodei, D. and Clark, J. (2016). Faulty reward functions in the wild.

[Anderson et al., 2018a] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018a). Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086. Reporter: Proceedings of the IEEE conference on computer vision and pattern recognition.

[Anderson et al., 2018b] Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018b). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

[Andreas et al., 2017] Andreas, J., Klein, D., and Levine, S. (2017). Learning with latent language. *arXiv preprint arXiv:1711.00482*. Reporter: arXiv preprint arXiv:1711.00482.

[Antol et al., 2015] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

[Argall et al., 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

[Arkin et al., 2017] Arkin, J., Walter, M. R., Boteanu, A., Napoli, M. E., Biggie, H., Kress-Gazit, H., and Howard, T. M. (2017). Contextual awareness: Understanding monologic natural language instructions for autonomous robots. In *Robot and Human Interactive Communication (RO-MAN), 2017 26th IEEE International Symposium on*, pages 502–509. IEEE.

[Artzi and Zettlemoyer, 2013] Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62. Reporter: Transactions of the Association for Computational Linguistics Publisher: MIT Press.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Barto and Mahadevan, 2003] Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77.

[Bisk et al., 2016] Bisk, Y., Yuret, D., and Marcu, D. (2016). Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761. Reporter: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

[Blukis et al., 2018a] Blukis, V., Brukhim, N., Bennett, A., Knepper, R. A., and Artzi, Y. (2018a). Following high-level navigation instructions on a simulated quadcopter with imitation learning. *arXiv preprint arXiv:1806.00047*. Reporter: arXiv preprint arXiv:1806.00047.

[Blukis et al., 2018b] Blukis, V., Misra, D., Knepper, R. A., and Artzi, Y. (2018b). Mapping Navigation Instructions to Continuous Control Actions with Position-Visitation Prediction. *arXiv:1811.04179 [cs]*. Reporter: arXiv:1811.04179 [cs] arXiv: 1811.04179.

[Blukis et al., 2019] Blukis, V., Terme, Y., Niklasson, E., Knepper, R. A., and Artzi, Y. (2019). Learning to Map Natural Language Instructions to Physical Quadcopter Control using Simulated Flight. *arXiv preprint arXiv:1910.09664*. Reporter: arXiv preprint arXiv:1910.09664.

[Branavan et al., 2012a] Branavan, S., Kushman, N., Lei, T., and Barzilay, R. (2012a). Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 126–135. Association for Computational Linguistics. Reporter: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.

[Branavan et al., 2012b] Branavan, S., Silver, D., and Barzilay, R. (2012b). Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704. Reporter: Journal of Artificial Intelligence Research.

[Brantley et al., 2019] Brantley, K., Sun, W., and Henaff, M. (2019). Disagreement-regularized imitation learning. In *International Conference on Learning Representations*.

[Broad et al., 2017] Broad, A., Arkin, J., Ratliff, N., Howard, T., and Argall, B. (2017). Real-time natural language corrections for assistive robotic manipulators. *The International Journal of Robotics Research*, 36(5-7):684–698. Number: 5-7 Reporter: The International Journal of Robotics Research Publisher: SAGE Publications Sage UK: London, England.

[Brown et al., 2019] Brown, D., Goo, W., Nagarajan, P., and Niekum, S. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR.

[Brys et al., 2015] Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., and Nowé, A. (2015). Reinforcement learning from demonstration through shaping. In *IJCAI*, pages 3352–3358.

[Burda et al., 2018] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

[Carroll et al., 2019] Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. (2019). On the utility of learning about humans for human-ai coordination. *Advances in Neural Information Processing Systems*, 32:5174–5185.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Co-Reyes et al., 2018] Co-Reyes, J. D., Gupta, A., Sanjeev, S., Altieri, N., Andreas, J., DeNero, J., Abbeel, P., and Levine, S. (2018). Guiding policies with language via meta-learning. *arXiv preprint arXiv:1811.07882*. Reporter: arXiv preprint arXiv:1811.07882.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

[Cui and Niekum, 2018] Cui, Y. and Niekum, S. (2018). Active reward learning from critiques. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6907–6914. IEEE.

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300.

[Duan et al., 2017] Duan, Y., Andrychowicz, M., Stadie, B., Ho, O. J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. (2017). One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098. Reporter: Advances in neural information processing systems.

[Finn et al., 2017] Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. (2017). One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*. Reporter: arXiv preprint arXiv:1709.04905.

[Fried et al., 2018] Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. (2018). Speaker-follower models for vision-and-language navigation. *arXiv preprint arXiv:1806.02724*.

[Garcıa and Fernández, 2015] Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

[Goyal et al., 2021a] Goyal, A., Didolkar, A., Ke, N. R., Blundell, C., Beaudoin, P., Heess, N., Mozer, M., and Bengio, Y. (2021a). Neural production systems. *arXiv preprint arXiv:2103.01937*.

[Goyal et al., 2021b] Goyal, P., Mooney, R. J., and Niekum, S. (2021b). Zero-shot task adaptation using natural language. *arXiv preprint arXiv:2106.02972*.

[Goyal et al., 2019a] Goyal, P., Niekum, S., and Mooney, R. J. (2019a). Using natural language for reward shaping in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China.

[Goyal et al., 2019b] Goyal, P., Niekum, S., and Mooney, R. J. (2019b). Using natural language for reward shaping in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China.

[Goyal et al., 2020] Goyal, P., Niekum, S., and Mooney, R. J. (2020). Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. In *Conference on Robot Learning*, Macao, China.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

[Hemachandra et al., 2015] Hemachandra, S., Duvallet, F., Howard, T. M., Roy, N., Stentz, A., and Walter, M. R. (2015). Learning models for following natural language directions in unknown environments. *arXiv preprint arXiv:1503.05079*.

[Ho and Ermon, 2016] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573. Reporter: Advances in neural information processing systems.

[Howard et al., 2014a] Howard, T. M., Chung, I., Propp, O., Walter, M. R., and Roy, N. (2014a). Efficient natural language interfaces for assistive robots. In *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS) Work. on Rehabilitation and Assistive Robotics*. Citeseer.

[Howard et al., 2014b] Howard, T. M., Tellex, S., and Roy, N. (2014b). A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. IEEE.

[Hutsebaut-Buysse et al., 2020a] Hutsebaut-Buysse, M., Mets, K., and Latre, S. (2020a). Language Grounded Task-Adaptation in Reinforcement Learning. *Computational Intelligence*, page 6. Reporter: Computational Intelligence.

[Hutsebaut-Buysse et al., 2020b] Hutsebaut-Buysse, M., Mets, K., and Latré, S. (2020b). Pre-trained Word Embeddings for Goal-conditional Transfer Learning in Reinforcement Learning. *arXiv:2007.05196 [cs, stat]*. Reporter: arXiv:2007.05196 [cs, stat] arXiv: 2007.05196.

[Kamlish et al., 2019] Kamlish, I., Chocron, I. B., and McCarthy, N. (2019). SentiMATE: Learning to play Chess through Natural Language Processing. *arXiv preprint arXiv:1907.08321*. Reporter: arXiv preprint arXiv:1907.08321.

[Kaplan et al., 2017] Kaplan, R., Sauer, C., and Sosa, A. (2017). Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539*. Reporter: arXiv preprint arXiv:1704.05539.

[Kazemzadeh et al., 2014] Kazemzadeh, S., Ordonez, V., Matten, M., and Berg, T. (2014). Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kroemer et al., 2019] Kroemer, O., Niekum, S., and Konidaris, G. (2019). A review of robot learning for manipulation: Challenges, representations, and algorithms. *arXiv preprint arXiv:1907.03146*.

[Kuhlmann et al., 2004] Kuhlmann, G., Stone, P., Mooney, R., and Shavlik, J. (2004). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*. San Jose, CA. Reporter: The AAAI-2004 workshop on supervisory control of learning and adaptive systems.

[Kurin et al., 2017] Kurin, V., Nowozin, S., Hofmann, K., Beyer, L., and Leibe, B. (2017). The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*.

[Lake et al., 2015] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[MacGlashan et al., 2015] MacGlashan, J., Babes-Vroman, M., desJardins, M., Littman, M. L., Muresan, S., Squire, S., Tellex, S., Arumugam, D., and Yang, L. (2015). Grounding English Commands to Reward Functions. In *Robotics: Science and Systems*. Reporter: Robotics: Science and Systems.

[Mehta and Goldwasser, 2019] Mehta, N. and Goldwasser, D. (2019). Improving Natural Language Interaction with Robots Using Advice. *arXiv preprint arXiv:1905.04655*. Reporter: arXiv preprint arXiv:1905.04655.

[Misra et al., 2018] Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. (2018). Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*. Reporter: arXiv preprint arXiv:1809.00786.

[Misra et al., 2016] Misra, D. K., Sung, J., Lee, K., and Saxena, A. (2016). Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3):281–300.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and others (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. Number: 7540 Reporter: Nature Publisher: Nature Publishing Group.

[Narasimhan et al., 2017] Narasimhan, K., Barzilay, R., and Jaakkola, T. (2017). Deep transfer in reinforcement learning by language grounding. *arXiv preprint arXiv:1708.00133*. Reporter: arXiv preprint arXiv:1708.00133.

[Narasimhan et al., 2018] Narasimhan, K., Barzilay, R., and Jaakkola, T. (2018). Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874. Reporter: Journal of Artificial Intelligence Research.

[Ng et al., 1999] Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287.

[Ng et al., 2000] Ng, A. Y., Russell, S. J., and others (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2. Reporter: Icml.

[Niekum et al., 2013] Niekum, S., Chitta, S., Barto, A. G., Marthi, B., and Osentoski, S. (2013). Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, pages 10–15607. Berlin, Germany.

[Niekum et al., 2012] Niekum, S., Osentoski, S., Konidaris, G., and Barto, A. G. (2012). Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246. IEEE.

[Nyga et al., 2018] Nyga, D., Roy, S., Paul, R., Park, D., Pomarlan, M., Beetz, M., and Roy, N. (2018). Grounding robot plans from natural language instructions with incomplete world knowledge. In *Conference on Robot Learning*, pages 714–723. Reporter: Conference on Robot Learning.

[Pathak et al., 2018] Pathak, D., Mahmoudieh, P., Luo, G., Agrawal, P., Chen, D., Shentu, Y., Shelhamer, E., Malik, J., Efros, A. A., and Darrell, T. (2018). Zero-Shot Visual Imitation. *ICLR 2018*. Reporter: ICLR 2018 arXiv: 1804.08606.

[Paxton et al., 2019] Paxton, C., Bisk, Y., Thomason, J., Byravan, A., and Foxl, D. (2019). Prospection: Interpretable plans from language by predicting the future. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6942–6948. IEEE. Reporter: 2019 International Conference on Robotics and Automation (ICRA).

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Plappert et al., 2018] Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

[Pomerleau, 1991] Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

[Ramachandran and Amir, 2007] Ramachandran, D. and Amir, E. (2007). Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7, pages 2586–2591. Reporter: IJCAI.

[Ramanishka et al., 2017] Ramanishka, V., Das, A., Zhang, J., and Saenko, K. (2017). Top-down visual saliency guided by captions. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

[Ross et al., 2011] Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *arXiv:1011.0686 [cs, stat]*. Reporter: arXiv:1011.0686 [cs, stat] arXiv: 1011.0686.

[Saran et al., 2020] Saran, A., Zhang, R., Short, E. S., and Niekum, S. (2020). Efficiently guiding imitation learning algorithms with human gaze. *arXiv preprint arXiv:2002.12500*.

[Schmidhuber, 1991] Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

[Schulman et al., 2015a] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. Reporter: International conference on machine learning.

[Schulman et al., 2015b] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Shao et al., 2020] Shao, L., Migimatsu, T., Zhang, Q., Yang, K., and Bohg, J. (2020). Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demonstrations. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation.

Meeting Name: Robotics: Science and Systems 2020 Reporter: Robotics: Science and Systems XVI.

[Shridhar et al., 2020] Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

[Stepputtis et al., 2020a] Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., and Amor, H. B. (2020a). Language-Conditioned Imitation Learning for Robot Manipulation Tasks. *arXiv:2010.12083 [cs]*. Reporter: arXiv:2010.12083 [cs] arXiv: 2010.12083.

[Stepputtis et al., 2020b] Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., and Amor, H. B. (2020b). Language-conditioned imitation learning for robot manipulation tasks. *arXiv preprint arXiv:2010.12083*.

[Sumers et al., 2020] Sumers, T. R., Ho, M. K., Hawkins, R. D., Narasimhan, K., and Griffiths, T. L. (2020). Learning Rewards from Linguistic Feedback. *arXiv:2009.14715 [cs]*. Reporter: arXiv:2009.14715 [cs] arXiv: 2009.14715.

[Sung et al., 2018a] Sung, J., Jin, S. H., and Saxena, A. (2018a). Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer.

[Sung et al., 2018b] Sung, J., Jin, S. H., and Saxena, A. (2018b). Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer. Reporter: Robotics Research.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Tambwekar et al., 2021] Tambwekar, P., Silva, A., Gopalan, N., and Gombolay, M. (2021). Interpretable Policy Specification and Synthesis through Natural Language and RL. *arXiv:2101.07140 [cs]*. Reporter: arXiv:2101.07140 [cs] arXiv: 2101.07140.

[Taylor and Stone, 2009] Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

[Tellex et al., 2011] Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S. J., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2.

[Thrun and Pratt, 2012] Thrun, S. and Pratt, L. (2012). *Learning to learn*. Springer Science & Business Media.

[Torabi et al., 2018] Torabi, F., Warnell, G., and Stone, P. (2018). Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Večerík et al., 2017] Večerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.

[Venugopalan et al., 2015a] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015a). Sequence to Sequence – Video to Text. In *Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, December 11-18, 2015*, ICCV '15, pages 4534–4542, Washington, DC, USA. IEEE Computer Society.

[Venugopalan et al., 2015b] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015b). Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542.

[Vezhnevets et al., 2017] Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR.

[Wang and Narasimhan, 2021] Wang, H. J. A. and Narasimhan, K. (2021). Grounding Language to Entities and Dynamics for Generalization in Reinforcement Learning. *arXiv:2101.07393 [cs]*. Reporter: arXiv:2101.07393 [cs] arXiv: 2101.07393.

[Wang et al., 2019] Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.-F., Wang, W. Y., and Zhang, L. (2019). Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

[Yang et al., 2016] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

[You et al., 2016] You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

[Yu et al., 2019] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2019). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*.

[Zhang et al., 2016] Zhang, Y., Marshall, I., and Wallace, B. C. (2016). Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 795. NIH Public Access.

[Ziebart et al., 2008a] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008a). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA. Reporter: Aaai.

[Ziebart et al., 2008b] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008b). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA.