

# Grounding Language for Transfer in Deep Reinforcement Learning

**Karthik Narasimhan**

*Department of Computer Science  
Princeton University  
35 Olden Street, Princeton, NJ 08540 USA*

KARTHIKN@CS.PRINCETON.EDU

**Regina Barzilay**

*Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology  
32 Vassar Street, Cambridge, MA 02139 USA*

REGINA@CSAIL.MIT.EDU

**Tommi Jaakkola**

*Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology  
32 Vassar Street, Cambridge, MA 02139 USA*

TOMMI@CSAIL.MIT.EDU

## Abstract

In this paper, we explore the utilization of natural language to drive transfer for reinforcement learning (RL). Despite the wide-spread application of deep RL techniques, learning generalized policy representations that work across domains remains a challenging problem. We demonstrate that textual descriptions of environments provide a compact intermediate channel to facilitate effective policy transfer. Specifically, by learning to ground the meaning of text to the dynamics of the environment such as transitions and rewards, an autonomous agent can effectively bootstrap policy learning on a new domain given its description. We employ a model-based RL approach consisting of a differentiable planning module, a model-free component and a factorized state representation to effectively use entity descriptions. Our model outperforms prior work on both transfer and multi-task scenarios in a variety of different environments. For instance, we achieve up to 14% and 11.5% absolute improvement over previously existing models in terms of average and initial rewards, respectively.

## 1. Introduction

Deep reinforcement learning has emerged as a method of choice for many control applications, ranging from computer games (Mnih et al., 2015; Silver et al., 2016) to robotics (Levine, Finn, Darrell, & Abbeel, 2016). However, the success of this approach depends on a substantial number of interactions with the environment during training, easily reaching millions of steps (Nair et al., 2015; Mnih et al., 2016). Moreover, given a new task, even a related one, this training process has to be performed from scratch. This inefficiency has motivated recent work in learning universal policies that can generalize across related tasks (Schaul, Horgan, Gregor, & Silver, 2015), as well as other transfer approaches (Parisotto, Ba, & Salakhutdinov, 2016; Rajendran, Lakshminarayanan, Khapra, Prasanna, & Ravindran, 2017). In this paper, we explore transfer methods that use text descriptions to facilitate policy generalization across tasks.



Figure 1: Examples of two different game environments, Boulderchase (**top**) and Bomberman (**bottom**). Each domain has text descriptions (collected using Amazon Mechanical Turk) associated with specific entities, describing characteristics such as movement and interactions with the player’s avatar. Note how certain pairs of entities across games share certain properties. For instance, the scorpion in Boulderchase and the spider in Bomberman are both mobile entities.

As an example, consider the game environments in Figure 1. The two games – *Boulderchase* and *Bomberman* – differ in their layouts and entity types. However, the high-level behavior of most entities in both games is similar. For instance, the *scorpion* in Boulderchase (top) is a moving entity which the agent has to avoid, similar to the *spider* in Bomberman (bottom). Though this similarity is clearly reflected in the text descriptions in Figure 1, it may take multiple environment interactions to discover. Therefore, exploiting these textual clues could help an autonomous agent understand this connection more effectively, leading to faster policy learning.

To test this hypothesis, we consider multiple environments augmented with textual descriptions. These descriptions provide a short overview of objects and their modes of interaction in the environment.<sup>1</sup> They do not describe control strategies, which were commonly used in prior work on grounding (Vogel & Jurafsky, 2010; Branavan, Silver, & Barzilay, 2012). Instead, they specify the dynamics of the environments, which are more conducive to cross-domain transfer.

In order to effectively use this type of information, we employ a model-based reinforcement learning approach. Typically, representations of the environment learned by these approaches are inherently domain-specific. We address this issue by using natural language as an implicit intermediate channel for transfer. Specifically, our model learns to map text descriptions to transitions and rewards in an environment, a capability that speeds up learning in unseen domains. We induce a two-part representation for the input state that generalizes over domains, incorporating both domain-specific information and textual knowledge. This representation is utilized by an action-

1. We do not require that every object to have an associated description.

value function, parametrized as a single deep neural network with a differentiable value iteration module (Tamar, Wu, Thomas, Levine, & Abbeel, 2016). The entire model is trained end-to-end using rewards from the environment.

We evaluate our model on several game worlds from the GVGAI framework (Perez-Liebana, Samothrakis, Togelius, Schaul, & Lucas, 2016). In our main evaluation scenario of transfer learning, an agent is trained on a set of source tasks and its learning performance is evaluated on a different set of target tasks. Across multiple evaluation metrics, our method consistently outperforms several baselines and an existing transfer approach for deep reinforcement learning called Actor Mimic (Parisotto et al., 2016). For instance, our model achieves up to 14% higher average reward and up to 11.5% higher initial reward - two key metrics used to evaluate transfer learning (Taylor & Stone, 2009). We also demonstrate our model’s improved performance on a multi-task setting where learning is simultaneously performed on multiple environments.

The rest of this paper is organized as follows. Section 2 summarizes related work on grounding and transfer for reinforcement learning; Section 3 provides an overview of the framework we use; Section 4 describes our model architecture and its various components; Section 5 details the experimental setup, and Section 6 contains our empirical results and analysis. We conclude and discuss some future directions for research in Section 7. Code for the experiments in this paper is available at <https://github.com/karthikncode/Grounded-RL-Transfer>.

## 2. Related Work

We now provide a brief overview of related work in the areas of language grounding and transfer for reinforcement learning.

### 2.1 Grounding Language in Interactive Environments

In recent years, there has been increasing interest in systems that can utilize textual knowledge to learn control policies. Such applications include interpreting help documentation (Branavan, Zettlemoyer, & Barzilay, 2010), instruction following (Vogel & Jurafsky, 2010; Kollar, Tellex, Roy, & Roy, 2010; Artzi & Zettlemoyer, 2013; Matuszek, Herbst, Zettlemoyer, & Fox, 2013; Andreas & Klein, 2015) and learning to play computer games (Branavan, Silver, & Barzilay, 2011, 2012; Narasimhan, Kulkarni, & Barzilay, 2015; He et al., 2016). In all these applications, the models are trained and tested on the same domain.

Our work represents two departures from prior work on grounding. First, rather than optimizing control performance for a single domain, we are interested in the multi-domain transfer scenario, where language descriptions drive generalization. Second, prior work used text in the form of strategy advice to directly learn the policy. Since the policies are typically optimized for a specific task, they may be harder to transfer across domains. Instead, we utilize text to bootstrap the induction of the environment dynamics, moving beyond task-specific strategies.

Another related line of work consists of systems that learn spatial and topographical maps of the environment for robot navigation using natural language descriptions (Walter, Hemachandra, Homberg, Tellex, & Teller, 2013; Hemachandra, Walter, Tellex, & Teller, 2014). These approaches use text mainly containing appearance and positional information, and integrate it with other semantic sources (such as appearance models) to obtain more accurate maps. In contrast, our work uses language describing the dynamics of the environment, such as entity movements and interactions, which is complementary to static positional information received through state observations.

Further, our goal is to help an agent learn policies that generalize over different stochastic domains, while their works consider a single domain.

## 2.2 Transfer in Reinforcement Learning

Transferring policies across domains is a challenging problem in reinforcement learning (Konidaris, 2006; Taylor & Stone, 2009). The main hurdle lies in learning a good mapping between the state and action spaces of different domains to enable effective transfer. Most previous approaches have either explored skill transfer (Konidaris & Barto, 2007; Konidaris, Scheidwasser, & Barto, 2012) or value function/policy transfer (Liu & Stone, 2006; Taylor, Stone, & Liu, 2007; Taylor & Stone, 2007). There have also been attempts at model-based transfer for RL (Taylor, Jong, & Stone, 2008; Nguyen, Silander, & Leong, 2012; Gašić et al., 2013; Wang, Wen, Su, & Stylianou, 2015; Joshi & Chowdhary, 2018) but these methods either rely on hand-coded inter-task mappings for state and actions spaces or require significant interactions in the target task to learn an effective mapping. Our approach doesn't use any explicit mappings and can learn to predict the dynamics of a target task using its descriptions.

A closely related line of work concerns transfer methods for deep reinforcement learning. Parisotto et al. (2016) train a deep network to mimic pre-trained experts on source tasks using policy distillation. The learned parameters are then used to initialize a network on a target task to perform transfer. Rusu et al. (2016) facilitate transfer by freezing parameters learned on source tasks and adding a new set of parameters for every new target task, while using both sets to learn the new policy. Work by Rajendran et al. (2017) uses attention networks to selectively transfer from a set of expert policies to a new task. Barreto et al. (2017) use features based on successor representations (Dayan, 1993) for transfer across tasks in the same domain. Kansky et al. (2017) learn a generative model of causal physics in order to help zero-shot transfer learning. Our approach is orthogonal to all these directions since we use text to bootstrap transfer, and can potentially be combined with these methods to achieve more effective transfer.

There has also been prior work on zero-shot policy generalization for tasks with input goal specifications. Schaul, Horgan, Gregor, and Silver (2015) learn a universal value function approximator that can generalize across both states and goals. (Andreas, Klein, & Levine, 2017) use policy sketches, which are annotated sequences of subgoals, in order to learn a hierarchical policy that can generalize to new goals. Oh, Singh, Lee, and Kohli (2017) investigate zero-shot transfer for instruction following tasks, aiming to generalize to unseen instructions in the same domain. The main departure of our work compared to these is in the use of environment descriptions for generalization across domains rather than generalizing across text instructions.

Perhaps closest in spirit to our hypothesis is the recent work by (Harrison, Ehsan, & Riedl, 2017). Their approach makes use of paired instances of text descriptions and state-action information from human gameplay to learn a machine translation model. This model is incorporated into a policy shaping algorithm to better guide agent exploration. Although the motivation of language-guided control policies is similar to ours, their work considers transfer across tasks in a single domain, and requires human demonstrations to learn a policy.

### 2.3 Using Task Features for Transfer

The idea of using task features/dictionaries for zero-shot generalization has been explored previously in the context of image classification. Kodirov, Xiang, Fu, and Gong (2015) learn a joint feature embedding space between domains and also induce the corresponding projections onto this space from different class labels. Kolouri, Rostami, Owechko, and Kim (2018) learn a joint dictionary across visual features and class attributes using sparse coding techniques. Romera-Paredes and Torr (2015) model the relationship between input features, task attributes and classes as a linear model to achieve efficient yet simple zero-shot transfer for classification. Socher, Ganjoo, Manning, and Ng (2013) learn a joint semantic representation space for images and associated words to perform zero-shot transfer.

Task descriptors have also been explored in zero-shot generalization for control policies. Sinapov, Narvekar, Leonetti, and Stone (2015) use task meta-data as features to learn a mapping between pairs of tasks. This mapping is then used to select the most relevant source task to transfer a policy from. Isele, Rostami, and Eaton (2016) build on the ELLA framework (Ruvolo & Eaton, 2013; Ammar, Eaton, Ruvolo, & Taylor, 2014), and their key idea is to maintain two shared linear bases across tasks – one for the policy ( $L$ ) and the other for task descriptors ( $D$ ). Once these bases are learned on a set of source tasks, they can be used to predict policy parameters for a new task given its corresponding descriptor. In these lines of work, the task features were either manually engineered or directly taken from the underlying system parameters defining the dynamics of the environment. In contrast, our framework only requires access to crowd-sourced textual descriptions, alleviating the need for expert domain knowledge.

## 3. General Framework

Our goal in this work is to demonstrate the utility of natural language descriptions in assisting policy transfer across domains. In this section, we first describe our environment setup and the general framework of our approach. The details of our model and algorithm follow in Section 4.

### 3.1 Environment Setup

We model a single environment as a Markov Decision Process (MDP),  $E = \langle S, A, T, R, O, Z \rangle$ . Here,  $S$  is the state space, and  $A$  is the set of actions available to the agent. In this work, we consider every state  $s \in S$  to be a 2-dimensional grid of size  $m \times n$ , with each cell containing an entity symbol  $o \in O$ .<sup>2</sup>  $T$  is the transition distribution over all possible next states  $s'$  conditioned on the agent choosing action  $a$  in state  $s$ .  $R$  determines the reward provided to the agent at each time step. The agent does not have access to the true  $T$  and  $R$  of the environment. Each domain also has a goal state  $s_g \in S$  which determines when an episode terminates. Finally,  $Z$  is the complete set of text descriptions provided to the agent for this particular environment.

### 3.2 Reinforcement Learning (RL)

The goal of an autonomous agent is to maximize cumulative reward obtained from the environment. A traditional way to achieve this is by learning an action value function  $Q(s, a)$  through

---

2. In our experiments, we relax this assumption to allow for multiple entities per cell, but for ease of description, we shall assume a single entity per cell. The assumption of 2-D worlds can also be easily relaxed to generalize our model to other situations.

reinforcement. The  $Q$ -function predicts the expected future reward for choosing action  $a$  in state  $s$ . A straightforward policy then is to simply choose the action that maximizes the  $Q$ -value in the current state:

$$\pi(s) = \arg \max_a Q(s, a)$$

If we also make use of the descriptions, we have a text-conditioned policy:

$$\pi(s, Z) = \arg \max_a Q(s, a, Z) \quad (1)$$

A successful control policy for an environment will contain both knowledge of the environment dynamics and the capability to identify goal states. While the latter is task-specific, the former characteristic is more useful for learning a general policy that transfers to different domains. Based on this hypothesis, we employ a model-aware RL approach that can learn the dynamics of the world while estimating the optimal  $Q$ . Specifically, we make use of *Value Iteration (VI)* (Sutton & Barto, 1998), an algorithm based on dynamic programming. The update equations for value iteration in our setup are:

$$\begin{aligned} Q^{(n+1)}(s, a, Z) &= \sum_{s' \in S} T(s'|s, a, Z)[R(s', Z) + \gamma V^{(n)}(s', Z)] \\ V^{(n+1)}(s, Z) &= \max_a Q^{(n+1)}(s, a, Z) \end{aligned} \quad (2)$$

where  $\gamma$  is a discount factor and  $n$  is the iteration number. The updates require an estimate of  $T$  and  $R$ , which the agent must obtain through exploration of the environment.

### 3.3 Text Descriptions

Estimating the dynamics of the environment from interactive experience can require a significant number of samples. Our main hypothesis is that if an agent can derive information about the dynamics from text descriptions, it can determine  $T$  and  $R$  faster and more accurately.

For instance, consider the sentence “*Red bat that moves horizontally, left to right*”. This talks about the movement of a third-party entity (*bat*), independent of the agent’s goal. Provided the agent can learn to interpret this sentence, it can then infer the direction of movement of a different entity (e.g. “*A tan car moving slowly to the left*”) in a different domain. Further, this inference is useful even if the agent has a completely different goal. On the other hand, instruction-like text such as “*Move towards the wooden door*” is highly context-specific and only relevant to domains that have the mentioned goal.

With this in mind, we provide the agent with text descriptions that collectively portray characteristics of the world. These descriptions are crowdsourced by asking humans to view gameplay videos and describe entities. A single description talks about one particular entity in the world. The text contains (partial) information about the entity’s movement and interaction with the player avatar. Each description is also aligned to its corresponding entity in the environment and not all entities may have a description. Figure 2 provides some samples; more details on data collection and statistics are in Section 5.

- Scorpion2: *Red scorpion that moves up and down*
- Alien3: *This character slowly moves from right to left while having the ability to shoot upwards*
- Sword1: *This item is picked up and used by the player for attacking enemies*

Figure 2: Example text descriptions of entities in different environments, collected using Amazon Mechanical Turk. Turkers were shown videos of gameplay in the different environments and asked to describe each entity’s behavior or role. Note that these sentences are not instructive, since they provide no direct information on how to act in the environment.

### 3.4 Transfer for RL

In order to test our grounding hypothesis, we consider learning across multiple environments. Specifically, an agent can learn to ground language semantics in an environment  $E_1$  and then we can test its understanding capability by placing it in a new unseen domain,  $E_2$ . The agent can obtain unlimited experience in  $E_1$ , and after convergence of its policy, it is allowed to interact with and learn a policy for  $E_2$ . We do not provide the agent with any explicit mapping between different entities or goals across domains, either directly or through the text. For instance, even though the boulders in *Boulderchase* are impassable objects just like the walls in *Bomberman 1*, the agent does not have access to a mapping between these entities. In this setup, the agent’s goal is to re-utilize information obtained through its interactions in  $E_1$  to learn more efficiently in  $E_2$ .

## 4. Model

Grounding language for policy transfer across domains requires a model that meets two needs. First, it must allow for a flexible representation that fuses information from both state observations and text descriptions. This representation should capture the compositional nature of language while mapping linguistic semantics to characteristics of the world. Second, the model must have the capability to learn an accurate prototype of the environment (i.e. transitions and rewards) using only interactive feedback. Overall, the model must enable an agent to map text descriptions to environment dynamics; this allows it to predict transitions and rewards in a completely new world, without requiring substantial interaction.

To this end, we propose a neural architecture consisting of two main components: (1) a *representation generator* ( $\phi$ ), and (2) a *value iteration network* (VIN) (Tamar et al., 2016). First, the representation generator takes a state observation and a set of text descriptions as input and produces a tensor output, capturing information essential for decision-making. Then, the VIN module implicitly encodes the value iteration computation (Eq. 2) into a recurrent network with convolutional modules, producing an action-value function using the previously constructed tensor representation as input. Together, both modules form an end-to-end differentiable network that can be trained using gradient back-propagation.

### 4.1 Representation Generator

The main purpose of this module (Figure 3) is to fuse together information from two inputs – the state, and the text specifications. An important consideration, however, is the ability to handle

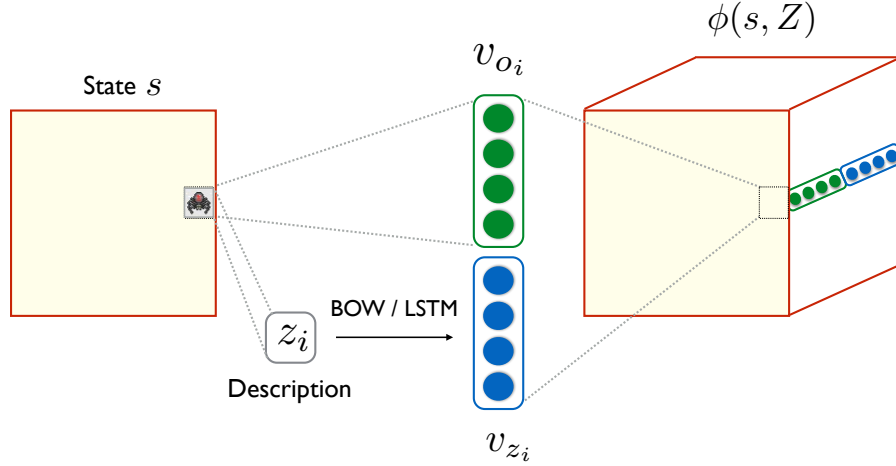


Figure 3: Representation generator combining both object-specific and description-informed vectors for each entity. Each cell in the input state (2-D matrix) is converted to a corresponding real-valued vector, resulting in a 3-D tensor output. The two-part representation allows us to exploit partial/noisy information from text while also learning other aspects of the environment dynamics directly through interaction.

partial or incomplete text descriptions, which may not contain all the particulars of an entity. Thus, we would like to incorporate useful information from the text, yet, not rely on it completely. This motivates us to use a representation that is factorized over the two input modalities.

Formally, given a state matrix  $s$  of size  $m \times n$  and a set of text descriptions  $Z$ , the module produces a tensor  $\phi(s, Z)$ . Recall that each cell in state  $s$  is occupied by a single entity. Consider one such cell containing an entity  $o_i$ , with a corresponding description  $z_i$  (if available). The representation generator performs two operations:

1. First, it generates an entity-specific vector  $v_{o_i}$  of dimension  $d$ . This vector is initialized arbitrarily and learned using rewards received by the agent in the environment. One can view this operation as an ‘object embedding’, similar to the notion of a word embedding (Mikolov, Chen, Corrado, & Dean, 2013).
2. Second, the description  $z_i$  is converted into a continuous valued vector  $v_{z_i}$  (also of dimension  $d$ ). This can be achieved in several different ways, but in this work, we experiment with using an LSTM recurrent neural network (Hochreiter & Schmidhuber, 1997) and a mean bag-of-words (BOW) approach, which entails taking the average over word vectors corresponding to each word in the description.

Both these sets of parameters (including the embeddings and LSTM weights) are all initialized at random and learned through reinforcement on the source tasks.

The two vectors,  $v_{o_i}$  and  $v_{z_i}$ , are then concatenated to produce a single representation for this cell:  $\phi_i = [v_{o_i}; v_{z_i}]$ . Performing the same operations over all cells of the state results in a tensor  $\phi(s, Z)$  with dimensions  $m \times n \times 2d$  for the entire state.<sup>3</sup> For cells with no entity (i.e. empty space),

3.  $d$  is a hyperparameter choice here. Also, one can have vectors  $v_{o_i}$  and  $v_{z_i}$  of different dimensions, say  $d_1$  and  $d_2$ , if necessary. We use the same dimensionality for simplicity.



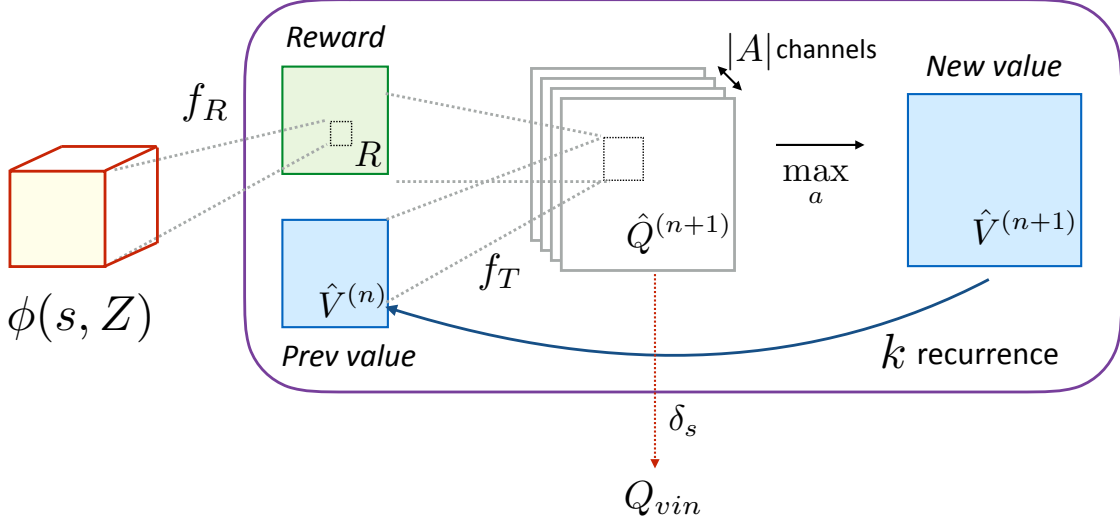


Figure 4: Value iteration network (VIN) module to compute  $Q_{vin}$  from  $\phi(s, Z)$ . The module approximates the value iteration computation using neural networks to predict reward and value maps, arranged in a recurrent fashion. Functions  $f_T$  and  $f_R$  are implemented using convolutional neural networks (CNNs).  $\delta_s$  is a selection function to pick out a single Q-value (at the agent’s current location) from the output Q-value map  $\hat{Q}^{(k)}$ .

$\phi_i$  is simply a zero vector, and for entities without a description,  $v_{z_i} = \vec{0}$ . Figure 3 illustrates this module.

This decomposition into  $v_o$  and  $v_z$  allows us to learn policies based on both the ID of an object and its described behavior in text. This enables the model to retain knowledge of observed entities while being adaptable to new entities. For instance, if a new environment contains some previously seen entities,<sup>4</sup> the agent can reuse the learned representations directly based on their symbols. For completely new entities (with unseen IDs), the model can form useful representations using their text descriptions.

## 4.2 Value Iteration Network

For a model-based RL approach to this task, we require some means to estimate  $T$  and  $R$  of an environment. One way to achieve this is by explicitly using predictive models for both functions and learning these through transitions experienced by the agent. These models can then be used to estimate the optimal  $Q$  using equation 2. However, this pipelined approach would result in errors propagating through the different stages of predictions.

A value iteration network (VIN) (Tamar et al., 2016) abstracts away explicit computation of  $T$  and  $R$  by directly predicting the outcome of value iteration (Figure 4), thereby avoiding the aforementioned error propagation. In this model, the VI computation is mimicked by a recurrent network with two key operations at each step. First, to compute  $Q$ , we have two functions –  $f_R$  and  $f_T$ .  $f_R$  is a reward predictor that operates on  $\phi(s, Z)$  while  $f_T$  uses the output of  $f_R$  and

4. Note that this is just a possible situation our model can handle. The different domains we consider in this work have no entity symbol overlap.

any previous  $V$  to predict  $Q$ . Both functions are parametrized as convolutional neural networks (CNNs),<sup>5</sup> to suit our tensor representation  $\phi$ . Subsequently, in the second operation, the network employs max pooling over the action channels in the  $Q$ -value map produced by  $f_T$  to obtain  $V$ . The value iteration computation (from Eq. 2) can thus be approximated as:

$$\hat{Q}^{(n+1)}(s, a, Z) = f_T\left(f_R(\phi(s, Z), a; \theta_R), \hat{V}^{(n)}(s, Z); \theta_T\right) \quad (3)$$

$$\hat{V}^{(n+1)}(s, Z) = \max_a \hat{Q}^{(n+1)}(s, a, Z) \quad (4)$$

Note that while the VIN operates on  $\phi(s, Z)$ , we write  $\hat{Q}$  and  $\hat{V}$  in terms of the original state input  $s$  and text  $Z$ , since these are independent of our chosen representation.

The outputs of both CNNs are real-valued tensors. The output of  $f_R$  has the same dimensions as the input state ( $m \times n$ ), while  $f_T$  produces  $\hat{Q}^{(n+1)}$  as a tensor of dimension  $m \times n \times |A|$ , where  $|A|$  is the number of actions available to the agent. A key point here is that the model produces  $Q$  and  $V$  values for each cell of the input state matrix, assuming the agent’s position to be that particular cell. The convolution filters help capture information from neighboring cells in our state matrix, which act as approximations for  $V^{(n)}(s', Z)$ . The parameters of the CNNs,  $\theta_R$  and  $\theta_T$ , approximate  $R$  and  $T$ , respectively. See the work of Tamar et al. (2016) for a more detailed discussion.

The recursive computation of traditional value iteration (Eq. 2) is captured by employing the CNNs in a recurrent fashion for  $k$  steps.<sup>6</sup> Intuitively, larger values of  $k$  imply a larger field of neighbors influencing the  $Q$ -value prediction for a particular cell as the information propagates longer. The final output of this recurrent computation,  $\hat{Q}^{(k)}$ , is a 3-D tensor of size  $m \times n \times |A|$ . However, since we need a policy only for the agent’s current location, we use an appropriate selection function  $\delta_s$ , which reduces this  $Q$ -value map to a single set of action values for the agent’s location:

$$Q_{vin}(s, a, Z; \Theta_1) = \delta_s(\hat{Q}^{(k)}(s, a, Z)) \quad (5)$$

This is simply an indexing operation performed on  $\hat{Q}^{(k)}$  to retrieve the  $|A|$ -dimensional vector from the cell corresponding to the agent’s location in state  $s$ .

### 4.3 Final Prediction

Games exhibit complex dynamics, which are challenging to capture precisely, especially for long-term prediction. VINs approximate the dynamics implicitly via learned convolutional operations. It is thus likely that the estimated  $Q_{vin}$  values are most helpful for short-term planning that corresponds to a limited number of iterations  $k$ . Therefore, we need to complement these ‘local’  $Q$ -values with estimates based on a more global view.

To this end, following the VIN specification by Tamar et al. (2016), our architecture also contains a model-free action-value function, implemented as a deep  $Q$ -network (DQN) (Mnih et al., 2015). This network provides a  $Q$ -value prediction –  $Q_r(s, a, Z; \Theta_2)$  – which is combined with  $Q_{vin}$  using a composition function  $g$ :<sup>7</sup>

$$Q(s, a, Z; \Theta) = g(Q_{vin}(s, a, Z; \Theta_1), Q_r(s, a, Z; \Theta_2)) \quad (6)$$

5. Other parameterizations are possible for different input types, as noted by Tamar et al. (2016).

6.  $k$  is a model hyperparameter.

7. Although  $g$  can also be learned, we use component-wise addition in our experiments.

**Algorithm 1** MULTITASK-TRAIN ( $\mathcal{E}$ )

---

```

1: Initialize parameters  $\Theta$  and experience replay  $\mathcal{D}$ 
2: for  $k = 1, M$  do ▷ New episode
3:   Choose next environment  $E_k \in \mathcal{E}$ 
4:   Initialize  $E_k$ ; get start state  $s_1 \in S_k$ 
5:   for  $t = 1, N$  do ▷ New step
6:     Select  $a_t \sim \text{EPS-GREEDY}(s_t, Q_\Theta, Z_k, \epsilon)$ 
7:     Execute action  $a_t$ , observe reward  $r_t$  and new state  $s_{t+1}$ 
8:      $\mathcal{D} = \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1}, Z_k)$ 
9:     Sample mini batch  $(s_j, a_j, r_j, s_{j+1}, Z_k) \sim \mathcal{D}$ 
10:    Perform gradient descent on loss  $\mathcal{L}$  to update  $\Theta$ 
11:    if  $s_{t+1}$  is terminal then break
12: Return  $\Theta$ 

```

---

**Algorithm 2** EPS-GREEDY ( $s, Q, Z, \epsilon$ )

---

```

1: if  $\text{random}() < \epsilon$  then
2:   Return random action  $a$ 
3: else
4:   Return  $\arg \max_a Q(s, a, Z)$ 
5: Return  $\Theta$ 

```

---

The fusion of our model components enables our agent to establish the connection between input text descriptions, represented as vectors, and the environment’s transitions and rewards, encoded as VIN parameters. In a new domain, the model can produce a reasonable policy using corresponding text, even before receiving any interactive feedback.

#### 4.4 Parameter Learning

Our entire model is end-to-end differentiable. We perform updates derived from the Bellman equation (Sutton & Barto, 1998):

$$Q_{i+1}(s, a, Z) = \mathbb{E}[r + \gamma \max_{a'} Q_i(s', a', Z) \mid s, a] \quad (7)$$

where the expectation is over all transitions from state  $s$  with action  $a$  and  $i$  is the update number. To learn our parametrized Q-function (the result of Eq. 6), we can use backpropagation through the network to minimize the following loss:

$$\mathcal{L}(\Theta_i) = \mathbb{E}_{\hat{s}, \hat{a}} [(y_i - Q(\hat{s}, \hat{a}, Z; \Theta_i))^2] \quad (8)$$

where  $y_i = r + \gamma \max_{a'} Q(s', a', Z; \Theta_{i-1})$  is the target Q-value with parameters  $\Theta_{i-1}$  fixed from the previous iteration. We employ an experience replay memory  $\mathcal{D}$  to store transitions (Mnih et al., 2015), and periodically perform updates with random samples from this memory. We use an  $\epsilon$ -greedy policy (Sutton & Barto, 1998) for exploration.

## 4.5 Transfer Procedure

The traditional transfer learning scenario often involves a single task in both source and target environments. To better test generalization and robustness of our methods, in this work we consider transfer from *multiple* source tasks to *multiple* target tasks. We first train a model to achieve optimal performance on the set of source tasks. All model parameters ( $\Theta$ ) are shared between these tasks. The agent experiences one episode at a time, sampled from each environment in a round-robin fashion, along with the corresponding text descriptions. The parameters of the model are optimized using the reward-based feedback gathered across all these tasks. Algorithm 1 details this multi-task training procedure.

After training converges, we use the learned parameters to initialize a model for tasks in the target domain. Specifically, all parameters of the VIN are replicated, while most weights of the representation generator are reused. Previously seen objects and words retain their learned entity-specific embeddings ( $v_o$ ), whereas vectors for new objects/words in the target tasks are initialized randomly. Following this initialization, all parameters are then fine-tuned on the target tasks using the corresponding rewards, again with episodes sampled in a round-robin fashion.

## 5. Experimental Setup

We now detail our empirical setup including environments, text descriptions, evaluation metrics, baselines and model implementation details. Results follow in Section 6.

### 5.1 Environments

We perform experiments on a series of 2-D environments within the GVGAI framework (Perez-Liebana et al., 2016), which is used in an annual video game AI competition.<sup>8</sup> In addition to pre-specified games, the framework supports the creation of new games using the Py-VGDL description language (Schaul, 2013). We use four different games to evaluate transfer and multitask learning: *Freeway*, *Bombberman*, *Boulderchase* and *Friends & Enemies*. There are certain similarities between these games. For one, each game consists of a 16x16 grid with the player controlling a movable avatar with two degrees of freedom. Also, each domain contains other entities, both stationary and moving (e.g. diamonds, spiders), that can interact with the avatar.

However, each game also has its own distinct characteristics. In *Freeway*, the goal is to cross a multi-lane freeway while avoiding cars in the lanes. The cars move at various paces in either horizontal direction. *Bombberman* and *Boulderchase* involve the player seeking an exit door while avoiding enemies that either chase the player, run away or move at random. The agent also has to collect resources like diamonds and dig or place bombs to clear paths. These three games have five level variants each with different map layouts and starting entity placements.

*Friends & Enemies* (F&E) is a new environment we designed, with a larger variation of entity types. This game has a total of twenty different non-player entities, each with different types of movement and interaction with the player’s avatar. For instance, some entities move at random while some chase the avatar or shoot bullets that the avatar must avoid. The objective of the player is to meet all friendly entities while avoiding enemies. For each game instance, four non-player entities are sampled from this pool and randomly placed in the grid. This makes F&E instances significantly more varied than the previous three games. We created two versions of this game: F&E-1 and F&E-

---

8. <http://www.gvgai.net/>

2, with the sprites in F&E-2 moving faster, making it a harder environment. Table 1 contains all the different transfer scenarios we consider in our experiments.

Condition	Source	Target	% vocab overlap
F&E-1 $\rightarrow$ F&E-2	7	3	100
F&E-1 $\rightarrow$ Freeway	7	5	18.06
Bomberman $\rightarrow$ Boulderchase	5	5	19.6

Table 1: Statistics on source and target games for various transfer experiments. First two columns indicate the number of instances of each game, while the last column contains the percentage of overlap between the vocabularies of the corresponding text descriptions, collected using Amazon Mechanical Turk.

## 5.2 Text Descriptions

We collect textual descriptions using Amazon Mechanical Turk (Buhrmester et al., 2011). We provide annotators with sample gameplay videos of each game and ask them to describe specific entities in terms of their movement and interactions with the avatar. Since we ask the users to provide an independent account of each entity, we obtain *descriptive* sentences as opposed to *instructive* ones which inform the optimal course of action from the avatar’s viewpoint.<sup>9</sup>

We aggregated together four sets of descriptions, each from a different annotator, for every environment. This resulted in an average of around 36 unique sentences per domain, with *F&E* having the most: 78 sentences. Apart from lowercasing the text, we do not perform any extra pre-processing. Each description in an environment is aligned to one constituent entity. We also make sure that the entity names are not repeated across games (even for the same entity type). Table 2 provides corpus-level statistics on the collected data and Figure 2 has sample descriptions.

Unique word types	286
Avg. words / sentence	8.65
Avg. sentences / domain	36.25
Max sentence length	22

Table 2: Overall statistics of the text descriptions collected using Mechanical Turk.

## 5.3 Evaluation Metrics

We evaluate transfer performance using three metrics defined and employed in previous work (Taylor & Stone, 2009):

- *Average Reward*, which is the area under the reward curve divided by the number of test episodes.
- *Initial performance*, which is the average reward over first 50k steps.

9. Upon manual verification, we find less than 3% of the obtained annotations to be instructive, i.e. containing text that explicitly instruct the agent on steps to take in order to achieve the goal.

- *Asymptotic performance*, which is the average reward over 50k steps after convergence.

The first two metrics emphasize the speed at which a transfer method can enable learning on the target domain, while the last one evaluates its ability to achieve optimal performance on the task. An ideal method should provide gains on all three metrics. For the multitask scenario, we consider the average and asymptotic reward only. For each metric, we repeat experiments with nine different random seeds and report mean and standard deviation numbers.

## 5.4 Baselines

We explore several baseline models for empirical comparison. The different conditions we consider are:

- NO TRANSFER: A deep Q-network (DQN) (Mnih et al., 2015) is initialized randomly and trained from scratch on target tasks. This is the only case that does not use parameters transferred from source tasks.
- DQN: A DQN is trained on source tasks and its parameters are transferred to target tasks. This model does not make use of text descriptions.
- TEXT-DQN: This is a DQN with our hybrid representation  $\phi(s, Z)$ , using the text descriptions. This is essentially a reactive-only version of our model, i.e. without the VIN planning module.
- AMN: The Actor-Mimic network is a recently proposed transfer method (Parisotto et al., 2016) for deep RL. AMN employs policy distillation to train a single network using expert policies previously learned on multiple different tasks. This network is then used to initialize a model for a new task.<sup>10</sup>
- VIN: A value iteration network is trained on the source tasks without making use of the text descriptions. This is effectively an ablation of our full model that only receives state observations as input.

## 5.5 Implementation Details

We now provide details on our model implementations. For all models, we set  $\gamma = 0.8$ ,  $|\mathcal{D}| = 250k$ , and the embedding size  $d = 10$ . We used the *Adam* (Kingma & Ba, 2014) optimization scheme with a learning rate of  $10^{-4}$ , annealed linearly to  $5 \times 10^{-5}$ . The minibatch size was set to 32.  $\epsilon$  was annealed from 1 to 0.1 in the source tasks and set to 0.1 in the target tasks. For the value iteration module (VIN), we experimented with different levels of recurrence,  $k \in \{1, 2, 3, 5\}$  and found  $k = 1$  or  $k = 3$  to work best.<sup>11</sup> For DQN, we used two convolutional layers followed by a single fully connected layer, with ReLU non-linearities. The CNNs in the VIN had filters and strides of length 3. The CNNs in the model-free component used filters of sizes  $\{4, 2\}$  and corresponding strides of size  $\{3, 2\}$ . All embeddings are initialized at random.<sup>12</sup>

10. We only evaluate AMN on transfer since it does not perform online multitask learning and is not directly comparable.

11. We still observed transfer gains with all  $k$  values.

12. We also experimented with using pre-trained word embeddings for text but obtained equal or worse performance.

<b>F&amp;E-1 <math>\rightarrow</math> F&amp;E-2</b>			
<i>Model</i>	<i>Average</i>	<i>Initial</i>	<i>Asymptotic</i>
NO TRANSFER	0.88 (0.10)	-0.24 (0.09)	1.46 (0.07)
DQN	0.98 (0.14)	0.36 (0.10)	1.20 (0.09)
TEXT-DQN	0.93 (0.13)	0.47 (0.33)	1.21 (0.10)
AMN (Actor Mimic)	1.22 (0.05)	0.13 (0.10)	<b>1.64</b> (0.01)
VIN (3)	1.14 (0.08)	0.12 (0.16)	1.49 (0.07)
TEXT-VIN (3)	<b>1.32</b> (0.06)	<b>0.70</b> (0.22)	1.50 (0.05)

<b>F&amp;E-1 <math>\rightarrow</math> Freeway</b>			
<i>Model</i>	<i>Average</i>	<i>Initial</i>	<i>Asymptotic</i>
NO TRANSFER	0.22 (0.03)	-0.95 (0.10)	0.82 (0.03)
DQN	0.21 (0.16)	-0.78 (0.17)	0.78 (0.11)
TEXT-DQN	0.33 (0.10)	-0.72 (0.17)	0.83 (0.01)
AMN (Actor Mimic)	0.08 (0.03)	-0.84 (0.04)	0.75 (0.005)
VIN (1)	0.59 (0.16)	-0.32 (0.57)	<b>0.85</b> (0.01)
TEXT-VIN (3)	<b>0.73</b> (0.01)	<b>-0.01</b> (0.09)	<b>0.85</b> (0.01)

<b>Bombberman <math>\rightarrow</math> Boulderchase</b>			
<i>Model</i>	<i>Average</i>	<i>Initial</i>	<i>Asymptotic</i>
NO TRANSFER	8.16 (0.79)	2.88 (0.29)	10.67 (1.37)
DQN	7.30 (1.39)	3.77 (0.45)	9.24 (1.83)
TEXT-DQN	7.92 (0.64)	3.44 (0.54)	10.10 (1.69)
AMN (Actor Mimic)	5.58 (0.53)	1.08 (0.38)	8.66 (0.97)
VIN (3)	9.84 (0.51)	3.77 (0.44)	<b>12.22</b> (0.48)
TEXT-VIN (3)	<b>11.17</b> (0.44)	<b>5.37</b> (0.78)	12.08 (0.31)

Table 3: Transfer learning results under the different metrics for different domains. Numbers in parentheses for VIN and TEXT-VIN indicate the  $k$  value for the best model. TEXT- models make use of textual descriptions. Numbers are averaged over 9 independent runs (3 source  $\times$  3 target); higher scores are better; bold indicates best numbers; standard deviation numbers are in parentheses. The max reward attainable (ignoring step penalties) in the target environments is 2.0, 1.0 and at least 25.0 in F&E, Freeway and Boulderchase, respectively.

## 6. Results

We now present empirical evidence that demonstrates the effectiveness of our approach. We first begin by analyzing performance under the transfer condition, followed by the multi-task results and an analysis of the model.

## 6.1 Transfer Performance

Table 3 demonstrates that transferring policies positively assists learning in new domains. Our model, TEXT-VIN achieves superior performance to the baselines on average and initial rewards in all transfer conditions.

### 6.1.1 AVERAGE REWARD

On the first metric of *average reward*, TEXT-VIN (3) achieves a 5% gain (absolute) over the nearest competitor AMN on F&E-1  $\rightarrow$  F&E-2, and a 14% gain (absolute) over VIN (1) on F&E-1  $\rightarrow$  Freeway. In the case of Bomberman  $\rightarrow$  Boulderchase, TEXT-VIN achieves an average reward of 11.17, which is 1.33 points higher than the nearest baseline of VIN, which doesn’t make use of the text. Most of this performance gap stems from the fact that our model is able to learn good policies very quickly, reusing knowledge from the source environment, and hence achieving higher rewards from the start. This fact is also evident from the sample reward curves shown in Figure 5.

### 6.1.2 INITIAL REWARD

On the metric of *initial reward*, all the transfer approaches outperform the NO TRANSFER baseline, except for AMN on Bomberman  $\rightarrow$  Boulderchase. TEXT-VIN achieves the highest numbers in all transfer settings, up to 11.5% better than TEXT-DQN on F&E-1  $\rightarrow$  F&E-2. This demonstrates our model’s effective utilization of text descriptions to bootstrap learning in a new environment. Interestingly, TEXT-DQN demonstrates good jump-start behavior on two of the conditions, but not on F&E-1  $\rightarrow$  Freeway.

### 6.1.3 ASYMPTOTIC REWARD

On the final metric of *asymptotic performance*, our model improves performance over NO TRANSFER and is at par or outperforms the other baselines, except on F&E-1  $\rightarrow$  F&E-2, where AMN obtains a score of 1.64. This is partly due to its smoother convergence;<sup>13</sup> improving the stability of our model training could boost its asymptotic performance. Thus, our approach not only speeds up learning on an unseen domain, but also results in better optimal policies.

Another observation from Table 3 is that TEXT-VIN consistently outperforms TEXT-DQN in all conditions. This demonstrates the importance of having a model-aware policy, which can ground text descriptions onto environment dynamics while retaining flexibility to accommodate different policies for varying task types. TEXT-DQN, on the other hand, couples both knowledge of the environment and the policy into a single network, making it less suitable for transfer.

### 6.1.4 NEGATIVE TRANSFER

We also observe the challenging nature of policy transfer in some scenarios. For example, in Bomberman  $\rightarrow$  Boulderchase, DQN, TEXT-DQN and AMN achieve a *lower* average reward and *lower* asymptotic reward than the NO TRANSFER model, exhibiting negative transfer (Taylor & Stone, 2009). Further, TEXT-DQN has a lower *initial reward* than a vanilla DQN in such cases, which further underlines the need for a model-aware approach to truly take advantage of the text descriptions for transfer.

---

13. This fact is also noted in (Parisotto et al., 2016)



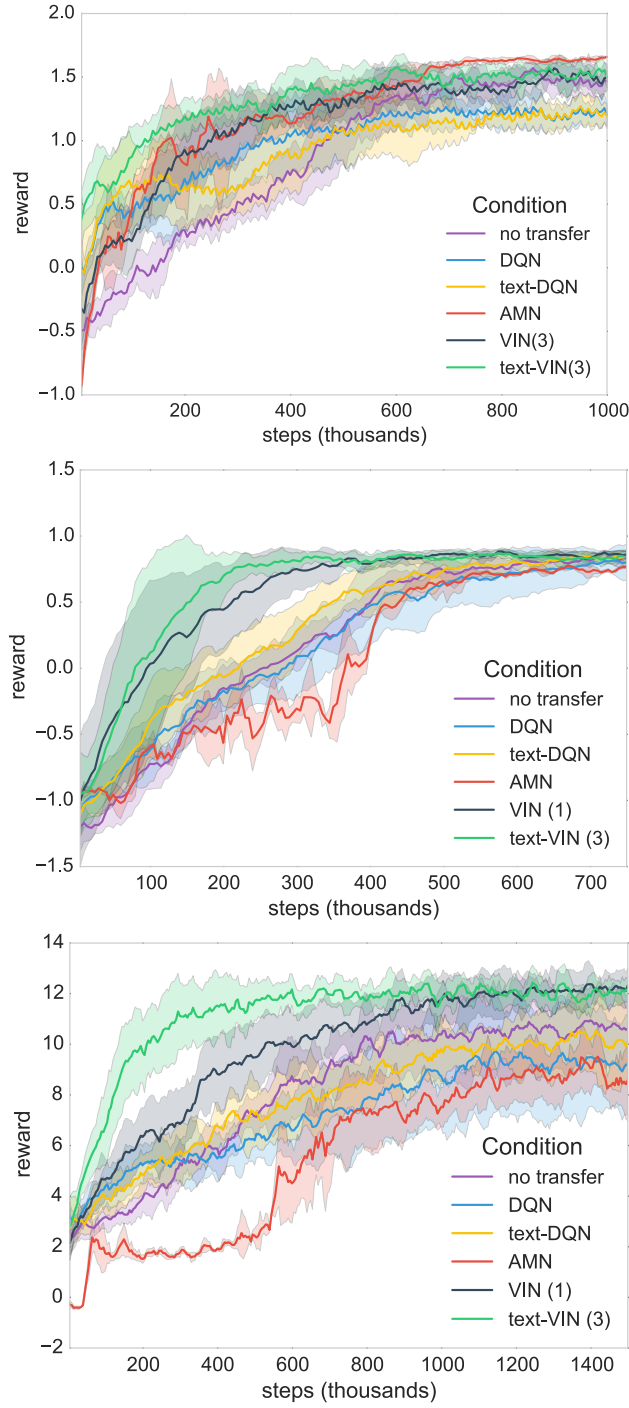


Figure 5: Reward curves for transfer conditions **(top)** F&E-1 → F&E-2, **(middle)** F&E-1 → Freeway, and **(bottom)** Bomberman → Boulderchase (best viewed in color). Numbers in parentheses for TEXT-VIN indicate  $k$  value. All graphs were produced by averaging over 9 runs with different random seeds; shaded areas represent standard deviation.

## 6.2 Multi-task Performance

In addition to transfer, we also investigate learning in the multi-task setting (Caruana, 1997), where the agent learns to perform multiple tasks in a single domain, simultaneously. Specifically, we train a single model, with the same set of parameters, using feedback from all the different tasks. We find that the learning benefits of our model observed in the transfer scenario also hold for multi-task learning, with benefits stemming from both the representation generator as well as the value iteration network. Table 4 details the average reward and asymptotic reward obtained by different models across twenty variants of the F&E-2 domain. Our model is able to use the text to learn faster as well as achieve higher optimum scores, with TEXT-VIN (1) showing gains over DQN of 28.5% and 12% on average and asymptotic rewards, respectively. Figure 6 shows the corresponding reward curves for the various models.

Model	Avg.	Asymp.
DQN	0.80 (0.08)	1.38 (0.07)
TEXT-DQN	0.79 (0.09)	1.45 (0.08)
VIN (1)	1.35 (0.04)	1.61 (0.05)
TEXT-VIN (1)	<b>1.37</b> (0.03)	<b>1.62</b> (0.02)

Table 4: Average (Avg.) and asymptotic (Asymp.) rewards for multitask learning over 20 games in F&E-2. All numbers are averaged over 9 different runs; numbers in parentheses are standard deviations.

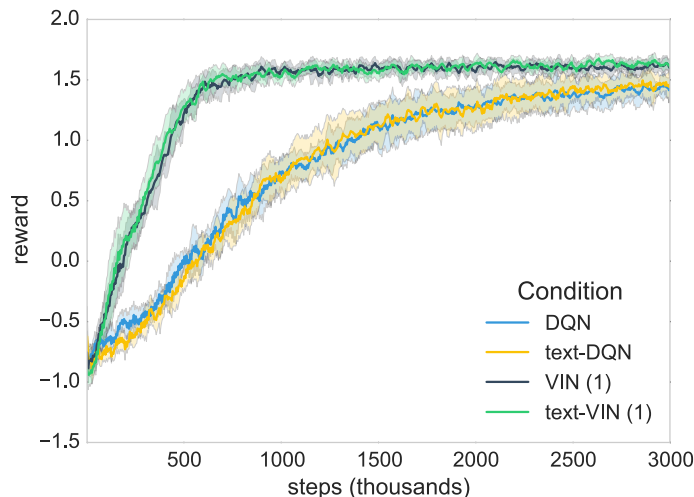


Figure 6: Reward curve for multitask learning in F&E-2. Numbers in parentheses for TEXT-VIN indicate  $k$  value. All graphs averaged over 9 runs with different random seeds; shaded areas represent standard deviation.

Condition	Model	Average	Initial	Asymptotic
F&E-1 $\rightarrow$ F&E-1	Text only	1.64 (0.02)	0.48 (0.09)	<b>1.78</b> (0.00)
	Text+entity ID	<b>1.70</b> (0.02)	<b>1.07</b> (0.19)	<b>1.78</b> (0.00)
F&E-1 $\rightarrow$ F&E-2	Text only	0.86 (0.01)	0.49 (0.04)	1.11 (0.01)
	Text+entity ID	<b>1.32</b> (0.06)	<b>0.70</b> (0.22)	<b>1.50</b> (0.05)

Table 5: Transfer results using different input representations with TEXT-VIN (3). *Text only* means only a text-based vector is used, i.e.  $\phi(s, Z) = v_z(s, Z)$ . *Text+entity ID* is our full representation,  $\phi(s, Z) = [v_o(s); v_z(s, Z)]$ . All numbers are averaged over 9 different runs; numbers in parentheses are standard deviations.

Condition	Model	BOW	LSTM
F&E-1 $\rightarrow$ F&E-2	TEXT-VIN (1)	1.17 (0.08)	1.28 (0.06)
	TEXT-VIN (3)	<b>1.32</b> (0.06)	1.20 (0.06)
F&E-1 $\rightarrow$ Freeway	TEXT-VIN (1)	0.57 (0.09)	0.63 (0.02)
	TEXT-VIN (3)	0.57 (0.05)	<b>0.73</b> (0.01)
Bomberman $\rightarrow$ Boulderchase	TEXT-VIN (1)	10.09 (0.94)	11.10 (0.22)
	TEXT-VIN (3)	9.66 (0.77)	<b>11.17</b> (0.44)

Table 6: Average rewards in Bomberman  $\rightarrow$  Boulderchase with different text representations: Mean bag-of-words (*BOW*), or a vector generated by running an *LSTM*-based recurrent neural network over the entire sentence. All numbers are averaged over 9 different runs; numbers in parentheses are standard deviations.

### 6.3 Analysis

We further analyze the performance of our model by performing ablation studies to investigate the effects of different state and text representations, as well as a qualitative analysis of the value maps produced by the model.

#### 6.3.1 EFFECT OF FACTORIZED REPRESENTATION

We investigate the usefulness of our factorized representation by training a variant of our model using only a text-based vector representation (*Text only*) for each entity, i.e.  $\phi(s, Z) = v_z(s, Z)$ . We consider two different transfer scenarios – (a) when both source and target instances are from the same domain (F&E-1  $\rightarrow$  F&E-1) and (b) when the source/target instances are in different domains (F&E-1  $\rightarrow$  F&E-2). In both cases, we see that our two-part representation results in faster learning and more effective transfer, obtaining 23% higher average reward and 19% more asymptotic reward in F&E-1  $\rightarrow$  F&E-2 transfer (Table 5). Our representation is able to transfer prior knowledge through the text-based component while retaining the ability to learn new entity-specific representations quickly.

#### 6.3.2 TEXT REPRESENTATION: BOW VS. LSTM

Another question to consider is the relative impact of using LSTM vs. mean BOW to generate vector representations from the text descriptions in our model. Table 6 provides a comparison of

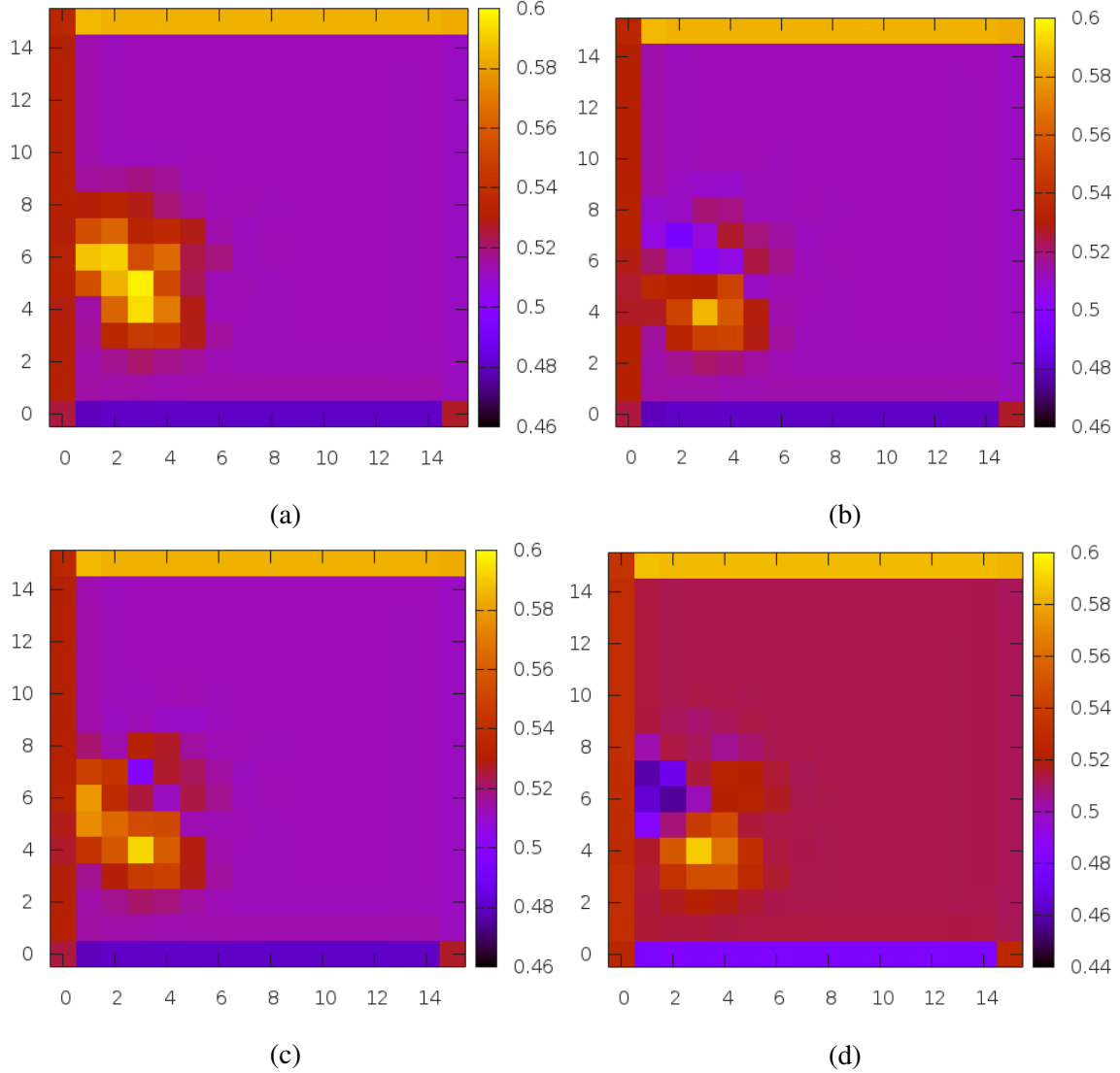


Figure 7: Value maps ( $\hat{V}^{(k)}(s, Z)$ ) produced by the VIN module for (a) seen entity (friend), (b) unseen entity with no description, (c) unseen entity with ‘friendly’ description, and (d) unseen entity with ‘enemy’ description. Agent is at (4,4) and the non-player entity is at (2,6). Notice how the value of cell (2,6) changes with the type of description: higher for ‘friendly’ and lower for ‘enemy’ compared to the case with no description.

transfer performance between these two representations on the different conditions. We observe that using an LSTM provides significantly better results on F&E-1  $\rightarrow$  Freeway and Bomberman  $\rightarrow$  Boulderchase, and slightly worse than BOW (1.28 vs 1.32) on F&E-1  $\rightarrow$  F&E-2. This indicates that a good text representation which can capture linguistic compositionality works better in our model. Exploration of other recently proposed representations like the Transformer (Vaswani et al., 2017) could lead to further improvements.

### 6.3.3 VALUE ANALYSIS

Finally, we provide some qualitative evidence to demonstrate the generalization capacity of TEXT-VIN. Figure 7 shows visualizations of four value maps produced by the VIN module of a trained model, with the agent’s avatar at position (4,4) and a single entity at (2,6) in each map. In the first map, the entity is known and friendly, which leads to high values in the surrounding areas, as expected. In the second map, the entity is unseen and without any descriptions; hence, the values are uninformed. The third and fourth maps, however, contain unseen entities with descriptions. In these cases, the module predicts higher or lower values around the entity depending on whether the text portrays it as a friend or enemy. Thus, even before a single interaction in a new domain, our model can utilize text to generate good value maps. This bootstraps the learning process, making it more efficient.

## 7. Conclusion

In this work, we have explored a novel method to tackle the long-standing challenge of transfer for reinforcement learning. Transferring policies is hard mainly due to the difficulty in learning effective mappings between source and target domains, often resulting in negative transfer (Taylor & Stone, 2009) as a result of incorrect mappings. We have proposed utilizing natural language to drive transfer for reinforcement learning (RL) and shown that textual descriptions of environments provide a compact intermediate channel to facilitate effective policy transfer. In contrast to most existing systems, we have employed a model-aware RL approach that aims to capture the dynamics of the environment. For this, we utilized a value iteration network (VIN), which encapsulates the iterative computation of a value function into a single differentiable neural network. We have also introduced a two-part state representation in order to combine text with input observations. This representation allows us to distill useful information while being robust to incomplete or noisy descriptions.

By effectively utilizing descriptions, our technique can bootstrap learning on new unseen domains. Over several empirical tests across a variety of environments, we have shown that our approach is at par or outperforms several existing systems on different metrics for transfer learning. Our model achieves up to 14% higher average reward and up to 11.5% higher initial reward compared to the most competitive baselines. We have also performed evaluation on a multi-task setting where learning is simultaneously carried out in multiple environments and demonstrate the superior performance of our approach.

There are several possible avenues of future work. One could explore the combination of different transfer approaches. Leveraging language for policy transfer in deep RL is complementary to other techniques such as policy reuse (Glatt & Costa, 2017) or skill transfer (Gupta, Devin, Liu, Abbeel, & Levine, 2017) among other approaches (Du, Gabriel, Irwin, & Taylor, 2016; Tobin et al., 2017; Yin & Pan, 2017). A combination of one of these methods with language-guided transfer could result in further improvements. Another area for investigation is on techniques that can operate without requiring explicit one-to-one mappings between descriptions and entities in the environment. One can either learn these mappings simultaneously with the policy, or operate using descriptions that involve multiple entities and global relations.

In this work, since we factorize our input representation (using both text and direct observations), the method works at least as well as when not using the text descriptions i.e. the model could learn to rely less on the descriptions if they do not contain useful information. However,

one potential case for failure could be if the text contains misleading/incorrect descriptions of the environment. Addressing this issue of robustness to adversarial inputs is another potential direction of investigation. Finally, a major component behind our model’s generalization performance is the value iteration network (VIN). However, in its current form, the VIN requires specifying a recurrence hyper-parameter  $k$ , whose optimal value might vary from one domain to another. Investigating models that can perform multiple levels of recurrent VI computation, possibly in a dynamic fashion, would allow an agent to simultaneously plan and act over multiple temporal scales.

## Acknowledgements

This work was done while Karthik Narasimhan was affiliated with MIT. We thank Adam Fisch, Victor Quach, and members of the MIT NLP group for their comments on earlier drafts of this paper.

## References

- Ammar, H. B., Eaton, E., Ruvolo, P., & Taylor, M. (2014). Online multi-task learning for policy gradient methods. In *International Conference on Machine Learning*, pp. 1206–1214.
- Andreas, J., & Klein, D. (2015). Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Andreas, J., Klein, D., & Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. *ICML*.
- Artzi, Y., & Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1), 49–62.
- Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065.
- Branavan, S., Silver, D., & Barzilay, R. (2011). Non-linear monte-carlo search in Civilization II.. AAAI Press/International Joint Conferences on Artificial Intelligence.
- Branavan, S., Silver, D., & Barzilay, R. (2012). Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43, 661–704.
- Branavan, S., Zettlemoyer, L. S., & Barzilay, R. (2010). Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1268–1277. Association for Computational Linguistics.
- Buhrmester, M., Kwang, T., & Gosling, S. D. (2011). Amazon’s mechanical turk. *Perspectives on Psychological Science*, 6(1), 3–5. PMID: 26162106.
- Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1), 41–75.
- Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4), 613–624.

- Du, Y., Gabriel, V., Irwin, J., & Taylor, M. E. (2016). Initial progress in transfer for deep reinforcement learning algorithms. In *Proceedings of Deep Reinforcement Learning: Frontiers and Challenges Workshop, New York City, NY, USA*.
- Gašić, M., et al. (2013). Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of SIGDIAL*.
- Glatt, R., & Costa, A. H. R. (2017). Policy reuse in deep reinforcement learning.. In *AAAI*, pp. 4929–4930.
- Gupta, A., Devin, C., Liu, Y., Abbeel, P., & Levine, S. (2017). Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*.
- Harrison, B., Ehsan, U., & Riedl, M. O. (2017). Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616*.
- He, J., et al. (2016). Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1621–1630, Berlin, Germany. Association for Computational Linguistics.
- Hemachandra, S., Walter, M. R., Tellex, S., & Teller, S. (2014). Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2623–2630. IEEE.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Isele, D., Rostami, M., & Eaton, E. (2016). Using task features for zero-shot knowledge transfer in lifelong learning.. In *IJCAI*, pp. 1620–1626.
- Joshi, G., & Chowdhary, G. (2018). Cross-domain transfer in reinforcement learning using target apprentice. *ICRA*.
- Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., & George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*.
- Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kodirov, E., Xiang, T., Fu, Z., & Gong, S. (2015). Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2452–2460.
- Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pp. 259–266. IEEE.
- Kolouri, S., Rostami, M., Owechko, Y., & Kim, K. (2018). Joint dictionaries for zero-shot learning. *AAAI*.
- Konidaris, G., & Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning.. In *IJCAI*, Vol. 7, pp. 895–900.
- Konidaris, G., Scheidwasser, I., & Barto, A. (2012). Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 13(May), 1333–1371.

- Konidaris, G. D. (2006). A framework for transfer in reinforcement learning. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*.
- Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39), 1–40.
- Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the national conference on artificial intelligence*, Vol. 21, p. 415. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Matuszek, C., Herbst, E., Zettlemoyer, L., & Fox, D. (2013). Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pp. 403–415. Springer.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning..
- Narasimhan, K., Kulkarni, T., & Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.
- Nguyen, T., Silander, T., & Leong, T. Y. (2012). Transferring expectations in model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2555–2563.
- Oh, J., Singh, S., Lee, H., & Kohli, P. (2017). Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*.
- Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2016). Actor-mimic: Deep multitask and transfer reinforcement learning. *International Conference on Learning Representations*.
- Perez-Liebana, D., Samothrakis, S., Togelius, J., Schaul, T., & Lucas, S. M. (2016). General video game ai: Competition, challenges and opportunities. In *Thirtieth AAAI Conference on Artificial Intelligence*.
- Rajendran, J., Lakshminarayanan, A., Khapra, M. M., Prasanna, P., & Ravindran, B. (2017). *a2t*: Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources..
- Romera-Paredes, B., & Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pp. 2152–2161.
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.



- Ruvolo, P., & Eaton, E. (2013). Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pp. 507–515.
- Schaul, T. (2013). A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pp. 1–8. IEEE.
- Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1312–1320.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.
- Sinapov, J., Narvekar, S., Leonetti, M., & Stone, P. (2015). Learning inter-task transferability in the absence of target task samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 725–733. International Foundation for Autonomous Agents and Multiagent Systems.
- Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pp. 935–943.
- Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. MIT Press.
- Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016). Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162.
- Taylor, M. E., Jong, N. K., & Stone, P. (2008). Transferring instances for model-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 488–505. Springer.
- Taylor, M. E., & Stone, P. (2007). Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, pp. 879–886, New York, NY, USA. ACM.
- Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul), 1633–1685.
- Taylor, M. E., Stone, P., & Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep), 2125–2167.
- Tobin, J., et al. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.
- Vogel, A., & Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 806–814. Association for Computational Linguistics.
- Walter, M. R., Hemachandra, S., Homberg, B., Tellex, S., & Teller, S. (2013). Learning semantic maps from natural language descriptions.. *Robotics: Science and Systems*.

- Wang, Z., Wen, T.-H., Su, P.-H., & Stylianou, Y. (2015). Learning domain-independent dialogue policies via ontology parameterisation.. In *SIGDIAL Conference*, pp. 412–416.
- Yin, H., & Pan, S. J. (2017). Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 1640–1646.