Masters Thesis, Department of Computer Science, The University of Texas at Austin.

The Thesis Committee for Clara Cecilia Cannon
certifies that this is the approved version of the following dissertation:

Supervised Attention from Natural Language Feedback for Reinforcement Learning

Approved by Supervising Committee:

Raymond Mooney, Supervisor

Scott Niekum

Supervised Attention from Natural Language Feedback for Reinforcement Learning

by

Clara Cecilia Cannon

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Masters of Science in Computer Science

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2021

Supervised Attention from Natural Language Feedback for Reinforcement Learning

Clara Cecilia Cannon, M.S.Comp.Sci.
The University of Texas at Austin, 2021

Supervisor: Raymond Mooney

In this paper, we introduce a new approach to Reinforcement Learning (RL) called "supervised attention" from human feedback which focuses on novel task learning from human interaction on relevant features of the environment, which we hypothesize will allow for effective learning from limited training data. We wanted to answer the following question: does the addition of language to existing RL frameworks improve agent learning? We wanted to show that language helps the agent pick out the most important features in its perception. We tested many methods for implementing this concept and settled on incorporating language feedback via a template matching scheme. While more sophisticated techniques, such as attention, would be better at grounding the language, we discovered this task is non trivial for our choice of environment. Using deep learning methods, we translate human linguistic narration to a saliency map over the perceptual field. This saliency map is used to inform a deep-reinforcement learning system which features in the visual

observation are most important relative to its position in the environment and optimize task learning. We establish a baseline model using deep TAMER and test our framework on Montezuma's Revenge, the most difficult game in the Atari Arcade suite. However, our final framework demonstrates the incompatibility of language in the Atari suite in a supervised attention setting. The ultimate result showed that as long as the agent's position in the observation was clear, the model ignores surrounding contextual information, regardless of potential benefit. We conclude that the Atari network of games is unsuitable for grounding natural language in the high dimensional state spaces. Further development of sophisticated simulations is required.

Table of Contents

Chapter 1

Introduction

Advances in machine learning have contributed to impressive progress in the development of reinforcement learning agents' ability to produce robust policies that generalize across differing situations seen and unseen during training (Deisenroth and Rasmussen [2011], Levine et al. [2016]). However, these methods typically only work under carefully designed testing conditions or simulated environments where experts hand select features (Bahdanau et al. [2014]) reward functions, and initial conditions (Skinner [2007]). This scheme works great under the assumption that an RL expert is on hand to ensure the optimal training and testing of the intelligent agent occurs without mishap. However, if we truly wish to develop general purpose agents on a wide scale for real world tasks under the tutelage of a non-expert, a more sustainable model is needed to quickly and robustly characterize novel tasks in unknown environments, utilizing information more intuitive to the average person, rendering the intervention of expert trainers unnecessary.

In the quest to realize this ideal, various approaches have been employed to allow a human user to train a RL agent through our natural forms of instruction and interaction, the most relevant to our project being language.

One of the first ground breaking approaches and the one we reference the most throughout the remainder of this paper is TAMER (Knox and Stone [2008]) and its extension into deep neural architectures, deep TAMER (Warnell et al. [2018]). These are two method for allowing a human to provide continuous evaluative feedback on an agent's performance. They demonstrate how scalar feedback can significantly decrease learning time. While most other areas of research in the Artificial Intelligence (AI) domain might require training episodes to the scale of thousands and hundred thousands, very rarely do we see a RL training scheme less than the order of millions (and that is being conservative). Deep TAMER also addresses the credit assignment problem. By assigning each action sequence a window of relevancy for each received feedback signal, they were able to discern which aspects of the current state made a particular action a good or bad option. However, one must admit the scalar reward as feedback is rudimentary and more tailored to the digital world than our own. We want to show that language feedback is just as valuable if not more significant to shortening agent learning on a larger scale.

Why do we think language is going to be so valuable in the RL domain? First, consider how humans learn and teach each other new skills. When a person is providing feedback or demonstrating a task for another person, they can describe what they are doing in natural language, providing context, clarification, and/or explanations for their evaluations or actions. Therefore, this work focuses on enabling intelligent agents to perform efficient and robust learning from feedback by leveraging auxiliary natural language narration as context.

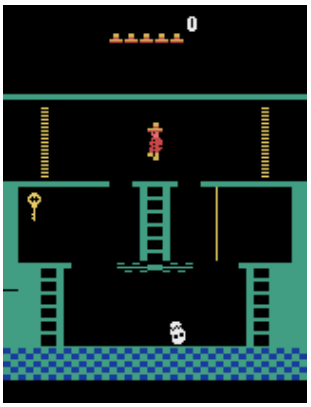

Figure 1.1: Room 1 in Montezuma's Revenge. The sprite/agent must navigate the terrain, avoiding the skull, in order to obtain the key. Once the key is reached the agent can move towards one of the gates and use the key to move onto another level.

We show that this contextual information allows agents to intelligently disambiguate, generalize, and rapidly learn from human instruction a complex task in a sparse reward environment, the Atari game Montezuma's Revenge. We develop, implement, and evaluate our new approach to using language to aid task learning. We were inspired by ideas from language grounding, explanation for deep learning, and learning from rationales.

Our approach uses language narration as a supervisory signal that focuses learning on relevant features of the environment, thereby allowing effective learning from limited training data. Our system pre-processes image inputs using language to focus agent's perception on task or environment relevant features. For instance, given Room 1 (See Figure 1.1) in Atari's Montezuma's Revenge, the human trainer can specify local optimal behaviours with language given the agent's progress towards the goal (key). When the

agent is at the top of the ladder, we do not want the agent to immediately focus on moving left towards the key, as there is no viable path to reach the key's location from the starting position. Instead, a trainer might instruct the agent to move right, away from its primary objective, but along an optimal trajectory for the given environment. To give a more concrete example, consider potentially useful feedback for either of the agents in Figure 1.2. First, we must accurately assess where the agent is in its progress towards task completion. Despite their positions being almost the same, the best sequence of actions each should perform is different. For the left hand side observation, we can now ascertain that the key is stationed above the ladder platform on the left side of the environment. This means that the agent has not yet captured the key. A human trainer can then provided the language feedback, "Climb down the ladder and head towards the key." However, this same feedback would not be appropriate for the agent on the right hand side of Figure 1.2. Notice that the key has moved positions from the above the ladder platform on the left side to the center top of the image. This means that the agent has already captured the key and can now proceed through one of the gates. In this instance, we would want the agent to perform the opposite set of actions compared to the left side agent. By leveraging prior work in video captioning (Venugopalan et al. [2014, 2015]) and template matching, we were able to construct a model and train it to use language embeddings within the environment observation space. The human linguistic narration is used to generate a saliency map over the perceptual field using a combination of methods described in Kaplan

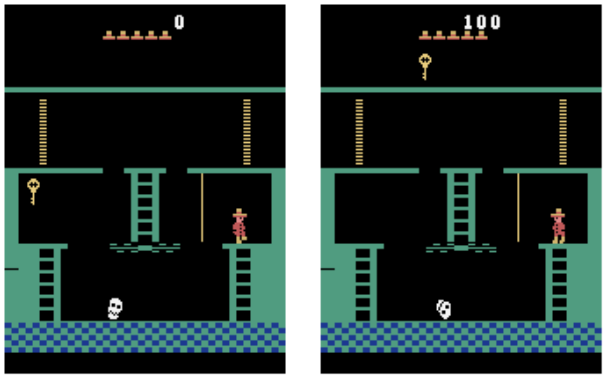

Figure 1.2: Example of two state observations where the agent is almost in the same position for both. However, we need to observe the environment features to determine what language feedback is most appropriate. For the left side of the figure, a human trainer might tell the agent, "Climb down the ladder and head left towards the key." This feedback is not suitable for agent in the right observation, because the agent has already collected the key and must now navigate back the way it came to one of the gates on the top left or right of the image. In this case, a human trainer might say, "Jump left onto the center ladder and climb up to the gates." This figure illustrates the benefit of language feedback in identifying dynamic environment variables, thus informing the agent of how to best adapt its policy.

et al. [2017] and Guan et al. [2020]. Finally, this saliency map is passed to the deep reinforcement learning system from deep TAMER (Warnell et al. [2018]) augmented to accept natural language feedback. Despite our best intentions, we actually discovered that the agent learns more effectively with little to no contextual information in the observation space than with language feedback. We distinguish ourselves from previous work by Goyal et al. [2019] and Kaplan et al. [2017] by focusing on natural language as feedback as opposed to a scalar-valued reward signal or set of instructions. We also do not map the language to the reward function or action selection. Language is incorporated

solely as a supervised attention signal over the features of the high dimensional state observation.

This work covers the development, implementation, and experimental evaluation of our novel method for augmenting agent learning with human feedback by exploiting the information gathered from linguistic narration. We evaluate our approach on Montezuma's Revenge, the most challenging game in the Atari Arcade Suite. We tested our agent on Room 1 using a fixed reward function across all training and testing sessions. Overall, our project dives into and investigates an entirely different use of language– evaluative narration of agent performance. Our new method has the potential to significantly expand the field of RL and but also draw attention to how far we still need to go before at-home personalized robot companions can actually be realized.

1.1 Thesis Outline

The remainder of this thesis document is organized as follows. Chapter 2 covers related works and background knowledge. Chapter 3 explains how we use language to generate saliency maps. Chapter 4 covers our model algorithm and implementation. We describe our experimental set up and discussion of the results in Chapter 5. Chapter 6 and 7 discuss directions for future work and concluding remarks respectively.

Chapter 2

Background and Related Work

2.1 Background

We make the common assumption that our learning task can be represented as a Markov Decision Process specified by the tuple (S, A, T, γ, D, R). S and A are the sets of possible states and actions respectively. T is a transition function, $T : S \times A \times S \to R$, which gives the probability, given a state and an action, of transitioning to another state on the next time step. γ, the discount factor, exponentially decreases the value of a future reward. D is the distribution of start states. R is a reward function, $R : S \times A \times S \to R$, where the reward is a function of the most recent state, the most recent action, and the next state s_t, a_t, and s_{t+1}.

Reinforcement learning algorithms (Sutton and Barto [2011]) seek to learn policies ($\pi : S \to A$) for an MDP that maximize return from each state-action pair, where $\textit{return} = \sum_{t=0}^{t}[\gamma^t R(s_t, a_t)]$. Within reinforcement learning, there are two general approaches to this problem. Policy-search algorithms fix the values of some set of parameters, observe the mean return received for the fixed policy over some number of episodes, and then use a rule to determine what parameter values to try next. The other reinforcement learning

approach models the expected return, or value, of a state or state-action pair when following a certain policy. Usually, the action with the highest expected return is selected (though sometimes with exploratory actions instead), and the agent updates the expected returns based on its experience. We use the latter approach when formulating our problem.

2.2 Sample Efficient Reinforcement Learning

Sampling efficiency is a very important consideration for any RL agent and requires careful consideration when selecting an appropriate algorithm, domain, and training scheme. In a nutshell, sample efficiency is concerned with the amount of experience an intelligent agent needs in an environment (e.g. actions and number state/rewards transition observations) during training in order to perform well on whatever task it has be assigned to do. Intuitively, a sample efficient algorithm squeezes every bit of information it can out of a single generated experience or uses prior knowledge to maximally optimize its policy. Ideally, we would like to design an algorithm that can get the most out of every sample, because more often not, the data we are using to train these agents is not infinite.

As human beings, we are not very much affected by this paradigm. For us to learn a new sport whether it be a golf swing or a perfect spiral pass of a football, it only takes a few examples before we acquire the basic mechanics and understanding to attempt it ourselves. In this light, humans beings are the most "sample efficient" agents in existence. In contrast, modern RL algorithms

need millions times more interaction and data within an environment, making them rather sample inefficient. Many different approaches have been proposed to tackle this problem, including use of different sampling algorithms (e.g. importance sampling) (Schulman et al. [2015], Foerster et al. [2017], Wang et al. [2016], Munos et al. [2016]), designing better reward functions (Laud [2004]), feature engineering (Chandrashekar and Sahin [2014]), designing good latent representations for high dimensional states (Vezzani et al. [2019], Allshire et al. [2021]), transfer learning models (e.g. learning from demonstration and learning from feedback) (Hester et al. [2018], Warnell et al. [2018]), and combining model-based and model-free methods (Qu et al. [2020]). In this paper, we are most interested in the transfer learning methods, specifically learning from feedback.

2.3 Language in Reinforcement Learning

Humans use a variety of means and multi-modalities to communicate information with each other. However, arguably one of the most essential is language. Wouldn't it be wonderful if we could use our natural inclination for using language to communicate as a means for teaching an RL agent? Recent advances in language embedding and representations have led to many proposed model architectures that learn domain knowledge via text and use this information to make decisions later in training/learning/model pipeline. Here, we discuss some methods of interest that are relevant to our research.

Incorporating natural language into the RL algorithm whether as a pri-

mary input or a means by which the agent interactions with the environment is hard. Despite it being second nature to most human adults, using natural language requires common sense, world knowledge, and context to resolve ambiguity. We also do not want to waste valuable computational resources on a language encoding, so the cheaper the better. The main question is whether agents can learn from accessory information encoded using language, along with rewards or demonstrations, to improve generalization and sample efficiency. Many sub areas of RL have emerged towards this goal, such as instruction following, mapping language to reward, and embedding the language in the state and action space (Luketina et al. [2019].

For instruction following, agents are asked to preform tasks defined by sequences of natural language instructions. Effective agents in this domain are supposed to execute actions corresponding to an optimal policy or reach a goal specified by the set of instructions. Depending on the system, some agents can even generalize to brand new instructions during testing. Another useful application of natural language instructions is to infer a reward function for an intelligent agent to optimize. This is particularly advantageous when the environment reward is sparse or unavailable entirely. In general, fully incorporating the complete set of instructions is often unnecessary for solving the underlying RL problem. However, the encoded information extracted from the language can assist in learning a good policy or provide auxiliary rewards (Goyal et al. [2019]).

Pulling this thread of inquiry further, we need to consider the work by

Kaplan et al. [2017], which is probably the most similar to ours. Using the Atari game Montezuma's Revenge, they come up with an agent that learns to beat the first level of this game with the aid of natural language instructions. The agent uses the embedded environment observations and natural language information to monitor its progress through a list of English instructions. It receives positive reward for successfully completing an instruction and increasing the game score. Our approach differs in that our agent is not using the extracted language embedding to infer reward. We use our language as supervisory feedback signal to train our model.

Language allows us to abstract, generalize, and communicate information ranging from plans to intentions and detailed requirements. Building intelligent agents with similar capabilities would be hugely beneficial to society. However, agents trained using more traditional approaches in RL lack such means of transferring information as humans do, and struggle to efficiently learn from interactions with rich and diverse environments, which is quintessential for real world deployment and sustainable operation. Most real-world tasks would require artificial agents to process and interpret language. Our novel RL model bring us one step closer towards complete linguistically capably agents.

2.4 Saliency Maps

Deep learning systems are often treated as black boxes, where the inner workings of the systems are esoteric and hard to interpret. This led to work

in generating explanations that help interpret the decisions made by deep networks such as CNNs (Goyal et al. [2016]) and RNNs (Ramanishka et al. [2017]). Systems such as Grad-CAM (Selvaraju et al. [2017]) and Caption-guided visual saliency (Ramanishka et al. [2017]) are able to analyze a network's processing of a particular example and generate a "heat map" showing which features of the input most influenced the generated output. Caption-guided visual saliency takes a video captioning neural network and a given input/output pair, and produces heat maps over the input frames denoting which parts most influenced the computed output. EXPAND (EXPlanation AugmeNted feeDback) by Guan et al. [2020] uses a human in the loop RL framework to provide visual explanations from saliency maps and binary feedback. They showed that the addition of state salient information boosts agent performance. Along this vein, we used a technique called template matching in conjunction with the Gaussian perturbation over the pixels to produce heat maps from state observations. Template matching is a technique that identifies the parts on an image that match a predefined template. We generate these heat maps from natural language feedback and use them to supervise the training of our model weights.

2.5 TAMER and Deep TAMER

In our work, we concentrate on two of the most popular frameworks for learning from human feedback, TAMER (Knox and Stone [2008]), and Deep TAMER (Warnell et al. [2018]). We augmented the Deep TAMER frame-

work to use linguistic information collected from human feedback to produce saliency maps.

The TAMER Framework is an approach to the Shaping Problem (Bouton [2007]). TAMER assumes human reinforcement to be fully informative about the quality of an action given the current state. It uses established supervised learning techniques to model a hypothetical human reinforcement function, $H : S \times A \rightarrow R$, treating the scalar human reinforcement value as a label for a state-action sample. Briefly, the TAMER algorithm operates in the following manner. First, the agent receives a scalar reward from the human trainer for the previous time step. If the human reward is nonzero, then the error is calculated as the difference between the human reward and model predicted reward, $\hat{H}(s, a)$. The loss is then propagated backward, along with the previous feature vector, to update the model. In the forward pass, when the agent takes a single step in the environment to reach a new state, it selects the action with the largest predicted reward according to the model.

Deep TAMER, an extension of the TAMER framework, leverages the representational power of deep neural networks in order to learn complex tasks in just a short amount of time with a human trainer. We use this approach to implement our supervised attention model with human feedback instead of a binary reward input from the human trainer.

Chapter 3

Language Guided Template Matching

Our approach uses language narration to supervise the template matching saliency map generation. We hypothesise that instead of allowing the model to see/have access to all image features for every step in the training episode, thereby having to figure out on its own which features are the most important predicting reward and which ones can be ignore through a series of trials and errors, training time will be reduced if we simply hand over the features of the image we think are most important. Having a selection of the features is useful for efficiency but designing them by hand is not scalable – therefore we use language. We test this by transferring the language information obtained from the human trainer to the image observation and outputting a heat map. The saliency map is supposed to help focus model weights on relevant features of the environment, thereby allowing more effective learning from limited training data. We follow the language template matching scheme outlined in Kaplan et al. [2017] with free form natural language feedback to generate a mask. We then apply a Gaussian filter over the original image to perturb the pixels and combine this blurred observation to obtain the final saliency map (Greydanus et al. [2018]). Applying Gaussian perturbations over irrelevant regions adds spatial uncertainty and motivates the agent to focus

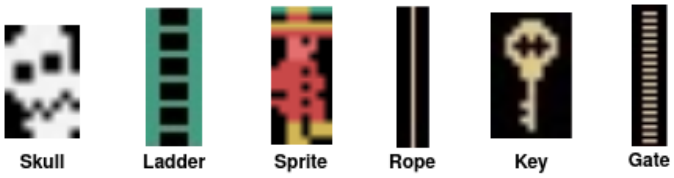

Figure 3.1: This figure depicts the templates we extracted from Room 1 of Montezuma's Revenge.

more on the clear relevant regions. We experimented with other perturbation methods, but found the Gaussian filter with blur radius of 3 worked the best for our purposes.

3.1 Using Language to Produce Saliency Maps

A fundamental challenge in RL is given a reward for a particular sequence of states and actions, ascertaining which features of the states and actions are most important for determining the reward. Using language to describe why an agent's actions were good or bad is valuable information. Our approach allows a neural model to focus on relevant state features communicated via a saliency map (aka heat map) highlighting the importance of image features through pixel weighting or masking.

We first implemented an algorithm that maps the language to the perception of the environment using templates we created (See Figure 3.1). Formally, given a natural language feedback provided from a synthetic language generator, we parse the utterance into separate word tokens. With these tokens, we preform a search across the generated template for the Montezuma's

Revenge environment. The templates are tightly bounded images of important obstacles and features of the environment, including ladders, gates, the agent, the skull, and the key. We select the templates by matching word tokens in the feedback language to the template name. Once all of the relevant generated templates are gathered based off of the language feedback, we compute a mask over the observation features. The mask leaves the important features in the observation space and masks out all others. We then apply a Gaussian filter to the original image which consequently blurs pixel appearance. After applying the mask to the now blurry observation, the resulting array contains the features with relevant object information with unimportant features obscured (see Figure 3.2). In essence we generate heat maps over the input frames denoting which objects most influenced the computed output. This heat map is then passed as an input during training to deep RL model in a supervised setting, allowing the system to use the language to improve learning from feedback. We experimented with using saliency maps without the Gaussian filter perturbation over the pixel space. These saliency maps faded the irrelevant features, however the model was still able to pick them out. Visually, we can see illustrated in the right most image of Figure 3.3. Even though the background pixels are muted in color, they are still clear and easy to distinguish. There is not enough spatial uncertainly introduced by this variety of saliency map to motivate the model to ignore the extraneous features.

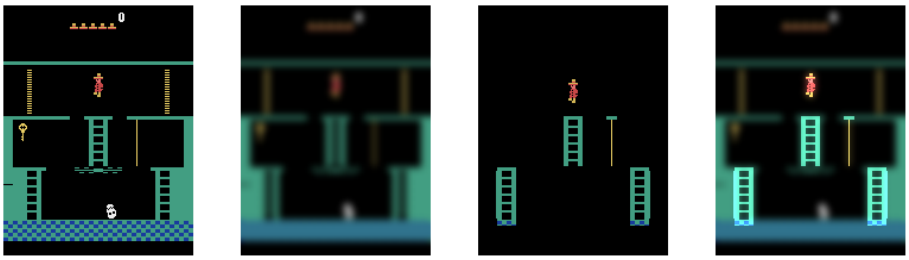

Figure 3.2: Saliency Map Generation. We produce heat maps for the model based on a pseudo random interval to authentically simulate a human in the training loop. A Gaussian filter with blur radius of 3 is applied to the observation image. The resulting vector is added to the template mask. The final saliency map is depicted in the far right image of the figure.

3.2 Template Matching in Atari

We use the cv2 library in Python to perform the template matching. Template matching is a method for searching and finding the location of a template image in a larger image. The OpenCV library comes with a built in that slides the template image over the input image and compares the template and patch of input image under the template image. It returns a grayscale image, where each pixel denotes how much the neighbours of that pixel match with the template. Once we have the result, we can find the maximum/minimum value. However, this will not provide us will all the locations of an object when multiple occurrences of the same template are found.

For our purposes, we implemented template matching with multiple objects. Consider Room 1 in Montezuma's Revenge. If the human trainer provides feedback such as "Climbing down the ladder was a good action", we want to locate all of the ladders across the observation so the model can

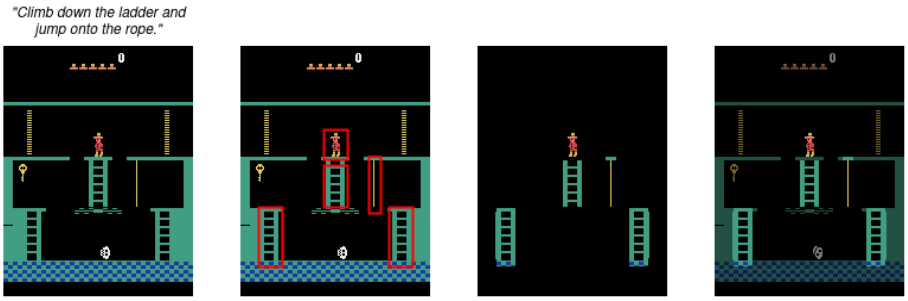

Figure 3.3: Template matching in Montezuma's Revenge. Given the example feedback language "Climb downs the ladder and avoid the skull", we can see that the template matching module matching the text tokens to the correct templates, in this case the ladders and the skull. Once the templates are matched and identified in the original observation, we compute the mask of the observation where on the pixels matching the selected templates remain. We always include the template for the agent in the mask generation. We are then able to produce the far right image from just the mask and observation, where the important pixels can be seen as the foreground and all others are faded into the background.

interpret that ladders should be associated with climbing up and down actions. In this case, we used a thresholding approach. If a pixel value is greater than the threshold value, it is assigned 1 else it is assigned 0. In our case, the threshold is 1, meaning the pixel in the template is a direct match for the pixel in the observation image. Next we compare a template against overlapped image regions. As we slides through sections of the image, we compare the overlapped patches of size $w \times h$ against the template and store the comparison results. We used the template matching normalized correlation coefficient comparison method:

$$\hat{I}(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{3.1}$$

where I denotes image, T template, $\hat{I}$ result. After the function finishes the comparison, the best matches are found as global maximums.

Chapter 4

Deep TAMER + *Feedback*

Deep TAMER (Warnell et al. [2018]) extends TAMER (Knox and Stone [2008]) a LfD framework. A human observes an autonomous agent trying to perform a task in a high-dimensional environment and provides scalar-valued feedback as a means by which to shape agent behavior. Deep TAMER + *Feedback* enriches the Deep TAMER observation space with natural language feedback, potentially enabling better sample efficiency and training time.

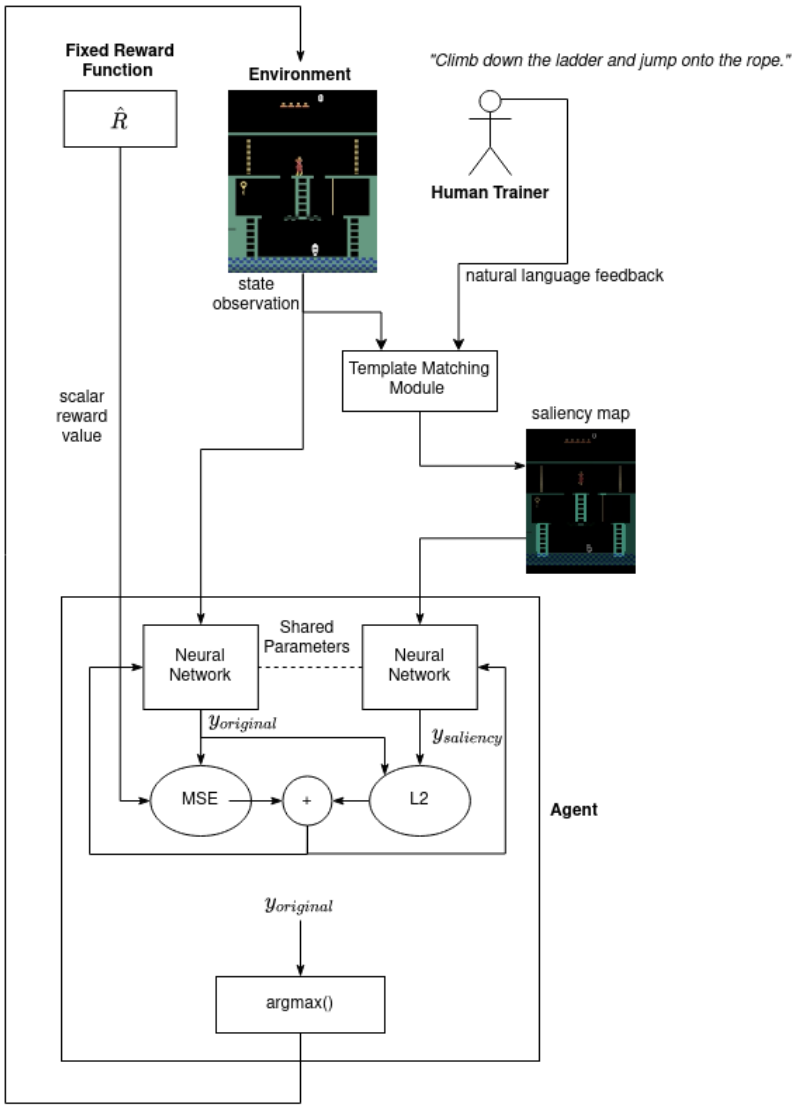

Figure 4.1: The TAMER + *Feedback* framework. A human trainer observes an autonomous agent trying to navigate and perform a task in a high-dimensional environment and provides natural language feedback. The language is used to generate a saliency map which aids the agent in learning the parameters of a deep neural network, that is used to predict the reward. This prediction then drives the agent's behavior policy. We use the Atari game of Montezuma's Revenge as our environment, which uses a pixel-level state space.

4.1 Algorithm

We use the problem formulation proposed in Warnell et al. [2018] with a few modifications (See Figure 4.1). Instead of a binary reward signal, we use language feedback collected from human trainers. We also modify the binary reward signal in order to standardize non-environment reward during training. More formally, Let S denote the set of states in the agent's environment, and let A denote the set of legal actions the agent can execute. The execution of actions $(a_1, a_2, \cdots, a_n)$ result in a state trajectory $\tau = (s_0, s_1, s_2, \cdots, s_n)$ where n is either the terminal state in an episode of the training horizon ($<$ the max number of training steps). The human trainer observes the state trajectory and provides natural language feedback, $(\ell_1, \ell_2, \cdots)$, that convey their personal evaluation of the agent's behavior. We hard code the scalar-valued feedback signals, $(h_1, h_2, \cdots)$, as a means to optimize the training process and allow for uniformity across training sessions. We hypothesize that improved agent performance would be the result of quality language feedback, not a dense reward function. We model the reward signal $R(s, a)$ for a given state and action pair by taking the Euclidean distance d between the agent and nearest critical point along the optimal trajectory, determined by a human expert (see Figure 4.2).

$$d = \sqrt{(x - x')^2 + (y - y')^2} \tag{4.1}$$

$$R(s, a) = \begin{cases} 1 & d < \alpha \\ 0 & \alpha < d < \beta \\ -1 & \beta < d \end{cases} \tag{4.2}$$

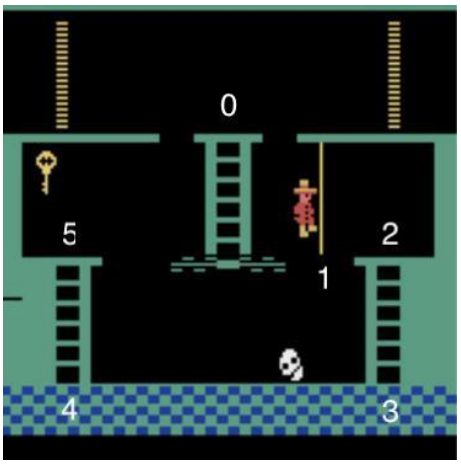

Figure 4.2: Map of the critical positions encoded in Room 1 of Montezuma's Revenge. These positions inform the reward function $R(s, a)$ the appropriate scalar reward signal to feed to the environment.

where the lower threshold β and upper threshold α are set the 8 and 20 respectively. We determined the best threshold values from repeated trial and error (or hyperparameter search). We also penalize the agent with negative reward if it is stationary in the same state for more than 10 time steps. We do this because early on in training, we noticed the agent tends to hide in the regions of the environment with 0 reward. Thus to encourage exploration and prevent the agent from getting "stuck", we include this extra reward shaping condition.

When ℓ_i is available, we use it to compute the saliency map using the template matching technique described in Chapter 3. Then, we compute the predicted reward given the original state observation s,and the predicted re-

ward with the saliency map s'. When the model is updated and the loss propagated backward through the weights, we use these two rewards to compute an $L2$ regularization and add it onto the calculated MSE loss. We include this additional loss to force the model weights to 0 for unimportant state observation features.

$$L2 = \sum_{i=1}^{n} |y_{original} - y_{saliency}|^2 \tag{4.3}$$

where $y_{original}$ is the predicted reward when the original observation is fed as input into our deep model and $y_{saliency}$ is the predicted reward given the saliency map as input. When ℓ is not available, we ignore the L2 loss entirely and set it to 0. Now, our completed loss function that we aim to minimize looks like:

$$L = MSE + \alpha \cdot L2 \tag{4.4}$$

where the regularization rate is 0.0001. We adopt a greedy policy for action selection. Given the vector of predicted rewards from the model, the agent selects the action associated with the greatest reward.

Chapter 5

Experiments and Results

We pre-trained an auto encoder for feature extraction on the image data set from Goyal et al. [2019], which was adapted from the Atari Grand Challenge data set (Kurin et al. [2017]), which contains hundreds of crowd-sourced trajectories of human game plays on 5 Atari games, including Montezuma's Revenge. We only used image from Room 1 in Montezuma's Revenge, which amounted to roughly $2,000$ images with which to train our auto encoder.

We used the vanilla Deep TAMER model as our baseline. While the Deep TAMER with Credit Assignment was the best performing model in Warnell et al. [2018], the success was largely based on the hand selected credit assignment window. We chose a simpler TAMER implementation to narrow the field of experimental variables in order to better highlight the effects language had on agent performance. We focus more on exploring the benefit of language throughout the training process rather than task completion. We set the training interval to $10,000$ steps, with 100 warm up steps at the beginning of each trial to introduce variety in the agent starting position. We compute performance as an average across 10 sessions. This was necessary due to the extreme variability in training sessions. We could not rely on a sin-

gle session to demonstrate an authentic data trend. Our experimental design was constructed to reveal the effects natural language feedback have on agent learning via the generated heat maps. The ultimate results showed no correlation between added language and improved learning, however they led us down an intriguing investigation into the suitability of Montezuma's Revenge for grounding languange in deep RL frameworks. Not only does a small subset of language feedback appear to harm agent learning, but the model requires little to no information about its surrounding environment to memorize the optimal trajectory and translate that knowledge to an effective policy.

Current Position	Feedback
-1	"Go back to the top of the ladder and jump onto the rope."
0	"Jump right from the ladder onto the rope."
1	"Jump right from the rope onto the lower platform."
2	"Climb down the ladder onto the ground."
3	"Move left and avoid the skull."
4	"Climb up the ladder."
5	"Jump up to get the key."

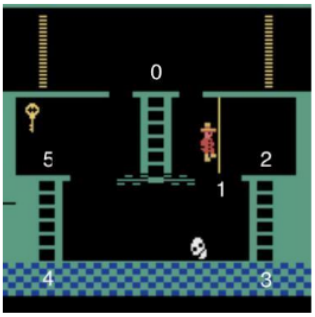

Figure 5.1: Synthetic Feedback Language. Table showing the synthetic language assigned to each critical position along the optimal trajectory. The position -1 is everywhere else in the environment that is not marked with a position numbered $0-5$. Each utterance contains the template name that we reason are important for the model to focus in that stage of the task.

Our initial intention was to collect natural language feedback, however the results of our preliminary experiments revealed data collection would be wasted in the current environment. However, we still believe it is relevant to review the synthetic data set we used to created the saliency maps for training

the vanilla TAMER model. The language feedback was hard coded into a generator function that returned the most applicable feedback given the agent's current position and its previous position (See Figure 5.1). As mentioned previously, we captured the agent's progress throughout Room 1 in Montezuma's Revenge by marking "critical positions" along the optimal trajectory set by a human expert. When the agent found itself in one of these defined critical positions, the language generation function would deliver feedback aimed toward getting the agent to the next critical position. When the agent is not on the optimal path or in transition from one critical position to the next, its position is labeled -1. When deciding which feedback is best suited for transitionary states (when the agent's current position is -1), we need to consider its position along the optimal path. We find the agent's distance to the nearest critical position, all lives intact, and send feedback relevant to reaching that location. Because the saliency map is being generated via template matching scheme, the actual semantics of the feedback is less important than the content from which we extract tokens used to search the template space for matches. The agent often gets "stuck" in the upper left and right corners of environment near the gates. In this case, we use a position buffer to track its movement. If the position buffer contains the same coordinates, we set the appropriate flag and stop feeding the model saliency maps. The intuition behind this decision is that the agent will become unstuck faster when the model receives complete information of the environment. Since we do not use live human trainers for feedback, we also needed to reason about how often

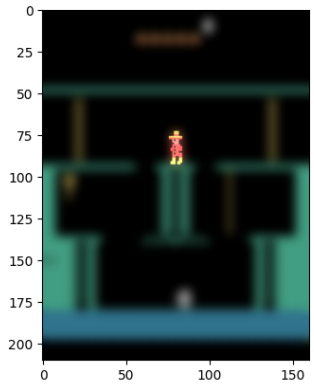

Figure 5.2: Blind Agent. An example of the generated saliency map for the "blind agent" trials. Instead of using language feedback and the full template matching scheme, we created a mask using only the sprite template. We applied the Gaussian filter to the original observation image and applied the agent mask. The obscured environment features focused the model's attention directly on the agent's state.

feedback would be available, and consequently saliency maps. It is unrealistic to expect a human to provided natural language feedback at every time step. Not only would this precedent be laborious for the participant, but contradictory to our goal of making agent training faster and more efficient. We settled on a feedback interval of 10 seconds after several trials, determining the parameter value struck the right balance between often enough and too much. That being said, none of our experimental set up with the natural language yielded improved results over vanilla TAMER with no language feedback. As we puzzled over this dilemma, we began excluding more an more of the information from the saliency map, until the natural language essentially became an accessory, not the cornerstone of the self supervised network. We called

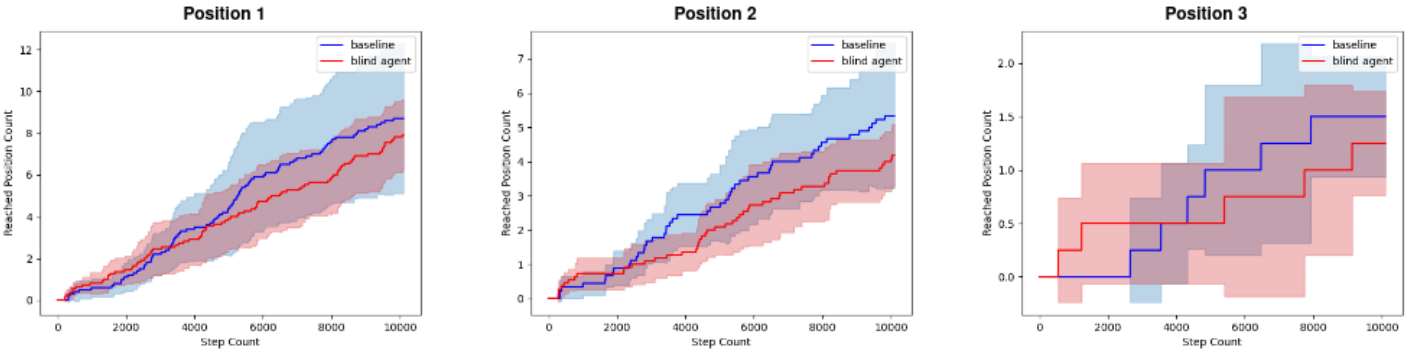

Figure 5.3: Comparison of the vanilla TAMER baseline (blue) and the TAMER model with blind agent observations without $L2$ loss. Shows the average agent progress to critical positions 1 - 3 with 95% confidence interval over 10 trials.

these the "blind agent" trials, where the saliency map obscures everything in the state observation except the agent's position (see Figure 5.2). First, we replaced the observation with the blind agent observation and set alpha, the regularization rate for the $L2$ loss, to 0. Setting alpha to 0 neutralizes the $L2$ loss and effectively reverts the model to vanilla TAMER, which is trained only using an MSE loss. This experiment helped us determine if the model required information about any other environment features besides the agent position to better its policy. Figure 5.3 compares the baseline vanilla TAMER model with TAMER given the blind agent as the observation (red). Even though the network had a hard time extracting any type of environment information from the state observation, overall performance was hardly effected. The blind agent model predicted the reward with better accuracy, which we can surmise from the tightened confidence interval. We can conclude the language

is irrelevant when the agent position is clear. A dense reward policy is more important for task completion in this environment than high dimensional state features. The model ignores the information communicated via the saliency map and does not pay attention to any other objects in the scene because they are stationary. Non stationary environment obstacles, such as the skull, or rearrangement of the environment layout within the same room would force the model to seek auxiliary information and adapt to the changing surroundings. Given enough training samples, the model has ample time to memorize sequence of actions correlated with the optimal trajectory. Montezuma's Revenge is too uniform and consistent for language to gain any kind of holding or leverage over the model feature weights. However, for tasks with more variety and less uniformity/consistency in the state space, the model would be forced out of its "comfort zone". Pure memorization would fail to yield a successful policy and the added information contributed by language feedback would then be invaluable. Our next experiment reinstated the full image observation, returned the regularization rate to 0.0001, and replaced the saliecy may with the blind agent at every time step. In essence, this is equivalent to natural language feedback saying, "Pay attention to where you are" in our framework. The results in Figure 5.4 requires an equally enlightened and thought provoking discussion as the previous experiment. Figure 5.4 displays the comparison amongst the vanilla TAMER baseline, the TAMER model with natural language feedback provided every time step (which means the model is constantly being trained with the additional $L2$ loss), and the TAMER model with blind

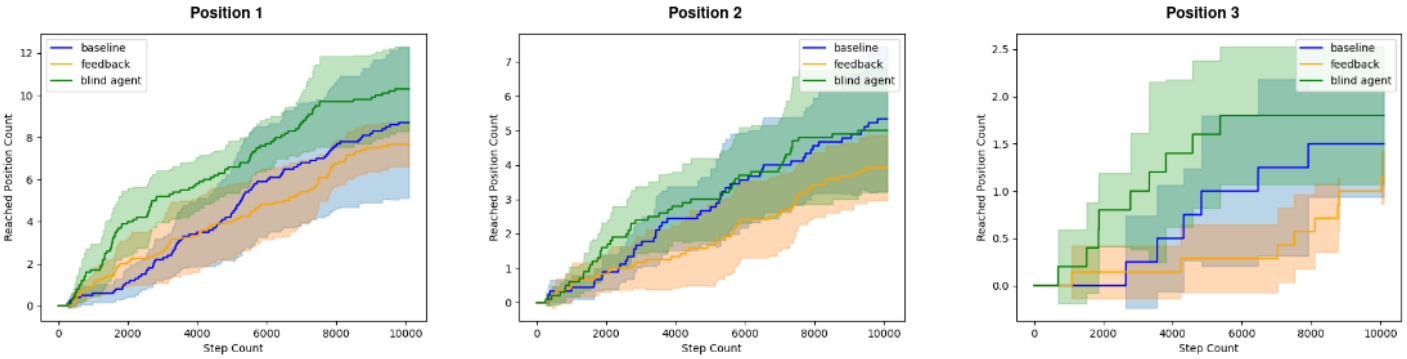

Figure 5.4: Comparison of the vanilla TAMER baseline (blue), TAME with natural language feedback (orange), and TAMER with blind agent saliency maps (green). The saliency maps were produced every time step. Shows the average agent progress to critical positions 1 - 3 with 95% confidence interval over 10 trials.

agent saliency map being generated every time step. The TAMER model with the blind agent saliency map performed better than the baseline and TAMER with language feedback for reaching positions 1 and 3. At first glance, this may seem like a strange result, but upon closer inspection it supports the results of Figure 5.3. Passing the blind agent saliency map the model is equivalent to a human trainer supplying "focus on your own position" as language feedback. We have already established that the model only needs a clear depiction of the agent position in the state space to learn a decent policy. Reinforcing that concept with language in our proposed framework allows the model the reach that ultimate conclusion faster.

We tested the statistical significance of our results using an unpaired t-test and found the difference between the baseline model and the language augmented model and the difference between the baseline model and the blind

agent model were not statistically significant at 5% significance level. Even though we fail to reject the null hypothesis, the lack of significance between TAMER with saliency maps and the TAMER baseline still supports our major finding — language is irrelevant to improving agent's success for the proposed task.

To further corroborate the precedent revealed by the data, we trained a model under the opposite conditions. Instead of the concealing the environment, we hid the agent's position. We do not show these results as there was nothing for us to display. After ten independent trials, the agent was not able to make any progress along the optimal trajectory. While observed, the agent appeared to either walk off the platform to its demise or remain stationary in its starting position.

Our findings shed new light on the feasibility of successfully testing our original hypothesis on video game environments. We can determine that Montezuma's Revenge, the hardest task in the Atari game suit, is incompatible with augmenting the state space with language feedback. Unfortunately, the affect of language as self attention or a supervisory signal over the perceptual field on agent learning and sample efficiency is still an open question. However, we have uncovered certain desirable and undesirable environmental qualities that lend themselves to solving this task.

Chapter 6

Future Work

More investigation into RL environments well matched for language feedback is needed. Our results demonstrate that the wrong choice in domain renders the addition of language ineffective. Other tasks with attributes that indicate better chances of success need to be explored. These qualities include non static obstacles and changing environment dynamics. Radically changing the agent's starting position and the goal location, along with the room layout all promise good integration with language. Montezuma's Revenge possesses some non-static obstacles and changing environment dynamics in different levels, however with the sparse environment reward, the agent struggles to pass Room 1. Finding environments with an innate dense reward function and drastic fluctuations in layout would removing the agent's ability to memorize a single optimal trajectory. It would have to pay attention to the language feedback to determine with features are most important and successfully complete the task. Despite Montezuma's Revenge reputation for being the most difficult game in Atari for RL models, we think that it might still be too easy for language to have a remarkable impact. Increasing the task difficulty could allow for a richer variety of language feedback and potential state features. An agent would have limited interaction with the complete set of environ-

ment experiences, requiring additional information from other inputs, such as language feedback. One of the most time consuming aspects of any research project is selecting the right environment to test the hypothesis. Video game environments are portable and easy to integrate with existing software platforms. Creating deeper network architectures for RL is great, but it is not beneficial if the model only needs the agent's relative position. Our results advocate for environments where all of the state information must be utilized for the model to comprehend a task and generalize to other environments.

The realm of extending RL feedback methods to incorporate language is full of potential and many unexplored avenues from a model architecture perspective. Template matching is a simple, effective technique for saliency map generation. However it is not scalable because each template is constructed by hand. For larger and more varied environments, this technique becomes infeasible. This module could be replaced with a model that generates more authentic heat maps. Video and image captioning (Selvaraju et al. [2017], Ramanishka et al. [2017]), referring expressions (Gundel et al. [1993], Clarke et al. [2013], Krahmer and Van Deemter [2012]), and learning a shared embedding using contrastive loss (Oord et al. [2018]) could be viable upgrades. Incorporating the saliency maps into the policy training is another area for future investigation, particularly the embedding an attention mechanism within the model architecture. We also do not consider the addition of language feedback to COACH (Arumugam et al. [2019]), another popular learning from feedback model. This field opens the door to an array of exciting advancements and

we barely scratched the surface of integrating sophisticated language with an agent learning scheme.

Chapter 7

Conclusion

We explore a novel approach to Reinforcement Learning (RL) that uses natural language feedback to provide supervised attention. We compared a TAMER model with the additional of natural language generated saliency maps to the baseline and found language had no significant improvement or worsened performance on Montezuma's Revenge, a popular video game in the Atari Arcade suite. This led to the construction of a blind agent training scheme, where the salient information passed to the model only contained the agent position. These experiments validated our growing suspicions that language was ineffective for Montezuma's Revenge. As long as the model has access to state features containing the agent's position, it will ignore other potentially useful features suggested by the language generated saliency map. We concluded that not all RL environments are well suited for the additional on language in the latent observation space and requires further investigation. Ultimately, we failed to show that our framework improved the effectiveness of agent learning from limited training data. However, we discovered that our inability to make any deterministic claim about our framework resulted primarily from our choice in domain and not from error in the incorporation of natural language feedback. We need to test our framework on more dynamic domains

where the environment changes from episode to episode and the language can focus the agent on the important features of the dynamic environment. While our initial approach did not produce satisfactory results, the exploration into the this intersection of natural language processing, grounding language, and RL will continue.

Bibliography

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg. Laser: Learning a latent action space for efficient reinforcement learning. *arXiv preprint arXiv:2103.15793*, 2021.

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

D. Arumugam, J. K. Lee, S. Saskin, and M. L. Littman. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019.

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

M. Bansal, C. Matuszek, J. Andreas, Y. Artzi, and Y. Bisk. Proceedings of the first workshop on language grounding for robotics. In *Proceedings of the First Workshop on Language Grounding for Robotics*, 2017.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. Technical report, Springrer, 2008.

M. E. Bouton. *Learning and behavior: A contemporary synthesis.* Sinauer Associates, 2007.

G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

A. D. F. Clarke, M. Elsner, and H. Rohde. Where's wally: the influence of visual salience on referring expression generation. *Frontiers in psychology*, 4:329, 2013.

M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

J. Donahue and K. Grauman. Annotator rationales for visual recognition. In *2011 International Conference on Computer Vision*, pages 1395–1402. IEEE, 2011.

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforce-

ment learning. In *International conference on machine learning*, pages 1146–1155. PMLR, 2017.

P. Goyal, S. Niekum, and R. J. Mooney. Using natural language for reward shaping in reinforcement learning, 2019.

Y. Goyal, A. Mohapatra, D. Parikh, and D. Batra. Towards transparent ai systems: Interpreting visual question answering models. *arXiv preprint arXiv:1608.08974*, 2016.

S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

L. Guan, M. Verma, S. Guo, R. Zhang, and S. Kambhampati. E, 2020.

J. K. Gundel, N. Hedberg, and R. Zacharski. Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307, 1993.

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

T. Jie and P. Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in*

Neural Information Processing Systems, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/35cf8659cfcb13224cbd47863a34fc58-Paper.pdf.

R. Kaplan, C. Sauer, and A. Sosa. Beating atari with natural language guided reinforcement learning, 2017.

W. B. Knox and P. Stone. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297. IEEE, 2008.

W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

E. Krahmer and K. Van Deemter. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2012.

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

A. D. Laud. Theory and application of reward shaping in reinforcement learning. Technical report, 2004.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1): 1334–1373, 2016.

J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

T. Qiao, J. Dong, and D. Xu. Exploring human-like attention supervision in visual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

G. Qu, C. Yu, S. Low, and A. Wierman. Combining model-based and model-free methods for nonlinear control: A provably convergent policy gradient approach. *arXiv preprint arXiv:2006.07476*, 2020.

V. Ramanishka, A. Das, J. Zhang, and K. Saenko. Top-down visual saliency guided by captions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7206–7215, 2017.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

D. Skinner. Mark e. bouton learning and behavior: A contemporary synthesis. *CANADIAN PSYCHOLOGY*, 48(4):281, 2007.

R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 2011.

Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, Jun 2020. doi: 10.1145/3377930.3389847. URL `http://dx.doi.org/10.1145/3377930.3389847`.

J. Thomason, J. Sinapov, M. Svetlik, P. Stone, and R. J. Mooney. Learning multi-modal grounded linguistic semantics by playing" i spy". In *IJCAI*, pages 3477–3483, 2016.

J. Thomason, J. Sinapov, R. Mooney, and P. Stone. Guiding exploratory behaviors for multi-modal grounding of linguistic descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

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

S. Venugopalan, L. A. Hendricks, R. Mooney, and K. Saenko. Improving lstm-based video description with linguistic knowledge mined from text. *arXiv preprint arXiv:1604.01729*, 2016.

G. Vezzani, A. Gupta, L. Natale, and P. Abbeel. Learning latent state representation for speeding up exploration. *arXiv preprint arXiv:1905.12621*, 2019.

Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with

visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. 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, 2016.

O. Zaidan, J. Eisner, and C. Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, pages 260–267, 2007.

Y. Zhang, I. Marshall, and B. C. Wallace. 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, 2016.

Y. Zhang, J. C. Niebles, and A. Soto. Interpretable visual question answering by visual grounding from attention supervision mining. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 349–357. IEEE, 2019.

Vita

Clara Cecilia Cannon was born in Seattle, Washington on 13 December 1996, the daughter of Lance Victor Cannon and Mariana Genevieve Cannon. She received a Bachelor of Science in Computer Engineering from the University of Idaho in 2019. She began her education at The University of Texas at Austin in Fall 2019 to pursue a Master of Science in Computer Science, focusing on Natural Language Processing and Reinforcement Learning. Clara has participated in internships at Micron Technologies, Boise, ID and Nike, Portland, OR. After graduation, she will begin a full time position as a Data Engineer for McKinsey and Company in Austin, Texas.

Address: claracannon@utexas.edu

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.