Evaluating the Robustness of Natural Language Reward Shaping Models to Spatial Relations

Antony Yun

This document has been approved by my advisor for distribution to my committee.

Advisor:
Ray Mooney

Second Reader: Scott Niekum
Third Reader: Robert van de Geijn

Department of Computer Science
University of Texas at Austin
May 2020

Evaluating the Robustness of Natural Language Reward Shaping Models to Spatial Relations

Antony Yun

May 2020

Abstract

As part of an effort to bridge the gap between using reinforcement learning in simulation and in the real world, we probe whether current reward shaping models are able to encode relational data between objects in the environment. We construct an augmented dataset for controlling a robotic arm in the Meta-World platform to test whether current models are able to discriminate between target objects based on their relations. We found that state of the art models are indeed expressive enough to achieve performance comparable to the gold standard, so this specific experiment did not uncover any obvious shortcomings.

1 Introduction

Reinforcement learning (RL) is the science of learning what actions to take in order to best reach a goal state [Sutton and Barto, 2018]. RL agents are generally trained by interacting with an environment, simulated or real, and receiving feedback on the quality of actions based on the signal they receive from a reward function. The actor uses a policy to evaluate which action to take at each time step, and it learns the optimal policy through trial and error over many episodes. These agents often take advantage of deep learning, which can extract features for use in an RL policy or learn parameters and value estimates of the policies themselves [Francois-Lavet et al., 2018].

In contrast with supervised learning algorithms, the agent does not have access to a labeled dataset over which it can minimize a loss function for a static task. Instead, the agent must reach a goal state by repeatedly interacting with the environment while balancing exploration and exploitation. For particularly robust environments, executing these training episodes can be quite computationally expensive while lacking the embarrassingly parallel nature of many deep learning models. Slow convergence due to large sample complexity is of the more prominent limitations of training an RL agent and is the subject of active research [Kakade et al., 2003].

Training time is influenced by, among many factors, the reward function used. The most basic possible reward is a sparse binary reward

encoding whether or not the goal state was reached in that time step. Initially, the positive reward is only transferred to the states adjacent to the goal, but continued iterations propagate it throughout the state space, subject to a discount factor. Sparse rewards are easy to de ne but provide minimal training signal to the agent. In contrast, a dense reward function provides a stronger learning signal by assigning value to more of the state transitions, and thus training converges faster. However, dense reward functions require either careful tuning by a domain expert, or access to the internals of the environment. These methods can be di cult to set up and are also speci c to the task at hand, generalizing poorly to unseen environments.

We focus on a line of work that aims to bridge this gap by providing robust rewards with minimal expert knowledge. In particular, we leverage reward shaping, in which additional rewards are supplied to the learning agent in order to guide it towards the true goal [Ng et al., 1999]. We believe that natural language can richly encode information about tasks, so we derive additional signal from linguistic annotations of the target tasks. Our multimodal models combine sequences of visual state information with human supplied annotations in order to produce an auxiliary reward.

Prior work by Goyal et al. [2019] introduced LEARN, a model that learns the similarity between prior actions and natural language descriptions within an Atari game. Using the similarity prediction as a shaping reward improved agent performance on the task.

Goyal et al. [2020] then introduced Pix2R, a model that extends LEARN and reward shaping to the Meta-World robot manipulation domain. They showed that applying intermediate rewards derived from language and pixel data yields comparable performance to using an expert-crafted dense reward function. However, the nature of the environments and language collected were very simple{each object present in an environment was unique from the rest, and so we speculated that the linguistic model could simply have learned to classify and detect objects. Real world environments are more complex, as there can often be numerous instances of the same object class, and objects often exist within the context of their surroundings.

As part of an e ort to bridge the gap between using reinforcement learning in simulation and in the real world, we probe whether current reward shaping models are able to encode relational data between objects in the environment. We augment the dataset from Goyal et al. [2020] with environments that contain duplicate objects and with annotations that identify the goal states using more descriptive relational language. A task from the initial dataset that required pressing the button while ignoring the door would now require identifying which of two buttons to press while still avoiding the door. Our goal is to create a challenge dataset that either reinforces the robustness of state of the art models, or induces the development of new architectures that are expressive enough to ground complex language used for RL tasks.

We found that state of the art models were capable of learning the relational scenarios that we generated. In fact, even agents using models trained on the original dataset performed comparably to the dense reward agents on the challenge dataset. While these results do not reveal any

immediate room for improvement in reward shaping model architecture, they pave the road for future work on improving the challenge dataset to further investigate model performance in various conditions.

2 Background

2.1 Neural Networks

Both LEARN and Pix2R use deep neural networks to encode the input features and produce an output prediction. These networks are used to solve supervised learning problems, in which a model tries to optimize an objective function over a labeled training dataset. The network weights are adjusted through gradient descent and backpropagation, which determine the direction in which each parameter should be updated to increase the objective (or reduce a loss) [LeCun et al., 2015].

The most basic neural network building block is the fully-connected layer. Fully-connected layers project vectors from one hidden size to another using matrix multiplication followed by a nonlinear function. Each component of the output vector of a layer is a linear combination of all components of the input vector. Fully-connected networks are sufficient to approximate arbitrary functions [Cybenko, 1989], but other types of units have been introduced to encourage the network to model specific types of behavior.

A convolutional neural network (CNN) uses a fixed size kernel that applies a fully-connected transform on a local region of the input, striding across the entire length of each dimension [LeCun et al., 2015]. In the context of images, a 2-dimensional convolution iterates over each 3x3 region, for example, and produces a single output with multiple channels located at the center of the region. CNNs are especially powerful for image processing, and are the basis for most recent advances in classification, segmentation, and many other tasks.

A recurrent neural network (RNN) encodes temporal information by keeping a hidden state that is preserved and updated across timesteps. For example, the Long Short-Term Memory (LSTM) constructs a *memory cell* out of multiplicative gate units that control updates to two latent states [Hochreiter and Schmidhuber, 1997]. These networks can be unrolled and updated using backpropagation through time. RNN's are commonly used to produce representations of sequences with arbitrary length, such as natural language sentences.

2.2 Markov Decision Process

Reinforcement learning problems are often paramaterized by Markov Decision Processes (MDPs), which are a $\langle S; A; T; R; \gamma \rangle$ tuple that represent an agent interacting with its environment [Sutton and Barto, 2018]. S is the set of states, A is the set of actions, $T : S \times A \times S \to [0; 1]$ is a transition function for the environment that describes the probability of being in state s' after taking action a from state s, $R : S \times A \to \mathbb{R}$ is the reward function that determines the reward received by taking action a

from state s, and $\gamma \in [0, 1]$ is a discount factor that decays the influence of rewards to be obtained far in the future.

An RL agent will act upon a learned policy in order to maximize its expected future reward G_t:

$$G_t = \sum_{t=0}^{T} \gamma^t R_t$$

At each timestep, the agent will decide to take an action based on its policy and its observed state. The environment will transition to a new state sampled from the transition function, and the agent will gain a reward for taking the action.

Goyal et al. [2019] introduce the notion of a language-augmented MDP, or an MDP+L. This adds a natural language instruction L that describes the entire task to obtain the tuple $\langle S, A, T, R, \gamma, L \rangle$.

2.3 Proximal Policy Optimization

Policy gradient methods are a common approach for learning the optimal policy under an MDP. These methods act on a parameterized policy and try to find the optimal values of the parameters by roughly approximating a gradient-ascent process. Each parameter is updated in the direction of an estimate of the gradient of performance with respect to that parameter [Sutton and Barto, 2018].

Proximal Policy Optimization (PPO) is a policy gradient method that uses a clipped surrogate objective function Schulman et al. [2017]. In short, the clipped surrogate objective is an improved method of modeling how well a given policy selects the optimal action that penalizes large policy updates.

The models used in this paper are not specific to any particular algorithm for solving MDPs, so we use PPO due to its popularity and effectiveness.

2.4 Reward Shaping

Ng et al. [1999] introduce the concept of *reward shaping*, which is the process of adding an arbitrary auxiliary reward to an agent's existing reward in order to guide policy training. In other words, you can replace your initial reward R with $R' = R + F$, where F is any real valued function. They prove that R' is guaranteed to converge to the same optimal policy as R as long as F is a *potential-based* function, which must satisfy the following condition for consecutive states s and s' and discount factor γ:

$$F(s, a, s') = \gamma \Phi(s') - \Phi(s)$$

Intuitively, this formulation prevents the agent from repeatedly acting in cycles to gain net positive shaping-reward and exploit the nature of the reward function{returning to the same state does not yield any additional reward when using difference of potentials. We take advantage of this property by predicting a similarity score as the potential function, and then constructing a reward using difference of potentials.

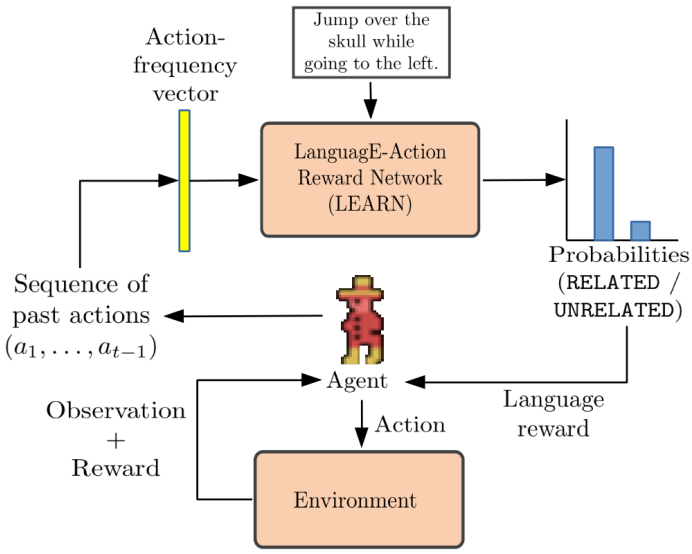


Figure 1: The LanguagE-Action Reward Network (LEARN) uses action frequency vectors and a natural language description to produce a relatedness score. This prediction is used as a shaping reward for a standard RL agent.

2.5 LEARN

The LanguagE Action Reward Network (LEARN) [Goyal et al., 2019] is the first framework to compute shaping rewards based using free-form natural language instructions and action histories. The framework was tested on the Montezuma's Revenge game from the Atari Learning Environment. The agent must decide which discretized action to take, such as moving left or jumping, based on the sequence of past actions and a natural language description of the task. The framework, depicted in Figure 1, consists of a supervised learning phase followed by an RL phase.

LEARN is a neural network that predicts whether a given language and trajectory are related. The trajectory is encoded as an *action frequency vector*, which is a vector whose elements are the frequencies of a particular type of action being taken in the history. Language instructions are encoded using pretrained embeddings and RNNs [LeCun et al., 2015] to produce a final vector. The action frequency vector is passed through fully-connected layers, concatenated with the language vector, and then passed through more fully-connected layers to predict the probability of relatedness.

Agents then learn a policy using a sparse binary reward for reaching the goal that is shaped by the output of the LEARN model. This policy completes the task 60% more often on average than the sparse reward alone, laying the foundation for further research on natural language reward shaping.

2.6 Pix2R

Pix2R extends the LEARN model to use visual input instead of using frequencies of discrete actions [Goyal et al., 2020]. Pix2R is used to manipulate a robot arm to complete tasks in the Meta-World environment. At a high level, the model is mostly similar to the LEARN model, replacing the fully-connected action frequency branch CNN [LeCun et al., 2015] feature extractors followed by an LSTM [LeCun et al., 2015]. We use Pix2R as the baseline model for our work, so the details will be described in further sections.

3 Dataset

3.1 Meta-World

Meta-World is a simulated benchmark platform for RL that supports various robot manipulation tasks [Yu et al., 2019]. Each of these tasks involves moving a Sawyer robot arm to interact with an everyday object, such as a button, door, or co ee machine. These interactions move the object from an initial state to a goal state. Since the platform focuses on creating a benchmark suite for evaluating RL meta-learning techniques, it prioritizes the dynamics of the objects over their appearances, which are approximated by colored shapes. The action space consists of four dimensions - 3D movement of the end-e ector of the robot arm, and a force applied to the gripper. The observation space consists of 3D Cartesian positions of the end-e ector, object, and goal position. The benchmark de nes the dense reward as a sum of multiple component rewards for reaching, grasping, placing, and pushing tasks. Based on the type of task being perfomed, the reward function is either de ned as $R = R_{\text{reach}} + R_{\text{grasp}} + R_{\text{place}}$, or $R = R_{\text{reach}} + R_{\text{push}}$. These component rewards are complex, expert crafted expressions that measure the success of a task based on the geometries of the observations. For example, the reaching reward $R_{\text{reach}} = jjh \quad ojj_2$ measures the distance between the hand and the object.

3.2 Pix2R dataset

Goyal et al. [2020] construct the dataset used for the Pix2R paper by selecting 13 of the 50 Meta-World tasks featuring 9 di erent objects. Example tasks include pressing the button on a co ee maker, turning a door handle counter clockwise, and opening a window. They generated environments for a particular task by placing its relevant object at a random position on the table, and then repeatedly sampling new positions to place one of the remaining objects until a collision occurred. The dataset contained 100 scenarios for each of the tasks, for a total of 1300 scenarios.

The Pix2R architecture uses a neural network to predict a similarity score for a sequence of observations and a linguistic description of the task. Training this model requires pairs of videos and captions of a robot accurately performing the task. These videos are generated by training an agent with access to the dense reward function de ned in the Meta-World paper. Intuitively, this dense reward acts as a sca old to generate

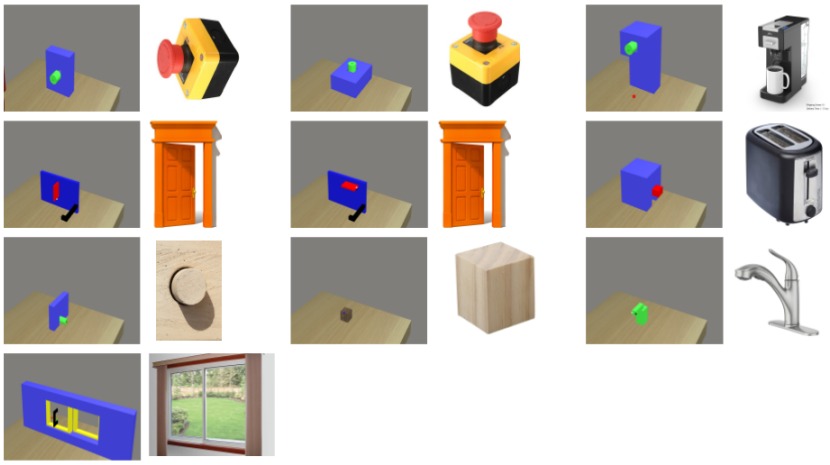


Figure 2: List of objects used by Goyal et al. [2020].

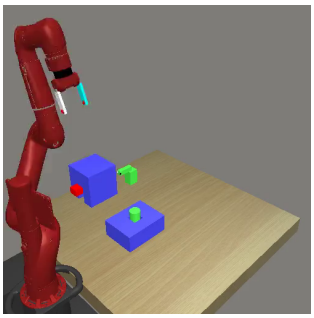
\ground-truth" data for the model to train on and mimic. At evaluation time, the model can learn a policy without access to a dense reward. These techniques can especially bene t domains with an abundance of demonstrations and layperson descriptions but no expert reward function.

The agent learns a policy for each of the scenarios using PPO [Schulman et al., 2017] and then uses that policy to record videos of a successful episode. Three videos are recorded from center, left, and right viewpoints for resilience to occlusions.

The corresponding annotations for these videos were collected through Amazon Mechanical Turk (AMT). The AMT workers were provided with 5 videos from distinct scenarios and asked to provide natural language instructions to the robot for completing each task.

Several methods were used for improving data quality. First, the workers were shown Figure 2, which depicts a mapping of abstract simulated objects to more realistic images. Images were used rather than assigning names to the objects in order to avoid priming the workers to certain words and to collect a more diverse vocabulary. Providing this table helped to some extent, but many annotations still used generic terminology such as `blue box' and `red peg'. Workers were asked to provide an instruction spoken to the robot, rather than a narration describing what the robot did. In order to validate this, workers could only proceed by selecting one of four sample descriptions for a video that used both the correct meaning and the correct voice. Further validation included ltering out short or meaningless descriptions.

Goyal et al. [2020] note that most of the descriptions for the original dataset only referenced the target objects, so training pairs could be generated by pairing a video with any description for that target object. As such, they were able to collect only 520 descriptions for the 1300 sets of videos.



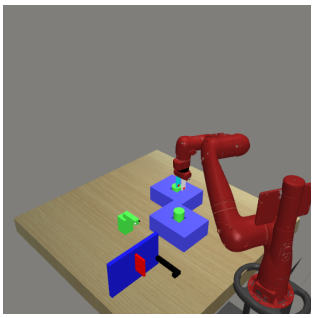


Figure 3: A simulated robot completing a task in the Meta-World domain. Scenarios from Goyal et al. [2020] (left) have at most one object from each class, while ours (right) include two instances of the target object class.

The dataset is split into 79 training scenarios, 18 validation, and 3 testing. The language is split into 5 validation, 3 testing, and the remaining for training. They generated positive examples by randomly pairing a scenario with an annotation within a given task class, and they generated negative examples for training by selecting a description associated with one of the other objects in the scene.

3.3 Our Dataset

The goal of our work was to create a dataset with more complex language, particularly with regards to describing relations between objects. We based our dataset on that used by Pix2R { we started by using the same methodologies and altered them accordingly when necessary. Our dataset is meant to be fully compatible with that of Pix2R, in that it can either be combined with the original, used for ne tuning, or used separately. As such, we maintain the format of *h*trajectory, language*i* pairs, where trajectories are three di erent video perspectives of a simulated environment, and the language is a human annotation.

We modi ed the video collection script to generate scenarios that would encourage more robust descriptions. We rst guaranteed that every environment would have two objects of the target class, as shown in Figure 3. Descriptions in the original dataset were often simple sentences instructing the robot to move towards the target object, so we decided to introduce ambiguity that would force the annotator to use more speci c language.

We generated videos by training PPO until 5 consecutive successes on a speci c scenario for a task for up to 2000 episodes. If the agent was unable to learn a policy, we would regenerate the environment and repeat until success. We noticed that certain tasks trained substantially quicker than others; perhaps the some of the larger or more di cult to manipulate objects interfered with each other, especially since our environments were more crowded than the original ones. We used 5 scenarios for each of the 6 tasks that we were able to train to completion,

for a total of 30 scenarios. These tasks were button_top, button_side, coffee_button, handle_press_top, door_lock, and door_unlock. The button tasks involved pressing a button attached to a box or a co ee machine, handle_press_top involved pressing down on the handle of a toaster, and the door tasks required the agent to rotate a door handle clockwise or counterclockwise.

We used AMT to collect at least 3 annotations for each scenario. We used the same experimental design as the original paper, but we encouraged the annotators to use more robust descriptions. We speci cally asked them to *`please ensure that the instruction you provide uniquely identi es the correct object, for example, by describing it with respect to other objects around it.'* This guideline encourages them to provide relational language while giving them freedom to use the types of relations they deem most appropriate. Even so, we still had to manually lter out responses that gave insu cient information to uniquely identify the target object. Within the remaining data, most workers opted to use simple, absolute, directional descriptions, such as \push the left button", although some did explicitly relate the target object to its neighbors, as seen in \push the button near the door". While more of the latter would make for a much more challenging dataset as intended, we decided that even the former relations would still provide some insight into whether the model was actually reasoning about the world or just detecting object classes.

In contrast to the original dataset, our new annotations were speci c to each particular scenario within a task, and thus could not be reused across the entire task. Instead, we ensured that we had collected at least 3 annotations per video to avoid over tting on a speci c sentence, and then selected a di erent random annotation for a given video during each epoch. Within each task, we used 3 scenarios for training, 1 for validation, and 1 for testing. We collected a total of 92 descriptions for training, 21 for validation, and 18 for testing. We manually selected negative language examples by reviewing similar videos and selecting the annotations that referred to similar collections of objects but with di erent relations.

4 Models

We evaluate the framework from Goyal et al. [2020] on our dataset. This approach uses the Pix2R architecture to maps the pixel representations of states to rewards given the natural language description of the task. The mapping is learned through a supervised learning phase, and the rewards are used in the policy training phase.

4.1 Pix2R: Supervised Learning

4.1.1 Network Architecture

The Pix2R model extends the LEARN model [Goyal et al., 2019] by replacing the action frequency vector with a sequence of frames. Three separate CNNs [LeCun et al., 2015] encode each of the three input perspectives. Each CNN encodes a frame into a xed size lower dimensional

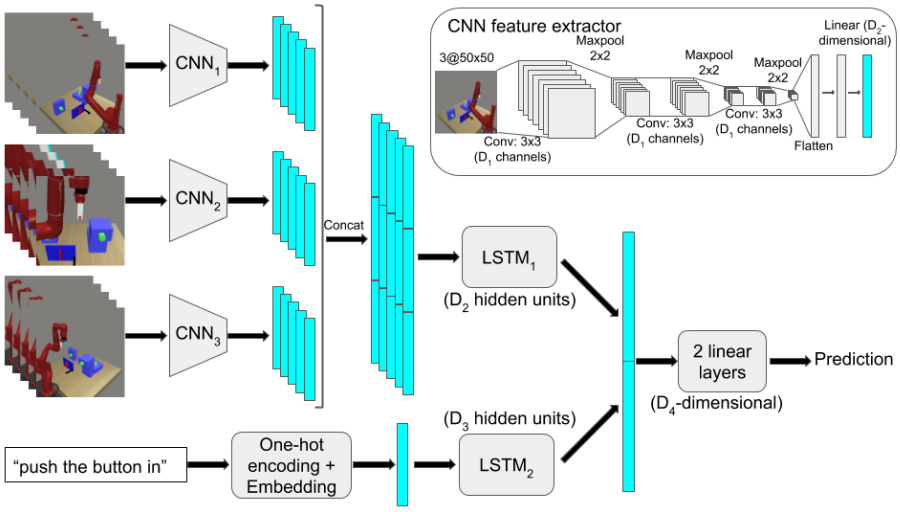


Figure 4: Neural network architecture for Pix2R model. Features are extracted from trajectory images by CNNs and then combined temporally by an LSTM. Natural language descriptions are passed through an embedding layer followed by an LSTM. The resulting vectors from each modality are concatenated and passed through linear layers for regression.

representation, and the results of the three CNNs are concatenated. The encoded vectors across all timesteps are passed through a two-layer LSTM [LeCun et al., 2015] to get an encoding of the entire trajectory. Goyal et al. [2020] also investigated replacing the LSTM with a mean-pooling layer, but found that the LSTM performed better, as it was able to encode crucial temporal information. Thus, we only evaluate the LSTM-based model in this work.

The language description is one-hot encoded and passed through an embedding layer and then a two-layer LSTM. The outputs of both the frame network and the language network are concatenated and passed through two linear layers to output a similarity score, as depicted in Figure 4.

4.1.2 Data Augmentation

Pix2R is trained using frame dropping, which samples a small fraction of frames from the entire trajectory. We retain a given frame within our trajectory with probability 0.1 and drop it with probability 0.9, e ectively creating a subsampled trajectory of about 10% the original length. This strategy accelerates training and increases robustness to minor variations in trajectories. For consistency, 1 out of every 10 frames are sampled when running inference on the Pix2R model during policy training.

During training, we sample one of the frames of the full trajectory uniformly at random and only consider frames up until that point. Us-

Hyperparameter	Values
D_1	{32, 64, 128, 256}
D_2	{64, 128, 256, 512}
D_3	{64, 128, 256, 512}
D_4	{128, 256, 512, 1024}
learning_rate	{1E-3, 1E-4, 1E-5}

Table 1: Set of values of hyperparameters experimented with. See Figure 4 for explanations of parameters D_i's.

ing incomplete trajectories exposes the model to typical inputs it would receive in the middle of policy training.

4.1.3 Training Objective

Goyal et al. [2020] consider two possible training objectives { binary classi- cation of RELATED and UNRELATED input pairs, and regression of a [1; 1] relatedness score. The regression objective produced statistically signi - cantly better results, so we chose to only evaluate the regression objective.

In regression, the output of the network is mapped from real-valued to [1; 1] by the tanh( ) function. For an incomplete trajectory, we use a ground truth score of s $\frac{l}{L}$, where s = 1 for positive examples, s = 0 for negative examples, l is the length of the incomplete trajectory, and L is the length of the complete trajectory. This creates a smooth function that interpolates between +1 for a completed related trajectory and 1 for an completed unrelated trajectory, where the magnitude of the score is proportional to the completeness of the trajectory. The network is trained to minimize mean squared error.

4.1.4 Training and Parameters

The network is trained end-to-end using an Adam optimizer. Network hyperparameters include learning rate, CNN channels, trajectory LSTM hidden units, language LSTM hidden units, and regression layer width. Optimal values are chosen by random search of the values in Table 1 and selecting the combination with the highest Spearman correlation between predicted and ground truth scores on the validation set. Mean squared error is an e ective training objective, but for validation, the Spearman correlation assesses whether there's a monotonic ordering of the variables, which is more important than matching the ground truth exactly.

4.2 RL Policy Training

During policy training, the agent has access to the natural language description of the goal, a sequence of images for the trajectory, and a sparse reward. An intermediate award can be generated through inference with a trained Pix2R model on the language and trajectories, where related input pairs produce a larger intermediate reward. The new reward function is a potential-based shaping reward de ned as $F(s_t)$ = (s_t) $(s_{t\ 1})$.

The potential function (s_t) is simply the output of the Pix2R model on state s_t. The use of a potential-based shaping reward ensures that the model only changes the rate at which it trains without a ecting the optimal policy.

5 Results and Discussion

We used the test split of the challenge dataset to evaluate the performance of various RL agents. Our goal was to determine whether the relational data was more challenging than the original data and to identify any room for improvement. We considered the following baseline and Pix2R agents trained on various subsets of the data in order to identify these potential performance gaps and to attempt to attribute them:

Sparse: PPO with binary goal reward.

Dense: PPO with dense Meta-World reward.

Original: PPO using Pix2R trained on original dataset.

Augmented: PPO using Pix2R trained on augmented dataset.

Reduced: PPO using Pix2R trained on reduced dataset.

We included the sparse, dense, and original models as baselines from Goyal et al. [2020]. The augmented model is trained from scratch on the union of the original and newly collected data. We noticed that the original dataset already contained several relational descriptions, so we also trained a model on a reduced dataset, which we created by excluding descriptions with `left' or `right' from the original dataset.

We rst train each supervised model using 8 di erent combinations of hyperparameters and select the model with the best validation score.

For a given trial, we use the same experimental setup as Goyal et al. [2020]. We run policy training for 500,000 timesteps, restricting each episode to 500 timesteps, and record the number of successful episodes every 1000 timesteps. We used 6 videos for testing{1 from each task{with 3 descriptions per video, for a total of 30 members of the test set.

For each member of the test set, we ran 5 trials with di erent random seeds. This produced a total of 90 trials for each model. We report the average number of successes across all trials at regular intervals in Figure 5. While the sparse model performed noticeably worse than all of the others, the dense model and all Pix2R models followed similar training curves. In fact, the augmented model performed the worst of the four, whereas the reduced model performed the best.

When we designed this new dataset, we expected for it to be challenging enough to where no model based on Pix2R would be able to perform as well as dense. Furthermore, we expected a model trained on the augmented dataset, or a vanilla model ne-tuned on the challenge data, to perform better than a vanilla Pix2R model. However, the results showed otherwise. We hypothesized that the small fraction of directional words in the original dataset was su cient to learn relations between objects, which motivated us to create the reduced model. Since this model also

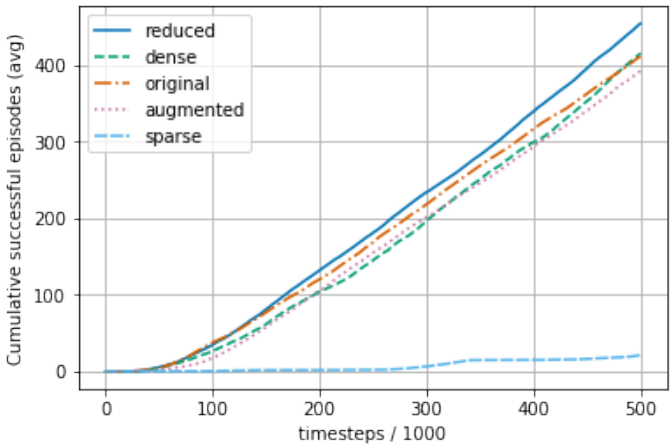


Figure 5: Policy training curves for baselines and Pix2R models. For each model type, the cumulative number of successes is recorded every 1000 timesteps and averaged across all 90 trials. All Pix2R reward agents perform comparably to the dense reward agents, while the sparse reward agents complete substantially fewer episodes.

performed equally as well, we are less con dent in this explanation of our results.

We speculate that the nature of the video generation process could have skewed the scenarios towards the easier side. If the dense reward PPO did not succeed 5 consecutive times by 2000 episodes, we abandoned the environment and generated a new one. We also experimented with an early stopping scheme that would recreate the environment after 1000 consecutive failures. The original Meta-World reward is designed for single-object environments, and thus some of its components greedily use the distance to the goal object without accounting for obstacles. An agent using this reward would have di culties succeeding in relational environments, such as those where the target object was obstructed by other objects or in a crowded neighborhood. If the scenarios skew towards having the goal placed in the front of the scene and far from other objects, then there is little advantage to using descriptive annotations. In order to test this hypothesis, we would need to devise a new scheme to generate videos that avoid this bias.

We also noticed quality issues with the descriptions that we collected from AMT. Many had to be ltered out for not uniquely identifying the target object from its duplicate, and even the remaining ones were grammatically and descriptively simplistic. We could remedy this by devising a two-stage AMT task. In the second stage, we would present a worker with another worker's description and the initial frames of the video, and we would only accept the description if the second worker is able to correctly identify the target object. Another improvement to the data collection process would be to have each worker provide a description for the both

target object and as well as one of the other objects, which would create negative examples without our manual intervention.

6 Conclusion

This paper proposes a challenge dataset that builds upon previous work on incorporating natural language into reinforcement learning reward functions. We constructed a robot arm manipulation dataset pairing demonstrations of complicated scenarios with relational language instructions. We used this dataset to evaluate the robustness of the Pix2R model, which produces an auxillary RL reward based on a visual trajectory and a natural language description of the task. We found that, even when removing the instructions using relational language from the existing dataset, Pix2R models trained on the original dataset performed on par with models that used a dense expert reward, and even Pix2R models trained on the augmented dataset. While we did not find evidence of any shortcomings in the Pix2R architecture, there is room for further experimentation.

7 Future Work

In this work, we produced an initial version of a relational language database, but there is still room for improvement of our methodology. We can improve the quality of the language data by running multi-stage AMT pipelines that are more carefully designed to filter out malformed and ambiguous descriptions. Additionally, we can revisit our environment generation setup to produce more diverse object placement.

Our work sets the stage for creating additional target datasets to probe the effectiveness of the Pix2R model, as well as other grounded language models or neural networks in general. Ultimately, our goal is to expose flaws in current models to inspire more expressive models, such as using joint visual-language attention within Meta-World tasks.

8 Acknowledgements

I would like to thank my faculty advisor, Ray Mooney, for his guidance throughout this process. He introduced me into the research community and provided me with all of the resources and direction that I needed to succeed.

I also greatly appreciate the help of Prasoon Goyal, who also advised me throughout this project. He provided extensive assistance with the LEARN and Pix2R codebases, as well as invaluable feedback on all of my ideas, implementation, and results.

References

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303{314, 1989.

Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*, 2019.

Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Pix2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. In *Under Submission*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.