Pix2R: Guiding Reinforcement Learning using Natural Language by Mapping Pixels to Rewards

Prasoon Goyal [1] **Scott Niekum** [1] **Raymond J. Mooney** [1]

Abstract

Reinforcement learning (RL), particularly in sparse reward settings, often requires prohibitively large numbers of interactions with the environment, thereby limiting its applicability to complex problems. To address this, several prior approaches have used natural language to guide the agent's exploration. However, these approaches typically operate on structured representations of the environment, and/or assume some structure in the natural language commands. In this work, we propose a model that directly maps pixels to rewards, given a free-form natural language description of the task, which can then be used for policy training. Our experiments on the Meta-World robot manipulation domain show that language-based rewards significantly improve learning. Further, we analyze the resulting framework using multiple ablation experiments to better understand the nature of these improvements.

1. Introduction

Reinforcement learning (RL) problems often involve a trade-off between the ease of designing a reward function and the ease of learning from this reward. At one end of the spectrum, a sparse reward function – e.g. a fixed positive reward for completing the task, and zero in all other states – is easy to design, but does not give the learning agent any learning signal until it reaches the goal. As such, the agent requires considerable exploration before any learning can take place. At the other end of the spectrum, a dense reward function – e.g. distance to the next waypoint – can be specified to provide the agent with a stronger learning signal, but is often harder to design and tune compared to sparse reward functions. To bridge this gap, several methods have been proposed, which involve guiding an agent using natural language commands.

[1]Department of Computer Science, The University of Texas at Austin.

However, these techniques are still quite restrictive, often requiring object properties to be predefined (MacGlashan et al., 2014; Williams et al., 2017) and/or assuming some structure in the natural language commands (Bahdanau et al., 2018), which is challenging to scale. In this work, we propose a framework that makes no such assumptions, and directly learns to map pixels to rewards given a free-form natural language description of the task. Our model is based on the framework proposed by Goyal et al. (2019), which consists of two phases – (1) a supervised learning phase that takes in paired (trajectory, language) data and learns a model of relatedness between a trajectory and a language command, and (2) a policy training phase with standard RL setup with an additional linguistic description of the task, wherein the relatedness model is used to generate intermediate rewards using the currently executed trajectory and the description of the task. A significant limitation of this work is that the relatedness model only uses the frequency with which each action is executed in the trajectory. As such, information in the language instructions that requires knowledge of the state (e.g. how to interact with objects) is not helpful.

For instance, consider the domain shown in Figure 1, which is adapted from the recently released Meta-World benchmark (Yu et al., 2019). The scene consists of a robot interacting with an object in the presence of zero or more other objects. Different scenarios can be created in the domain by randomizing the set of objects in the scene and their positions. Since linguistic descriptions of such tasks would typically be in terms of the object to be interacted with, whose positions could change across different scenarios, learning a relatedness model between actions and language without taking into account the state (i.e. the image) will not generalize across scenarios. Thus, we extend prior work to learn a relatedness model between sequences of states and language descriptions, and show that generating language-based scores from the resulting model improves the efficiency of policy training on unseen scenarios.

Using the sequence of states directly instead of the frequency of actions poses several challenges. First, unlike action frequencies, which are low-dimensional and discard most redundant information in the trajectories, sequences

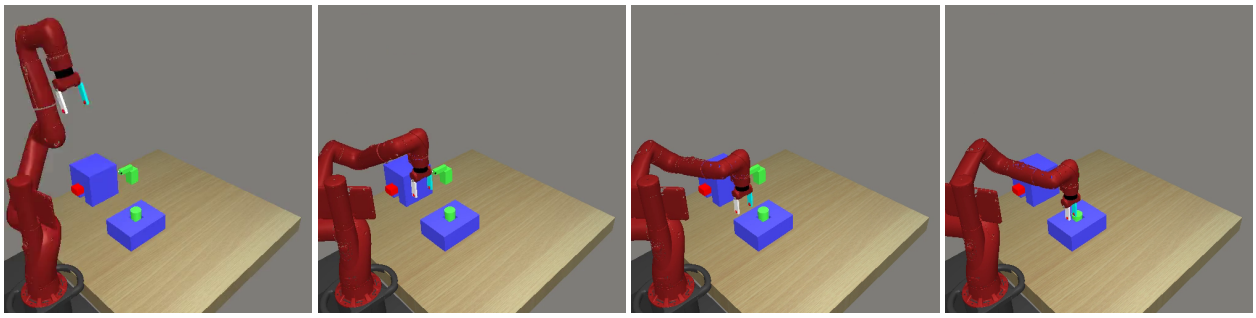

Figure 1. A simulated robot completing a task in the Meta-World domain

of states are high dimensional, and contain a lot of non-discriminatory information, making them prone to overfitting. Thus, a more careful data preprocessing might be required (see Section 5.2 for details). Another issue with using states is that a single viewpoint might make learning harder due to perceptual aliasing and occlusions. Finally, for the relatedness model to be effective during policy training, it must work with incomplete trajectories. While action frequencies extracted from partial trajectories are similar to those extracted from complete trajectories (particularly when the action space is small, as in prior work), the sequence of states for partial trajectories might be much harder to classify compared to the full sequence of states (for example, because the object being manipulated is only interacted with towards the end of the trajectory).

We modify the supervised training approach from prior work to address these challenges. Further, we analyze the resulting model along several dimensions, such as (1) the effect of encoding the temporal information, both in the sequence of states and the natural language command, (2) the effect of using a single viewpoint as opposed to multiple viewpoints, and (3) the influence of different words in a natural language command in the prediction of the relatedness model.

Our experiments highlight several useful properties of our approach. First, as mentioned above, our approach learns an association between language and trajectories in the environment purely from data, without any assumptions about the structure of the environment or the language. Nor does it require hand-engineering of features, which is difficult to scale as the number of objects and the variation in linguistic descriptions grow. Additionally, the relatedness model is agnostic to the end task on which policy training is performed. As such, the supervised training phase is required only once for a given domain, and the resulting model can then be used on any downstream policy training task. Finally, since we generate rewards from the relatedness model for policy training, our approach is compatible with any choice of RL algorithm.

2. Related Work

A number of prior approaches have been proposed to use language to guide a learning agent.

Some approaches involve mapping natural language instructions directly to an action sequence to be executed. Tellex et al. (2011) dynamically instantiate a graphical model given a language command, from which a plan for the agent is inferred. Sung et al. (2018) learn a neural network to predict relatedness between ⟨trajectory, language⟩ pairs and ⟨trajectory, point cloud⟩ pairs, which is then used to find the most likely trajectory given a new language and point cloud. Our approach is different from these approaches in that we use language to generate a reward for the current state, that can then be used to learn a policy using standard RL, which is a more general setting that does not require knowledge of the environment dynamics, and can also work in more complex environments because of the policy learning phase.

Several prior approaches map natural language to a reward function. MacGlashan et al. (2014) learn the conditional distribution of language commands given a task specification. Bayesian inference is then used to find the most likely task given a new command. Arumugam et al. (2017) propose using language to generate rewards at multiple levels of abstraction, by directly learning a conditional distribution of the level of abstraction and the reward function given a command. Williams et al. (2017) define a semantic representation to specify reward functions, and learn a parser to map natural language to this semantic representation. All these approaches assume a specific structure of the reward functions, while our approach does not make any such assumptions.

A number of approaches use a fixed set of linguistic instructions to guide the learning agent. Kuhlmann et al. (2004) generate rules in a custom language from a set of natural language instructions. For a new state, applicable rules are determined and the Q-value of the corresponding state-action pairs is modified. Branavan et al. (2012b) use a game manual to speed up learning, wherein the most rel-

evant sentence from the manual is found for the current state using a log-linear model, and features are extracted from the sentence to augment the state representation. Since the setting here involves working with a predefined set of instructions, these approaches use hand-designed features to find the most relevant instruction to follow at each state.

Kaplan et al. (2017) and Waytowich et al. (2019) learn a neural network that predicts the similarity between a natural language instruction and a state, and use that to follow a fixed sequence of natural language commands. These prior approaches hand-design features to create labeled data between states and each language description, whereas we propose to learn the association between language and trajectories from a small set of human-provided descriptions.

Some approaches learn to ground language while interacting with the environment. Branavan et al. (2012a) extract pairs of states that satisfy the precondition relation from text using a log-linear model, and use that to generate a sequence of subgoals for a given task. The log-linear model is trained jointly with the policy for the end task. Misra et al. (2018) learn a policy that directly maps state and language to actions using reinforcement learning. Bahdanau et al. (2018) learn a language-conditioned reward model using an adversarial learning framework, which is trained to discriminate between ground truth goal states for the given instruction and those that are generated by the policy. Our approach involves a separate supervised learning phase to ground language, which does not require interacting with the environment.

Fu et al. (2019) learn a language-conditioned reward function, but require knowledge of environment dynamics to compute the optimal policy during training. Narasimhan et al. (2015) use natural language to transfer dynamics across environments. Blukis et al. (2019) generate a state visitation distribution given a natural language instruction, which is then used to generate rewards for policy training.

3. Background

3.1. Markov Decision Process

Reinforcement learning consists of an agent interacting with an environment. The learning problem is typically represented using a Markov Decision Process (MDP) $M = \langle S, A, T, R, \gamma \rangle$. Here, S is the set of all states in the environment, A is the set of actions available to the agent, $T : S \times A \times S \rightarrow [0, 1]$ is the transition function of the environment, $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor.

At timestep t, the agent observes a state $s_t \in S$, and takes an action $a_t \in A$, according to some policy $\pi : S \times A \rightarrow [0, 1]$. The environment transitions to a new state $s_{t+1} \sim T(s_t, a_t, \cdot)$, and the agent receives a reward $R_t = R(s_t, a_t)$.

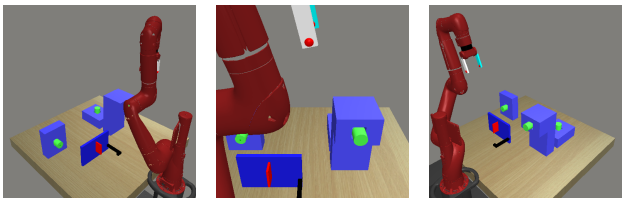

Figure 2. Viewpoints used for data collection and experiments.

The goal is to learn a policy π, such that the expected future return, G_t, defined as follows, is maximized:

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

In this work, we use an extension of the standard MDP, defined as $M' = \langle S, A, T, R, \gamma, L \rangle$, where L is an instruction describing the task using natural language, and the other quantities are as defined above. We denote this modified MDP as MDP+L (introduced by Goyal et al. (2019)).

3.2. Prior Work: LEARN

Goyal et al. (2019) proposed a framework for learning in an MDP+L, which consists of the following two phases.

Phase 1: A neural network – the LanguagE Action Reward Network (LEARN) – is trained to predict whether a given trajectory and language are related or not. This requires paired ⟨trajectory, language⟩ data in the environment. As a preprocessing step, each trajectory is first encoded into an *action frequency vector*, which is a vector of size $|A|$, with the i^{th} component proportional to the number of times action i appears in the trajectory. The neural network takes this action frequency vector and language as inputs to predict the relatedness between the trajectory and language, which is modelled as a binary classification problem.

Phase 2: Next, a policy is trained for a new task in the MDP+L setting – the extrinsic reward from the environment is assumed to be sparse (i.e. 1 if the agent successfully completes the task, and 0 otherwise), and the agent additionally gets a language command describing the task. At every step, an action frequency vector is created from the sequence of past actions and passed to the pretrained LEARN model along with the given command. The LEARN model generates probabilities over classes `RELATED` and `UNRELATED`, which are used to generate intermediate rewards for reward shaping (Ng et al., 1999).

4. Approach

In order to apply the framework described in Section 3.2 to domains where using the state information is crucial to

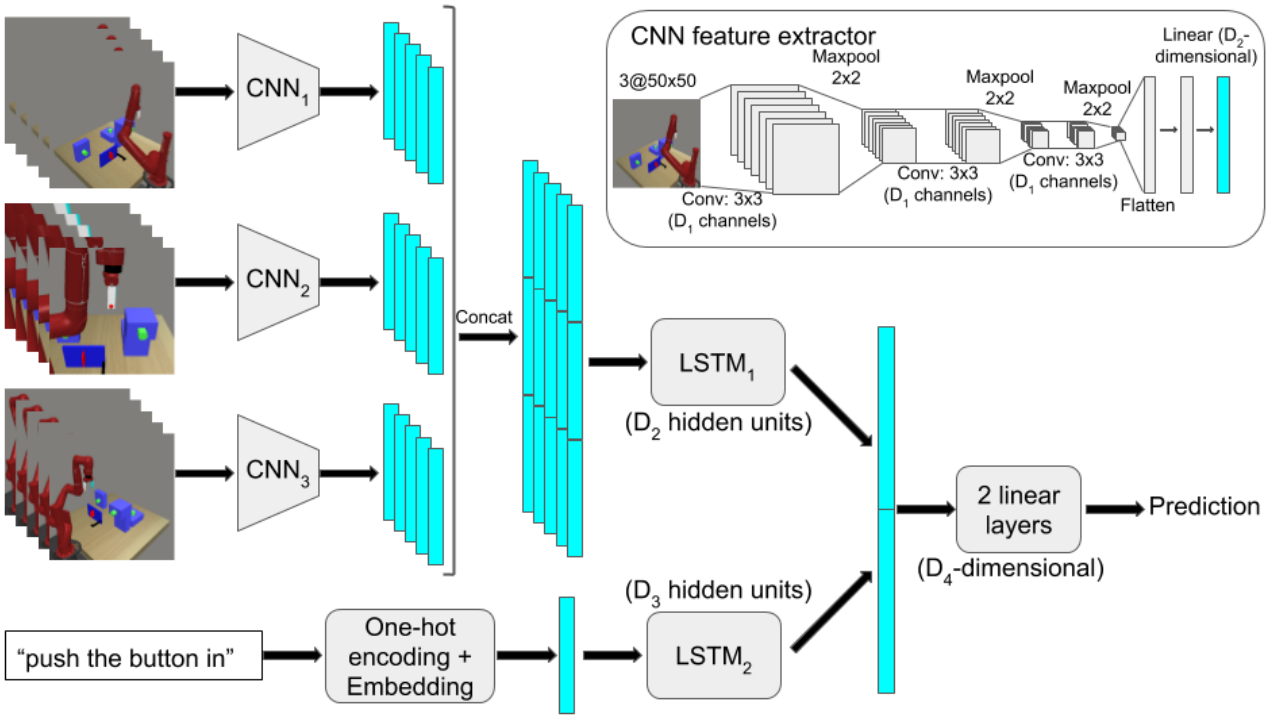

Figure 3. Neural network architecture: The sequence of frames from the three viewpoints are passed through three separate CNN feature extractors. The resulting feature vectors are concatenated across views. The sequence is then passed through an LSTM to obtain an encoding of the trajectory. The given linguistic description is converted to one-hot representation, and passed through an embedding layer, followed by an LSTM. The outputs of the two LSTMs is concatenated and passed through a sequence of 2 linear layers (with a ReLU activation between them) to generate the final prediction.

understanding language (such as the domain in Figure 1), we propose Pix2R, which takes in the pixel representations of the states and maps them to rewards given the natural language description of the task. The framework consists of a supervised learning phase and a policy training phase as before, but with modifications as described below.

4.1. Pix2R: Pixels to Reward

First, a relatedness model between a trajectory and a language is learned given paired data. Our model is based on that proposed by Goyal et al. (2019); however, instead of representing a trajectory using an action frequency vector before feeding into the neural network, we feed the sequence of frames in the trajectory directly into the neural network.

4.1.1. Network Architecture

Using sequence of frames instead of an action frequency vector requires addressing perceptual aliasing and occlusion. Thus, our network architecture is designed to take multiple views as inputs. Specifically, we use three different viewpoints, as shown in Figure 2. In our ablation experiments, we compare the model described here with a model that takes a single viewpoint as input.

An independent CNN is used for encoding the sequence of frames from each viewpoint to generate a fixed size representation for each frame. These sequence of vectors are concatenated across the views to generate a single sequence of fixed size vectors, which is then passed through a two-layer LSTM to get an encoding of the entire trajectory. In our ablation experiments, we use mean-pooing (as an alternate to LSTM) to combine the vectors across timesteps, which is analogous to the action frequency vector used in prior work that ignores temporal information in the sequence. Additionally, we also experiment with just using the last frame instead of the entire sequence of frames in our ablation experiments.

The language description is converted to a one-hot representation, and passed through an embedding layer, followed by a two-layer LSTM. The outputs of the LSTMs encoding the trajectory and the language are then concatenated, and passed through a sequence of fully-connected layers to generate a relatedness score. As for trajectories, we replace LSTM with mean-pooling to encode the language in our ablation experiments. A diagram of the network is shown in Figure 3.

4.1.2. Data Augmentation

Frame dropping. After sampling a trajectory, each frame is independently selected with a probability 0.1, and dropped with a probability 0.9. The resulting sequence of frames is passed through the network. This makes the training faster by reducing the input size, as well as making the network robust to minor variations in trajectories. During policy training, the trajectories are subsampled to keep 1 frame in every 10.

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

4.1.3. Training Objectives

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

As mentioned above, we train the model with trajectories of different lengths to better match the distribution of trajectories during policy training. However, smaller trajectories might sometimes be hard to classify as related or unrelated to the description, since it requires extrapolating the complete path the agent will follow. Our preliminary experiments suggest that these harder to classify examples affect learning – on unseen complete trajectories, a model trained with complete trajectories has a lower error, compared to a model trained with incomplete trajectories (which also include complete and nearly complete trajectories). This motivated us to experiment with a regression setting, instead of the classification setting, as described below.

Regression. In this setting, the model predicts a single relatedness score between the given trajectory and language, which is mapped to $[-1, 1]$ using the $\tanh()$ function. The ground truth score is defined as $s \cdot \frac{l}{L}$, where $s = 1$ for positive and $s = -1$ for negative examples, l is the length of the incomplete trajectory and L is the length of the complete trajectory as described above. Thus, given a description, a complete related trajectory has a ground truth score of 1, while a complete unrelated trajectory has a score of -1. Shorter trajectories smoothly interpolate between these values, with very small trajectories having a score close to 0. The network is trained to minimize the mean squared error.

4.1.4. Training Details

The network is trained end-to-end using an Adam optimizer (Kingma & Ba, 2014). We started by tuning the learning rate on a few different architectures – of the 3 values we tried (1E-3, 1E-4, 1E-5), we found 1E-4 to work the best. For the network architecture, we had 4 hyperparameters – D_1, D_2, D_3, D_4 – as shown in Figure 3. For each of these hyperparameters, we searched over the following values – $\{64, 96, 128, 192, 256, 384, 512\}$. We experimented with 8 different combinations of values for the hyperparameters using random search, and selected the model with the best performance on the validation set.

4.2. Policy Training Phase

Having learned a Pix2R model as described above, the relatedness scores from the model can be used to generate language-based intermediate rewards during policy training on new scenarios. During policy training, the agent receives a natural language description of the goal, in addition to the sparse reward. The Pix2R model is used to score trajectories executed by the agent against the given natural language description, to generate intermediate rewards. We used potential-based shaping rewards (Ng et al., 1999), which are of the form $F(s_t) = \gamma \cdot \phi(s_t) - \phi(s_{t-1})$, where s_t is the state at timestep t and $\phi : S \rightarrow \mathbb{R}$ is a potential function. In our case, s_t is the sequence of states encountered by the agent up to timestep t in the current episode. Ng et al. (1999) and Grzes (2017) show that potential-based shaping rewards do not change the optimal policy, that is, the optimal policies under the original reward function R and the new reward function $R + F$ are identical.

For the classification setting, we used the potential function $\phi(s_t) = p_R(s_t) - p_U(s_t)$, as defined by Goyal et al. (2019). Here, p_R and p_U are the probabilities assigned to the classes `RELATED` and `UNRELATED` respectively. For the regression setting, the relatedness score predicted by the model is directly used as the potential for the state. Note that for both the settings, the potential of any state lies in $[-1, 1]$.

5. Domain and Dataset

5.1. Description of the Domain

We use Meta-World (Yu et al., 2019), a recently proposed benchmark for meta-reinforcement learning, which consists of a simulated Sawyer robot and everyday objects such as a faucet, windows, coffee machine, etc. Tasks in this domain involve the robot interacting with these objects, such as turning the faucet clockwise, opening the window, pressing the button on the coffee machine, etc. Completing these tasks requires learning a policy for continuous control in a 4-dimensional space (3 dimensions for the end-effector position, and the fourth dimension for the force on the gripper).

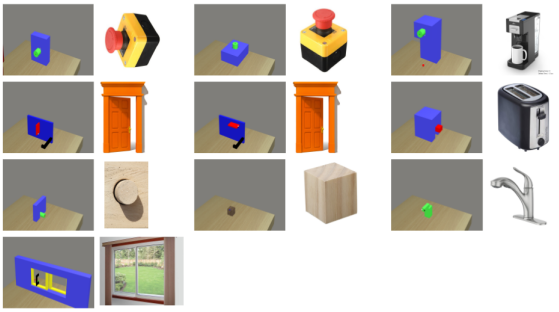

Figure 4. List of objects used

While the original task suite consists of only one object in every task, we create new environments which contain one or more objects in the scene, and the robot needs to interact with a pre-selected object amongst those. In a sparse reward setting, the agent is given a non-zero reward only on successfully interacting with the pre-selected object. In the absence of any other learning signal, the agent might have to learn to approach and interact with multiple objects in the scene in order to figure out the correct object. Using natural language to describe the task in addition to the sparse reward helps alleviate this issue.

5.2. Data Collection

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

A policy was trained for each of these scenarios independently using PPO (Schulman et al., 2017), which was then used to generate one video of the robot completing the task in the scenario. For this purpose, we used the dense rewards defined in the original Meta-World benchmark for various tasks. The median length of trajectories across all generated videos is 131 frames. Note that our algorithm does not need the policies used to generate the videos, so they could also be collected using human demonstrations.

To collect English descriptions of these tasks, Amazon Mechanical Turk (AMT) was used . Since the models of the objects in the environment are coarse, it is usually non-trivial to recognize the real-world objects they represent from the models alone. To guide the AMT workers to use the names of real-world objects the models represent, we showed a table of the models with prototypical images of real-world objects that closely match the models (shown in Figure 4). This enabled us to get descriptions that use the real-world object names, without priming the workers with specific words. [1]

Table 1. Examples of descriptions collected using AMT.

Object Id	Description
0	Press the button.
0	Pressing the button
1	Push peg in to hole.
1	Push the green button.
2	Turn on the coffee maker
2	push in the green button
3	Push toaster handle down
3	Push down the red block.
4	pressing down the object
4	pull down the red switch
5	move the plate down
5	push down the slider
6	Close the door
6	Open the door.
7	twisting the cube
7	rotate the object
8	Rotate the lever anticlockwise
8	Turn the faucet to the right.
9	rotating the object
9	turn on the faucet
10	Open the window.
10	Open the yellow window.
11	Slide the window to the left.
11	Close the Window.
12	pull out the green block
12	Pull out the green piece

Since using a single viewpoint is susceptible to perceptual aliasing and occlusion, we used 3 viewpoints, as shown in Figure 2.

The workers were first provided with the instructions and an example trajectory with a possible description. They were then shown a video and were given 4 possible descriptions to choose from. Only workers that passed this basic test were used to generate descriptions for the main tasks. [2] Each worker was asked to provide descriptions for 5 videos, which were sampled from the 1300 scenarios with the constraint that no two videos in the selected videos belong to the same task. We used simple heuristics (such as number of words and characters in the descriptions) to automatically filter out clearly bad descriptions. Some examples of descriptions (after filtering) are shown in Table 1.

[1] Despite using this technique, we still got some responses where people described the models directly instead of using the object names, e.g. "Pull the red box out slightly in blue square." instead of using the word *toaster*.

[2] The objects used for the example and the test are different from those used in the main tasks.

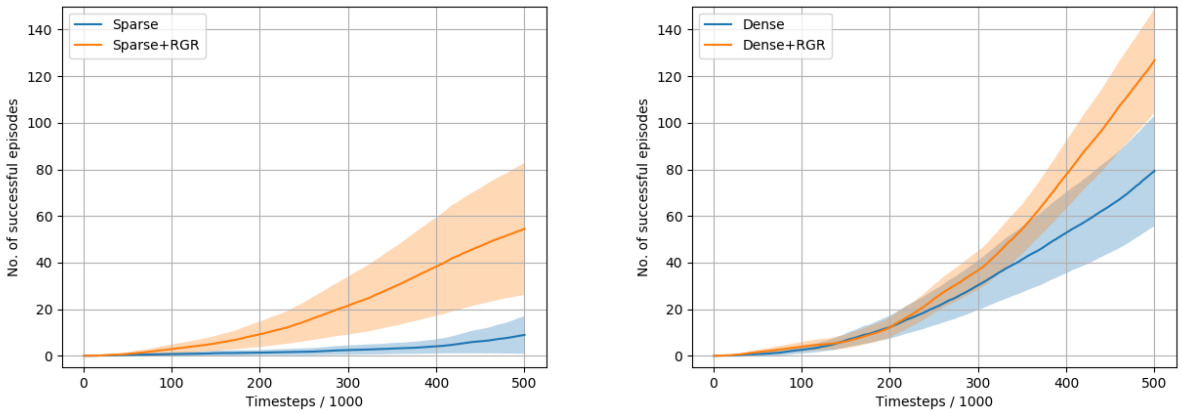

Figure 5. A comparison of policy training curves for different reward models. The shaded regions denote 95% confidence intervals.

Interestingly, most of the descriptions involve only the object being manipulated, with no reference to other objects in the scene. As such, a description collected for one scenario for a task can be paired with any of the 100 scenarios for the corresponding task. Therefore, we collected a total of 520 descriptions, which gives us 40 descriptions per task on average.

For each task, 79 scenarios were used for training, 18 for validation, and 3 for testing. Similarly, the descriptions for each task were split as follows – 5 for validation, 3 for testing, and the remaining for training (since there could be variable number of descriptions per task).

Given pairs of related ⟨trajectory, language⟩, positive examples are generated by pairing a scenario for one of the 13 tasks with a randomly sampled description of the corresponding task. To generate negative examples, if a scenario contains more than one object, then it is paired with the description of the task corresponding to one of the alternate objects in the scene; if there is only one object in the scene, then it is paired with the description of any of the remaining 12 tasks. Using such a scheme for generating negative examples is important because naively creating pairs of trajectories with descriptions of any other task randomly might result in most negative examples such that the task-relevant object mentioned in the description is not present in the scene. As such, the network might learn to use the *presence* of the mentioned object to compute relatedness, instead of whether the mentioned object is being *interacted with*.

6. Experiments

6.1. Policy Training with Language-based Rewards

To empirically evaluate the effectiveness of Pix2R, the following setup was used. For each of the 13 tasks, a policy was trained for the 3 test scenarios using the PPO algorithm. Each policy training was run for 500,000 timesteps, and the number of successful completions of the task were recorded. The maximum episode length was restricted to 500 timesteps. The robot's end-effector was set to a random position within a predefined region at the beginning of each episode.

First, policy training was run with 15 random seeds, both in the sparse reward setting (`Sparse`; 1 if the agent reaches the goal, and 0 otherwise) and the hand-designed dense reward setting (`Dense`; defined in the original Meta-World benchmark). Then, a Kruskal-Wallis test was used for each scenario to identify scenarios where there was a statistical significant difference between the number of successful episodes with sparse rewards and with dense rewards, and the mean successful episodes with dense rewards was higher than the mean successful episodes with sparse rewards. All subsequent comparisons were done on the 16 (out of 39) scenarios for which this was true. Intuitively, these 16 tasks are too difficult to learn from sparse rewards, while they can be learned using dense rewards. Therefore, language-based dense rewards should be useful on these tasks. The remaining tasks are presumably either too simple that they can be learned with sparse rewards alone, or are too difficult to learn within 500,000 timesteps even with hand-designed dense rewards.

Then, for each of the 16 selected scenarios, a policy was trained with language-based rewards using the regression setting, in addition to the sparse rewards (`Sparse+RGR`). For each scenario, 5 policies were trained with different seeds for each of the 3 test descriptions, resulting in a total of 15 policy training runs per scenario.

A comparison of policy training curves for `Sparse` and `Sparse+RGR` rewards is shown in Figure 5 (left). Each

curve is obtained by averaging over all runs (16 scenarios $\times$ 15 runs per scenario) for that reward type. The results verify that language-based rewards result in higher performance on average than sparse ones.

Next, language-based rewards were used in addition to hand-designed rewards using a similar methodology, and the corresponding learning curves for `Dense` and `Dense+RGR` are shown in Figure 5 (right). Interestingly, we find that using language-based rewards in conjunction with hand-designed rewards result in an improvement even over hand-designed rewards.

Further, the statistical significance was computed to compare the reward functions. For each type of reward, first the average number of successful episodes was computed across all the 15 runs for each scenario, giving 16 mean successful episode scores per reward type. Since the number of successful episodes across different scenarios vary quite a bit, the mean scores for each scenario were scaled to be at most 1, by dividing by the maximum value of the mean score across all reward types for that scenario (including the reward types used in ablation experiments described in Section 6.3).

A Wilcoxon signed-rank test was then performed between the sets of normalized scores across reward types. `Sparse+RGR` was found to be statistically significantly better than `Sparse` (p-value=0.007) and `Dense+RGR` was found to be statistically significantly better than `Dense` (p-value=0.034) rewards, at a 5% significance level . Thus, the proposed approach can be used to make policy training more efficient in both sparse and dense reward settings.

6.2. Word-level Analysis

In order to understand how the supervised learning phase is using different words in the description, the supervised model was used to make predictions on the test set, and the gradient of the loss was computed with respect to the continuous representation of the words in the descriptions (i.e. after the embedding layer). The mean of the absolute values of these gradients is then a measure of how much the prediction is affected by the corresponding word. The values are reported in Table 2, which were scaled so that the maximum value for any description is 1.

First, we observe that for all the descriptions, the words describing the main object have a very high average gradient magnitude – *green* and *button* in description 1, *red* and *block* in description 2, *lever* and *toaster* in description 3, *faucet* in description 4, *green* and *lever* in description 5, and *window* in description 6. Several verbs also have a high average gradient magnitude – *turn on* in description 4 and *open* in window. Verbs in other descriptions do not have a high gradient magnitude because for those descriptions, the object affords only one possible interaction, thus making the verb less discriminatory. For the objects *faucet* and *window*, there are two possible actions each (*turning the faucet on or off* and *opening or closing the window*); thus the verb also carries useful information for these objects.

Table 2. Average magnitude of gradients for different words in a description for the relatedness score prediction.

	Descriptions **Average magnitude of gradient for each word**						
1.	push	the	green	button			
	0.53	0.30	1.00	0.94			
2.	push	down	the	red	block		
	0.42	0.57	0.34	1.00	0.91		
3.	pull	down	the	lever	on	the	toaster
	0.16	0.31	0.15	0.75	0.58	0.36	1.00
4.	turn	on	the	faucet			
	0.94	1.00	0.44	0.87			
5.	slide	the	green	lever	to	the	left
	0.52	0.23	0.94	1.00	0.77	0.30	0.78
6.	open	the	window				
	0.83	0.32	1.00				

This analysis suggests that the model learns to identify the most salient words in the description that are useful to predict the relatedness between a trajectory and language.

6.3. Ablations

Having established that policy training works better with the language-based rewards, we ran some ablation experiments as described below. All the ablation experiments were performed with language-based rewards added to dense rewards, since most applications of RL currently use dense hand-designed rewards (which could be suboptimal for complex tasks), and it would be informative to learn which design decisions are most important to get an improvement by using language-based rewards in such settings.

- `LastFrame`: Instead of using the sequence of frames in the trajectory, only the last frame of the trajectory was used, both for training the Pix2R model, as well as for policy training.
- `MeanpoolLang`: The LSTM used to encode the language was replaced with the mean-pooling operation.
- `MeanpoolTraj`: The LSTM used to encode the sequence of encoded frames was replaced with the mean-pooling operation.
- `SingleView`: Instead of using 3 viewpoints for the trajectory, only one viewpoint was used.
- `Dense+CLS`: Instead of the regression loss, classification loss was used, as proposed in (Goyal et al., 2019).

For each ablation, the same setup was used as for `Dense+RGR` – training the Pix2R model with 8 random sets of values of hyperparameters, and choosing the model

Table 3. Comparison of various ablations to the Dense+RGR model. We report the mean number of successful episodes for each model, and the p-values for Wilcoxon test between the ablated and Dense models.

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

with the best validation accuracy. This model is used to generate rewards for policy training, for each of the 16 scenarios with 5 random seeds for all the 3 descriptions.[3]

The mean successful episodes across all runs are reported in Table 3. Further, the p-values for Wilcoxon tests between each ablation and the `Dense` rewards is reported, from which we can make the following observations:

- Using only the last frame (`LastFrame`), or using mean-pooling instead of an LSTM to encode the language (`MeanpoolLang`) does not substantially affect the performance of the model. In both these cases, the resulting model is still statistically significantly better than `Dense` rewards. Both of these results agree with intuition, since the last frame can be used to predict the progress in the task, and since the linguistic descriptions are not particularly complex in the given domain, simply looking at which words are present or absent is often sufficient to identify the task without using the ordering information between the words.
- Using mean-pooling instead of an LSTM to encode the sequence of frames (`MeanpoolTraj`) drastically reduces the number of successful episodes, and the resulting model is no longer statistically significantly better than `Dense`. Again, this agrees with intuition, since it is not possible to infer the direction of movement of the robot from an unordered set of frames.
- Using a single view instead of multiple views (`SingleView`) results in a decrease in the number of successful episodes, and the resulting model is no longer statisticially significantly better than `Dense`. As mentioned earlier, using frames to represent trajectories (instead of actions as in prior work) requires addressing challenges such as perceptual aliasing and occlusion, and these ablation results suggest that using multiple viewpoints alleviates these issues.
- Using classification loss instead of regression (`Dense+CLS`) also leads to a drop in performance, again making the resulting model no longer statistically significantly better than `Dense`. This is consistent with our initial observation, as described in Section 4.1.3, wherein, the learning problem becomes more difficult due to partial trajectories when the classification loss is used.

[3]For `SingleView`, we used 8 random sets of hyperparameter values for *each* of the three viewpoints, and chose the model with the best validation accuracy.

7. Conclusion

We proposed an approach for mapping pixels to rewards, conditioned on a free-form natural language description of the task. Given paired ⟨trajectory, language⟩ data, first, a relatedness model – Pix2R – is learned between a sequence of states and a natural language description using supervised learning. This model is then used to generate intermediate rewards for policy training, for a task with natural language description. Our experiments on a simulated robot manipulation domain show that the proposed approach can significantly speed up policy learning, both in sparse and dense reward settings. Further, the qualitative analysis of the model agrees with intuition, and our ablation experiments show the impact of various design choices in our model.

The proposed approach can be extended in multiple ways. First, the current model only works for a single instruction and could be extended to use a sequence of instructions, for instance, by starting with the first instruction in the sequence, and transitioning to the next instruction when the prediction of the Pix2R model is above a threshold. Next, Pix2R currently encodes the trajectory and language independently, which are then concatenated to obtain a relatedness score. For more complex scenarios, it might be helpful to use an attention-based model to learn a mapping between spatio-temporal regions of the trajectory and words or phrases in the language. Finally, it may be useful to fine-tune the Pix2R model on trajectories seen during policy training. Our preliminary experiments to fine-tune the model did not result in conclusive findings, but a more thorough analysis is required.

References

Arumugam, D., Karamcheti, S., Gopalan, N., Wong, L. L., and Tellex, S. Accurately and efficiently interpreting human-robot instructions of varying granularities. *arXiv preprint arXiv:1704.06616*, 2017.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, A., Kohli, P., and Grefenstette, E. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

Blukis, V., Terme, Y., Niklasson, E., Knepper, R. A., and Artzi, Y. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on Robot Learning (CoRL)*, 2019.

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

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

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

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

Grzes, M. Reward shaping in episodic reinforcement learning. 2017.

Kaplan, R., Sauer, C., and Sosa, A. Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539*, 2017.

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

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

MacGlashan, J., Littman, M., Loftin, R., Peng, B., Roberts, D., and Taylor, M. E. Training an agent to ground commands with reward and punishment. In *Proceedings of the AAAI Machine Learning for Interactive Systems Workshop*, 2014.

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

Narasimhan, K., Kulkarni, T., and Barzilay, R. Language understanding for text-based games using deep reinforcement learning. *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sung, J., Jin, S. H., and Saxena, A. Robobarista: Object part based transfer of manipulation trajectories from crowdsourcing in 3d pointclouds. In *Robotics Research*, pp. 701–720. Springer, 2018.

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

Waytowich, N., Barton, S. L., Lawhern, V., Stump, E., and Warnell, G. Grounding natural language commands to starcraft ii game states for narration-guided reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 110060S. International Society for Optics and Photonics, 2019.

Williams, E. C., Rhee, M., Gopalan, N., and Tellex, S. Learning to parse natural language to grounded reward functions with weak supervision. In *AAAI Fall Symposium on Natural Communication for Human-Robot Collaboration*, 2017.

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