PixL2R: Guiding Reinforcement Learning using Natural Language by Mapping Pixels to Rewards

Anonymous Authors[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

While reinforcement learning (RL) has been successfully used to solve many problems, designing good reward functions continues to remain a challenge, limiting its applicability to more complex problems. To address this problem, several methods have been proposed that involve guiding an agent using natural language commands.

However, these techniques are still quite restrictive, often requiring object properties to be predefined (MacGlashan et al., 2014; Arumugam et al., 2017; Williams et al., 2017), assuming some structure in the natural language commands (Bahdanau et al., 2018), and/or requiring hand-designed features (Kuhlmann et al., 2004; Branavan et al., 2012), 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 we describe next.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Prior Work: LEARN

Consider an extension of the standard Markov Decision Process (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 in a standard MDP. Goyal et al. (2019) proposed a framework for learning in this modified MDP (which they denote as MDP+L) consisting 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 ith 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).

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.

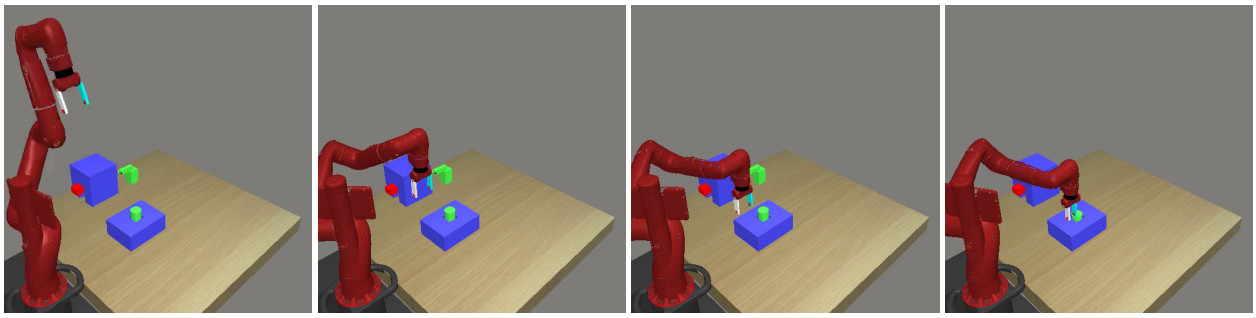

Figure 1. A simulated robot completing a task in the Meta-World domain

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 position 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.

3. Approach

To apply the framework described in Section 2 to domains where using the state information is crucial to understanding language (e.g. the domain in Figure 1), we propose PixL2R, which takes in the pixel representations of the states and a natural language description of the task, and maps them to rewards. The framework consists of a supervised learning phase (Section 3.1) and a policy training phase (Section 3.2) as in prior work, but with modifications as described below.

3.1. PixL2R: Pixels and Language 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.

3.1.1. 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.

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

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

See supplementary material (Section A) for the viewpoints used, and a diagram of the neural network.

3.1.2. Data Augmentation

Frame dropping. Given an input 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 uniformly 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.

3.1.3. 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.

Since partial trajectories might be difficult to classify as related or unrelated, we experiment with an alternate regression-based setting, 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.

3.2. Policy Training Phase

Having learned a PixL2R model as described above, the relatedness scores from the model can be used to generate language-based intermediate rewards during policy training on new scenarios. During policy training, the agent receives a natural language description of the goal, in addition to the reward from the environment. The PixL2R model is used to score trajectories executed by the agent against the given natural language description, to generate intermediate rewards. We used potential-based shaping rewards (Ng et al., 1999), which are of the form $F(s_t) = \gamma \cdot \phi(s_t) - \phi(s_{t-1})$, where s_t is the state at timestep t and $\phi : S \rightarrow \mathbb{R}$ 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 $[-1, 1]$.

4. Experiments

4.1. Dataset

We use Meta-World (Yu et al., 2019), a recently proposed benchmark for meta-reinforcement learning, which consists of a simulated Sawyer robot and everyday objects such as a faucet, windows, coffee machine, etc. Tasks in this domain involve the robot interacting with these objects, such as turning the faucet clockwise, opening the window, pressing the button on the coffee machine, etc. Completing these tasks requires learning a policy for continuous control in a 4-dimensional space (3 dimensions for the end-effector position, and the fourth dimension for the force on the gripper). While the original task suite consists of only one object in every task, we create new environments which contain one or more objects in the scene, and the robot needs to interact with a pre-selected object amongst those.

In a sparse reward setting, the agent is given a non-zero reward only on successfully interacting with the pre-selected object. In the absence of any other learning signal, the agent might have to learn to approach and interact with multiple objects in the scene in order to figure out the correct object. Using natural language to describe the task in addition to the sparse reward helps alleviate this issue.

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. Note that our algorithm does not need the policies used to generate the videos, so they could also be collected using human demonstrations.

We used Amazon Mechanical Turk to obtain free-form natural language descriptions for the videos. The details of the process, as well as some example descriptions can be found in the supplementary material (Section B).

4.2. Policy Training with Language-based Rewards

To empirically evaluate the effectiveness of PixL2R, 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

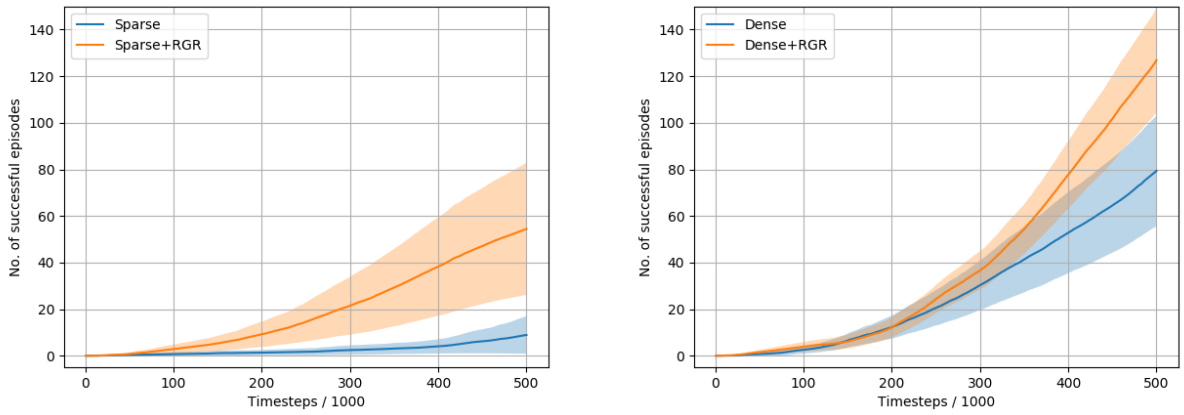

Figure 2. A comparison of policy training curves for different reward models. The shaded regions denote 95% confidence intervals.

position within a predefined region at the beginning of each episode.

Of these 39 scenarios, there was no statistically significant difference between policy training using sparse and dense rewards for 23 scenarios, suggesting that these scenarios are either too simple to learn even from sparse rewards, or are too difficult to learn within 500,000 timesteps even with hand-designed dense rewards. As such, we used only the remaining 16 scenarios for subsequent experiments.

First, 15 policies were trained using sparse rewards (`Sparse`) on these selected scenarios with different seeds. Next, policies were 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 2 (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 2 (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.

A Wilcoxon signed-rank test was then performed between the sets of mean successful episodes 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.

For the dense reward setting, policy training with language-based rewards using the classification setting (`Dense+CLS`) was also performed. Although it results in an improvement in the mean successful episodes over `Dense` rewards, the improvement was not statistically significant, suggesting that the proposed regression setting helps learning more effectively from partial trajectories. More details for the results on the classification setting, as well as other additional experiments used to analyze the supervised learning phase can be found in the supplementary material (Section C).

5. 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 – PixL2R – 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.

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.

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, 2012.

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.

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.

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.

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.

A. Neural Network architecture

Figure 3 shows the viewpoints used in our model. Figure 4 shows a diagram of the neural network architecture described in Section 3.1.1.

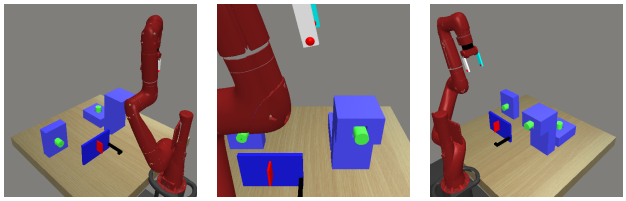

Figure 3. Viewpoints used for data collection and experiments.

B. Collecting Language Descriptions

To collect English descriptions from videos of the robot performing the 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 5). This enabled us to get descriptions that use the real-world object names, without priming the workers with specific words. [1]

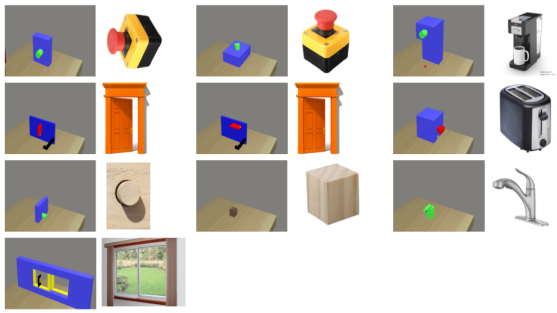

Figure 5. List of objects used

Some example descriptions collected are shown in Table 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

C. Additional Experiments

C.1. 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

[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*.

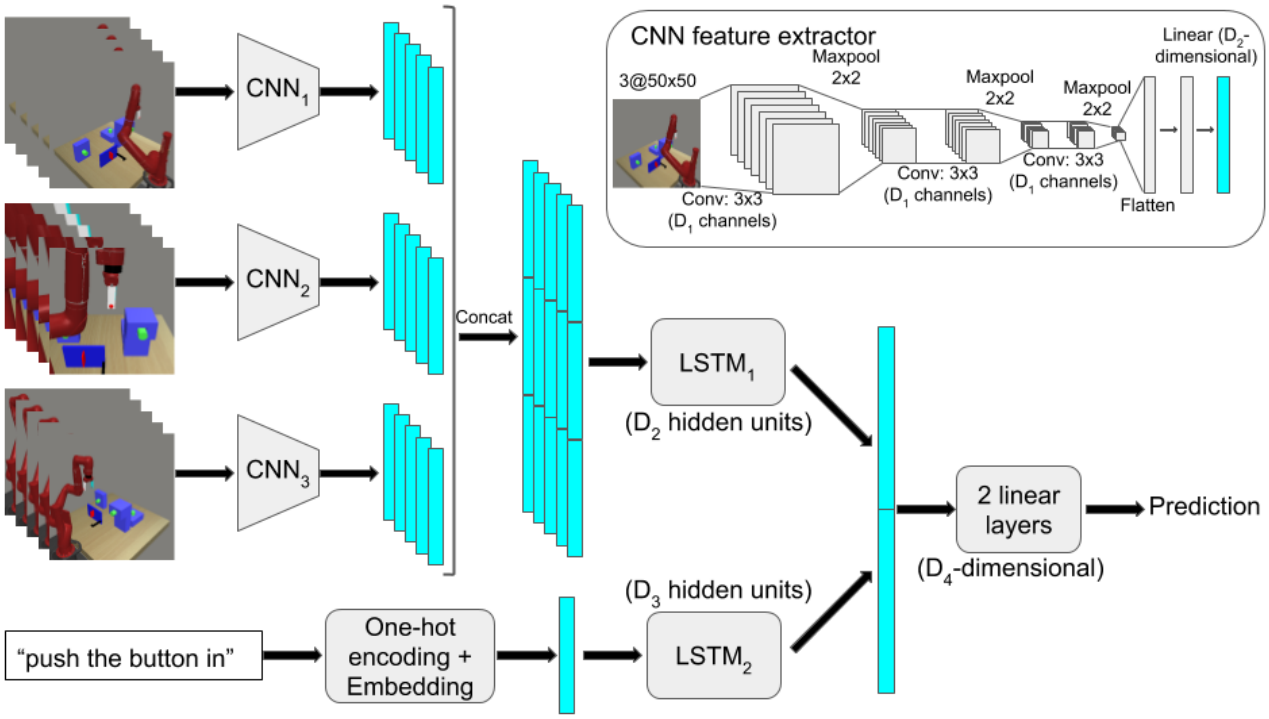

Figure 4. Neural network architecture: The sequence of frames from the three viewpoints are passed through three separate CNN feature extractors. The resulting feature vectors are concatenated across views. The sequence is then passed through an LSTM to obtain an encoding of the trajectory. The given 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.

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				

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.

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.

C.2. 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 PixL2R 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).

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

For each ablation, the same setup was used as for `Dense+RGR` – training the PixL2R model with 8 random sets of values of hyperparameters, and choosing the model 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.[2]

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 3.1.3, wherein, the learning problem becomes more difficult due to partial trajectories when the classification loss is used.

[2]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.