ZERO-SHOT REWARD SPECIFICATION VIA GROUNDED NATURAL LANGUAGE

Parsa Mahmoudieh[1], **Deepak Pathak**[2], **Trevor Darrell**[1]
[1] UC Berkeley, [2] Carnegie Mellon University
{parsa.m, trevor}@cs.berkeley.edu, dpathak@cs.cmu.edu

ABSTRACT

Reward signals in reinforcement learning are expensive to design and often require access to the true state which is not available in the real world. Common alternatives are usually demonstrations or goal images which can be labor-intensive to collect. On the other hand, text descriptions provide a general, natural, and low-effort way of communicating the desired task. However, prior works in learning text-conditioned policies still rely on rewards that are defined using either true state or labeled expert demonstrations. We use recent developments in building large-scale visuolanguage models like CLIP to devise a framework that generates the task reward signal just from goal text description and raw pixel observations which is then used to learn the task policy. We evaluate the proposed framework on control and robotic manipulation tasks. Finally, we distill the individual task policies into a single goal text conditioned policy that can generalize in a zero-shot manner to new tasks with unseen objects and unseen goal text descriptions.

1 INTRODUCTION

Previous efforts have explored image-based goal specification, with significant successes in visual navigation and manipulation tasks (Pathak et al., 2018; Nair et al., 2018; Fu et al., 2018; Singh et al., 2019). Yet existing image-based goal specification paradigms are limited because they are typically limited to a particular scene *instance* in an environment, whereas a *semantic* goal comprises multiple possible scene configurations. Reinforcement learning offers one of the most appealing premises in the study of AI: from a reward signal alone, algorithms which learn optimal policies that maximize expected reward can learn to perform navigation, dexterous manipulation, and host of other impactful tasks. However, discovering or specifying a reward function for a given task is often a very challenging problem, especially when one is considering agents that can learn from uninstrumented environments, e.g., from raw image observations alone. We wish to have an agent that can learn purely from pixels, with no access to the underlying state of the environment at any point during learning or task execution. Achieving this goal without access to an instrumented reward function has been exceedingly challenging.

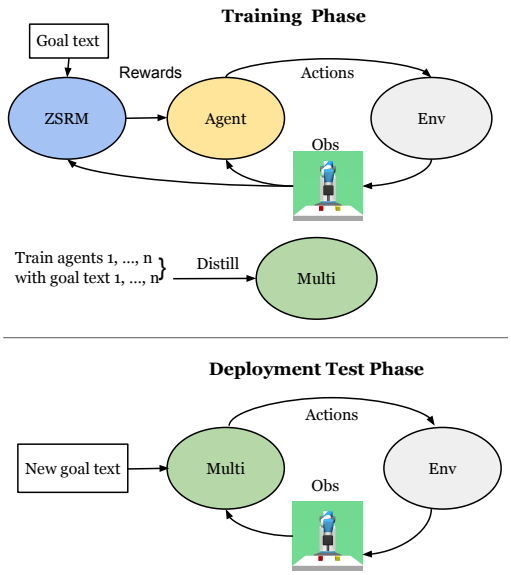

Figure 1: Our method uses only goal text from user and images from the environment at training time. We train several agents on several tasks and distill their policies into a single goal text conditioned policy that can generalize to new tasks with an unseen goal text description in a zero-shot manner. We assume no access to environment reward, state, demonstrations, or goal images at either train or test time.

One can use image-based reward specification to cause a robot agent to navigate to a particular chair next to a specific tall plant, but that agent may not always succeed at the generic task of "go to a chair next to a tall flowering plant": e.g., if the goal specification image shows a red chair next to a plant with a yellow flower the agent may navigate

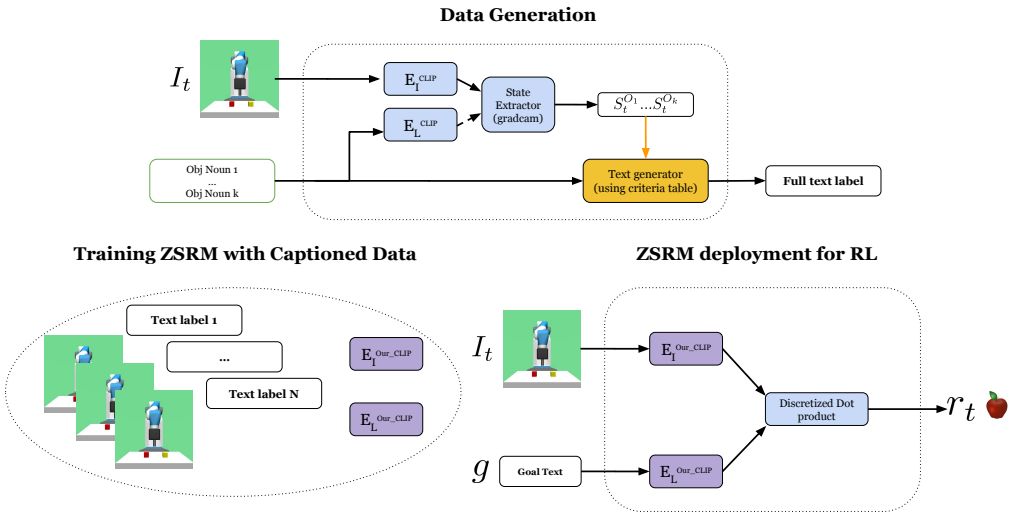

Figure 2: Our Zero-Shot Reward Model (ZSRM) is made by first generating spatial text labels of random exploration collected images with random initial states of arm and objects. We train ZSRM with the generated data in the same style of CLIP. We then use a discretized dot product of current image observation embedding and goal text embedding as our zero-shot reward. We train an agent via Reinforcement Learning with our reward model that is computed using only goal text and current observation as input. We assume no access to environment reward, state, demonstrations, or goal images at train or test time.

away from a scene with a blue chair next to a red flower, depending on the model's underlying image representation. To be sure that the true goal is properly specified irrespective of the invariances of the model's underlying perceptual representation, a user may have to provide a set of goal image examples that cover the variation of the target concept, potentially a very expensive undertaking to collect or generate.

We advocate *semantic reward specification via grounded natural language*, where a user describes a goal configuration in the world using a natural language description referring to entities in the world. This direction has long been a "holy grail" of AI research and a presumed capability of AI science fiction—the ability to instruct a robot with natural language—yet attempts have been limited by the state of the art in grounded language perception. It also falls under the general umbrella of leveraging large-scale passive data to bootstrap embodied learning, where we rely on language as a means to provide necessary reward signal which might be missing in visual data alone.

Several previous efforts train reward functions or policies that take natural language as input for goal description (Oh et al., 2017; Bahdanau et al., 2018; Zhou & Small, 2020; Goyal et al., 2020; Fu et al., 2019; Hermann et al., 2017; Shao et al., 2020). They all however rely on reward signals that have access to state of the system or demonstrations of the task distribution they are training on. There are works that use human videos to learn reward functions to train their agent with (Shao et al., 2020; Sermanet et al., 2018; 2016), but they require a curated dataset of humans performing the tasks.

Recently however, the advent of large-scale multimodal training data together with large capacity language and vision deep learning models has significantly advanced the state of the art. A steady series of innovations have advanced grounded language modeling, from early work on multimodal translation and fusion models (Barnard et al., 2003; Quattoni et al., 2007; Guadarrama et al., 2016), to large-scale joint embedding models (Frome et al., 2013; Radford et al., 2021), to the plethora of multimodal transformer models currently under investigation (Su et al., 2019; Lu et al., 2019; Chen et al., 2019; Hu & Singh, 2021). CLIP, in particular, demonstrated a transformative advance on zero-shot object recognition (Radford et al., 2021).

One way to communicate text-based goal to a robot is by simply offering a description of the goal configuration in natural language and using the CLIP embedding dot product with an observed image to evaluate proximity to goal state. Surprisingly, this can work in simple cases, for examples see the top example in figure 5. However, for more complex goals, such as those involving spatial relationships, the simple solution has poor performance as shown in the bottom example of figure 5.

To overcome these limitations and scale to complex manipulation tasks, we spatially ground the natural language goal in the image. We factor 'what' vs 'where/how' aspects of goal state, and offer a novel spatial-salience scheme to generate data using this factorization (Figure 2 top). We argue that existing (e.g., CLIP-like) models can be used to ground the *what* aspects of a goal quite effectively,

including appropriate attribute and concept-level generalization, while a separate *where/how* module can ground spatial relationship aspects of goal configuration. On this generated data, we first learn our Zero-Shot Reward Model (ZSRM) and then use ZSRM to learn individual text-conditioned policies. Finally, the individual task policies are distilled into a single goal-text-conditioned policy which is multi-task in nature and can execute unseen tasks in a zero-shot fashion – from a natural language description of task – without having to train a new policy for every new task.

2 METHOD

In our work we assume an agent only has access to a text description of the goal desired by the user and image observations from the environment. At no point during training or testing does the agent have access to demonstrations, goal images, or reward from the environment. The agent only has access to a reward model that takes images and goal text as input to provide progress towards goal text description with a reward score output. The reward is then used to teach the agent how to achieve the goal described by the text with online reinforcement learning. Given these assumptions we will now describe how we provide a zero-shot reward model by leveraging CLIP and how we learn a text conditioned policy with this model.

2.1 VANILLA DOT-PRODUCT VISUOLINGUISTIC BASE REWARD MODEL

Our base model is the most intuitive way to use the CLIP model to compute reward. Our base model simply computes reward by taking a dot product between the goal text feature and image observation feature through CLIP's language and image encoders respectively: $r_t = E_I(I_t) \cdot E_L(g)$ A subset of tasks can be learned with this reward model. We visualize the limits of this model on two tasks in Figure 5. One significant limitation of CLIP is that it cannot distinguish spatial relationship of objects in images. This limits our base reward model from being useful for tasks that have spatial goals. Our full zero-shot reward model remedies this issue by leveraging CLIP in a very different way.

2.2 SPATIALLY GROUNDED VISUOLINGUISTIC ZERO-SHOT REWARD MODEL (ZSRM)

Data Generation Using CLIP-Saliency Phrase Grounding We use a simple method that generates texts with spatial information for images using phrase grounding and spatial relationship processing. We assume we have access to the noun phrases that will be used by the user to specify their full goal text. Figure 2 illustrates how the object noun phrases are passed through CLIP's language encoder and the current image observation is passed through CLIP's image encoder. We use the language encodings as class embeddings of each object noun phrase to perform a saliency analysis (using Grad-CAM (Selvaraju et al., 2017)) on the image encoding of the observation on the last convolutional layer (specifically, the ReLU layer of CLIP's ResNet-50 backbone). Saliency models such as Grad-CAM generally output a heatmap of the features that indicate a class exists in the current image input. The state extractor in Figure 2 computes the "state" of each object using the argmax of the saliency heatmap (see Figure 3). Once we have the the object states (object pixel coordinates), we use the criteria described in the next section to generate a full text description of the image describing the spatial relationship between the objects. The images we use to label are randomly sampled robot arm and object locations with random actions taken by the agent to move the arm.

Spatial Language Label	Label Grounding Criteria
Obj1 on the left of Obj2	$O^2_x > O^1_x$
Obj1 on the right of Obj2	$O^1_x > O^2_x$
Obj1 on top of Obj2	$\lvert O^1_x - O^2_x \rvert < \epsilon_1$ & $O^2_y < O^1_y < O^2_y + \epsilon_2$
Obj1 below Obj2	$\lvert O^1_x - O^2_x \rvert < \epsilon_1$ & $O^1_y < O^2_y < O^1_y + \epsilon_2$
Obj1 in between Obj2, Obj3	$min(O^2_x, O^3_x) < O^1_x < max(O^2_x, O^3_x)$
Obj1 in front of Obj2	$O^1_{x2} > O^2_{x2}$
Obj1 behind Obj2	$O^2_{x2} > O^1_{x2}$
Obj1 close to Obj2	$\lVert O^1_{xy} - O^2_{xy} \rVert_2 < \epsilon$
Obj1 inside of Obj2	$\lVert O^1_{xy} - O^2_{xy} \rVert_2 < \epsilon$

Table 1: Spatial Language Grounding Criteria.

Spatial Language Grounding Criteria We generate a spatial text label using the object pixel coordinates of one or more camera views as input to match various criteria. The criteria that matches our object state determines the text label of the image. The text labels used are the simplest spatial descriptions we could think of. We did not engineer what text labels to use.

Our spatial language grounding criteria are fully defined in Table 2.2. The first set (left of, right of, on top of, below, and in between) assume coordinates in a front camera view and the semantics

Figure 3: a) Mask R-CNN object detection results b) Mask R-CNN detection results on far view c) Mask R-CNN instance segmentation results on far view d) Grad-CAM result with 'red block' text input (white being highest intensity) e) Grad-CAM result with 'yellow block' text input (white being highest intensity).

are defined in that camera view with positive y in the upward direction and positive x in the right direction. The second set (in front of & behind) has access to a left camera view with coordinate x2 pointing towards the right which is towards the front in the first camera view and y2 pointing upward similar to the first camera view. The second camera is needed to know if the object is placed in the front or behind another object correctly. The third set (close to & inside of) require access to a front camera view with a 45 degree downward tilt towards the ground. This camera view is needed to see if the object is getting closer to another object in two orthogonal directions at once where as a left only or front view only allows you to determine one dimension of closeness. For "inside of" a 45 degree camera helps the agent see if the object is going inside another object without occlusion. The ϵ threshold for "inside of" is much smaller than for "close to" since the centroid can come much closer when an object goes inside a container object.

Full Reward Model Training and Usage After generating spatial language labels for randomly collected images using CLIP-saliency phrase grounding, we train our full reward model similar to CLIP (Radford et al., 2021) with the similar visuolinguistc contrastive loss where the model essentially predicts which caption matches which image in every batch. The cosine similarity of the image and text embeddings of the correct pairs in the batch are maximized while the cosine similarity of the embeddings of the incorrect pairs are minimized. We initialized our model with the pretrained language and image encoders from CLIP. This, what we call 'full reward model', is essentially same as the base reward model but now it has been "fine-tuned" with spatial-grounded text-image data generated automatically. To prevent catastrophic forgetting and overfitting to just the new data, one could maintain a distillation loss with respect to the old model. However, we found that training the model from scratch on this new data to work well. Finally, the dot product of our new model can directly be used as reward but we found thresholding the dot product to obtain binarized reward to work better. This finetuned zero-shot reward model can be used for a broad set of manipulation tasks to push or place objects to semantic locations which we showcase in our results.

2.3 Language Conditioned Multi-task Policy

Now that we have a method for learning tasks in a zero-shot reward fashion, we would like to be able to not require training a new policy for every new task and ideally have a language conditioned multi-task policy that takes in the goal text description of an unseen task and executes the task without needing any new samples from the environment.

To approach this goal, we first learn several tasks via reinforcement learning using our zero-shot reward model. We then create a large dataset of the rollouts of those polices and pair each trajectory with the goal text description of the tasks it was trained to learn. We then use behavioral cloning (supervised learning for predicting actions) to learn a policy that takes images and text goal task description as input and actions as target outputs. We use CLIP's language encoder as the text goal embedding that is fed to the multi-task policy. In addition we use image augmentation techniques to aid behavioral cloning in learning more robust policies.

3 Experiments

3.1 Phrase grounding and Base Zero-shot Reward Model Visualization

Object detectors are one way to extract object states for our spatial text generator, however, they are usually not used off the shelf and need to be fine-tuned with in domain data. In Figure 3, we

show pretrained Mask R-CNN (He et al., 2017) outputs on different camera views for the block stack environment. As you can see in the first two subfigures, the blocks are not proposed as objects with Mask R-CNN from both far and close camera views. This is a demonstration of the need for in-domain fine-tuning of object detectors to work in the environment you want to use. Our Grad-CAM output from CLIP however, highlights exactly the objects we are interested in from the object noun phrases that describes each object in the last two subfigures. Another limitation with Mask R-CNN is that even if the object proposals were good, it would not necessarily classify objects it has not been trained for and therefore not output a filtered set of object proposals. In other words, we would not know which object is which if the detector hasn't been trained with the label of objects we care about.

In Figure 5 we show what our base reward model outputs on two goal descriptions: 1. inverted pendulum 2. yellow object close to a blue object. For the first goal description we observe that the dot product increases as the pendulum becomes more inverted from either side as desired. For the second goal description we observe that as the blue object gets closer to the yellow object the dot product increases except for the closest image where it dips which results in an undesired output. We observed this example and many others that the base reward model is not sufficient for recognizing object spatial relationships. The encoders are good at identifying what objects are in the image however, which is what we leverage to generate paired language image data for our full reward model.

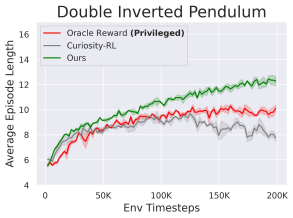

Figure 4: We showcase our base reward model trained policy in Double Inverted Pendulum performing slightly better than oracle reward. We average our results over 3 seeds per task.

3.2 Full Zero-shot Reward Model Results

In Figure 6 we show how our full reward model performs on three manipulation tasks. We train each task using our full zero-shot reward model output as reward for the PPO reinforcement learning algorithm (Schulman et al., 2017). We then train for the same tasks with other types of reward functions as baselines or privileged methods for comparison:

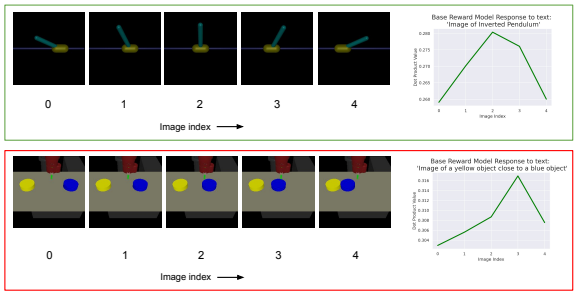

Figure 5: We visualize the results of the base reward model which is a trivial dot product between the goal language description and image observation. The top row (green box) displays a successful utilization of the base reward model and the bottom row (red box) shows a failure case. The x-axis represents image index.

a) Oracle reward (privileged): this reward function has direct access to state to determine task completion. In other words it uses true x,y,z spatial positions of the objects to determine if the desired spatial relationship is reached and outputs 1 to the RL algorithm for every timestep the conditions of the task are met.

b) VICE (privileged): VICE(Fu et al., 2018) is used as a privileged method that has access to task goal image for comparison. We train the VICE reward model as a binary goal classifier that is trained with true goal images of the task such as "red block on top of yellow block" as positive images and images that aren't in the correct goal configuration as negative images. We discretize the model to output 1 if it determines the current image is positive and 0 otherwise.

c) Ours-base: Our base zero-shot reward model that uses the dot product between the goal text feature and image observation feature of CLIP as reward.

d) Curiosity-RL (Pathak et al., 2017; Burda et al., 2018): Curiosity is used as a baseline since it only has access to images similar to our method for computing reward, but has only been successful for videogames such as Atari and Mario or locomotion where exploring new states leads to progressing through the task (going further in levels of game for example). It is less privileged however, in that it does not use language input for task specification.

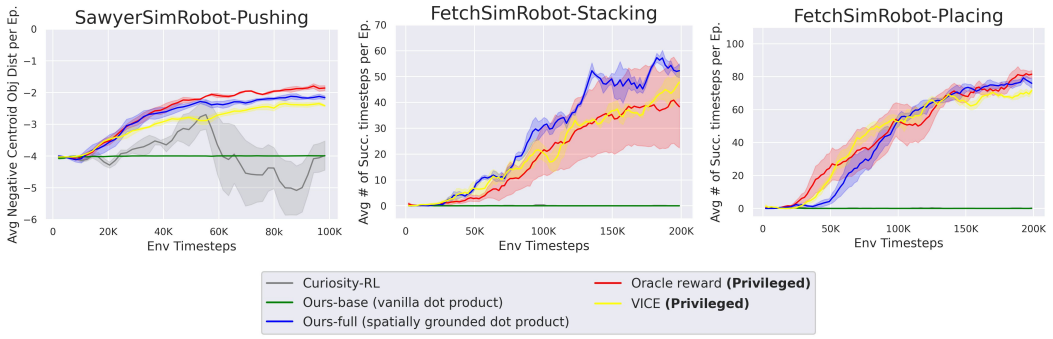

Figure 6: We showcase our Full Reward Model performing as well as oracle reward and VICE both of which are priviledged on SawyerSimRobot-Pushing (pushing a blue puck close to a yellow puck), FetchSimRobot-Stacking (stacking a yellow block on top of a red block), and FetchSimRobot-Placing (placing a yellow block on the right of a red block). We average results over 3 seeds per task. Oracle reward is privileged with true state information used to compute proximity to goal. VICE is privileged with true goal images used to train its reward model in classifying observations as close to goal or not.

In Fig. 6 we observe that our Full Reward Model performs as well as oracle reward and VICE both of which are priviledged on SawyerSimRobot-Pushing (pushing a blue puck close to a yellow puck), FetchSimRobot-Stacking (stacking a yellow block on top of a red block), and FetchSimRobot-Placing (placing a yellow block on the right of a red block). For Curiosity-RL we see that it learns the pushing pucks close together task but then starts learning separation of the pucks which reemphasizes that curiosity is only useful for tasks where exploring new dynamics leads to going farther in the task. Curiosity also has some trouble learning the double inverted pendulum task because the dynamics of the pendulum swinging can be hard to predict and therefore have misleading higher reward. Curiosity also does not learn the other manipulation tasks (stacking and placing) as those are more complex tasks that are harder to reach by exploration.

For Double inverted pendulum (Fig. 4) our base reward model does better than oracle by chance which we speculate is because the oracle reward was originally designed for state input and was not tuned for learning image to reward mapping. Our base reward model fails for pushing, stacking, and placing, which take "an image of a yellow block on top of a red block", "an image of a yellow object close to a blue object", and "an image of a yellow block on the right of a red block" as language input for those tasks respectively.

The oracle reward function for Double-Inverted-Pendulum is alive bonus minus distance penalty minus velocity penalty. The oracle for SawyerSimRobot-Pushing is a sparse reward that outputs one when the centroid distance between two pucks are below a threshold. The oracle reward for FetchSimRobot-Stacking is a sparse reward that outputs one when a yellow block is within a horizontal and vertical threshold distance of a red block. The oracle reward for FetchSimRobot-Placing is a sparse reward that outputs one when a yellow block is correctly placed on the right of a red block. See Figs. 3 & 5 for image observation examples of FetchSimRobot and SawyerSimRobot.

3.3 Multi-task Policy Results

In order to avoid having to train a new policy for every new task we want to learn, we train a multi-task policy with a set of training tasks and then show that it can generalize to a set of unseen test tasks by leveraging CLIP's language model to encode the goal text description of the tasks as conditioning input to our multi-task policy.

In FetchSimRobot env, we train 18 tasks with PPO each for 200K environment steps with our zero-shot reward model described in the previous section and show generalization results for 18 unseen test tasks by training a language conditioned policy with behavior cloning on rollouts of the 18 training tasks. We collect 5000 timesteps per task which is around 50 trajectories. We do this for pickplace-left, pickplace-right, and stack tasks for different object combinations. The object training colors are red, green, and yellow, and the object test color is blue. So the multi-task policy has never seen blue block in text input or image input.

	Seen distribution				Unseen distribution			
	train tasks		test tasks		train tasks		test tasks	
(episode reward stats)	mean	s.e.	mean	s.e.	mean	s.e.	mean	s.e.
No Conditioning	17.91	1.11	14.82	0.97	14.81	1.02	10.79	0.85
Primitive Code Cond.	26.71	1.23	17.20	1.03	17.03	1.07	11.74	0.87
Language Cond.	29.89	1.28	22.41	1.09	21.14	1.16	15.69	0.98

Table 2: Multi-task Policy Performance Results: We report multi-task average and standard error performance across all 18 training tasks and all 18 test tasks (unseen object color in goal text or image) for seen and unseen initial state distributions with no conditioning, primitive code conditioning, and language conditioning. The values reported are the episode reward averaged across 50 seeds per task policy rollout. The tasks are robotic arm manipulation of objects to different target semantic relationships of each other in the Fetch-SimRobot environment. While the policies were trained using our full zero-shot reward model, the reward metric reported is oracle reward to evaluate true performance.

In evaluation, for each task we average episode rewards over 50 seeds. We then average across all training tasks to get the train metric and across all test tasks to get the test metric respectively. We compare our language conditioned policy with a policy trained without task labels, and one trained with a primitive code as the conditioning input. The latter baseline simply labels the training tasks with integers 0 to 17. Since the Primitive code conditioned policy has only been trained with primitive codes of the training tasks, when evaluating it for test tasks we allow the policy to use CLIP's language embedding to find the closest corresponding training primitive code. This is done by comparing the text description embedding of all the training tasks to the test task text description embedding and choosing the training task primitive code corresponding to the smallest L2 distance.

We use the same oracle reward that has direct access to state in figure 6 to measure the performance of our multi-task models. Every timestep the objects are in the correct target text description the agent is rewarded 1 point. The objects never start in the correct target state. Therefore it always takes several timesteps for the policy to move the objects to the correct state. All episodes timeout after 100 timesteps or if one of the objects falls off the table. In RL training and in policy rollout collection for behavior cloning each object has a one block width of variation in starting point location. We tested the policies on same distribution of start point variation interval (seen initial state distribution) and double the start point variation interval (unseen initial state distribution).

In Table 2 we see that no conditioning policy has the lowest performance as expected since there is full ambiguity in what task to perform given only an image. However, since the no conditioning policy has been trained across many different object colors going to different target states it has learned general displacement of blocks to different semantic locations randomly that can sometimes be moved to the correct semantic location by chance during testing. For the training tasks it has higher performance than testing tasks since it can memorize to execute some of the training tasks that have almost the exact same initial states sampled during testing as those in the behavior cloning dataset and thereby execute correctly as memorized. We observe that the primitive code conditioned policy has much higher training performance than the no conditioning policy since it can disambiguate what task it needs to execute.

We observe that the language conditioned policy performs significantly better on test tasks than the primitive code conditioned policy since it can use the language embedding to infer the target task determined by the goal text description. The primitive code conditioning policy has access to the language embedding only to determine the closest training primitive code to the target task description. It's interesting to observe that the primitive code conditioning performs better than no conditioning on the test tasks because the closest primitive code can sometimes lead to a successful execution of the test task since it has some signal towards the correct task. One such example is that the test goal description task of "a blue block on the right of a yellow block" is mapped to the primitive code of the training goal description task of "a red block on the right of a yellow block". In Table 2 we also observe the same trend in the policy performances in unseen intial state distribution: the no conditioning policy performing the poorest on train and test tasks, the primitive code conditioned policy performing significantly better on both, the language conditioned policy performing the best at training and test tasks via its language structured specification of tasks.

We train the language conditioned policy with ResNet18 image encoding that is concatenated with CLIP language encoding both of which are pushed through an fc layer before concatenation. The concatenated vector is then inserted into two fc layers before predicting actions. The primitive code conditioned policy swaps the CLIP language encoding with an integer from 0-17 representing the primitive code input. The no conditioned policy only has two fc layers on top of the same image encoding as the other two policies. We apply an L2 regression loss on the output for predicting

continuous actions (behavior cloning). The policy is trained using Adam optimizer with AMS grad with learning rate of 1e-4. The images are augmented with PyTorch RandomResizedCrop of 0.95 to 1.0 area and 0.98 to 1.02 aspect ratio randomization and resized to original image dimensions of 128x128. All the policies are trained for 300 epochs.

4 Related Work:

Goal conditioned policies Goal conditioned policies allow a user to specify the agent's goal. States (Schaul et al., 2015; Andrychowicz et al., 2017) and Images (Pathak et al., 2018; Nair et al., 2018; Fu et al., 2018; Singh et al., 2019) are one way of specifying goal. However, they assume that the user has access to a photo or state of the completed task to give to the agent. We assume no access to goal images or state to minimize human labor and instrumentation, and use language which provides a natural form of supervision.

Goal text conditioned policies Several previous efforts train reward functions or policies that take natural language as input for goal description (Oh et al., 2017; Bahdanau et al., 2018; Zhou & Small, 2020; Goyal et al., 2020; Fu et al., 2019; Hermann et al., 2017; Shao et al., 2020). They all however rely on reward signals that have access to state of the system or demonstrations of the task distribution they are training on.

Learning reward functions There are works that use human videos to learn reward functions to train their agent with (Sermanet et al., 2018; 2016; Shao et al., 2020). We however, don't need a curated dataset of humans performing the tasks we want our agent to train with. Having humans perform all tasks we are interested in may not scale well with the amount of labor needed for recording and curating those datasets. CLIP has the advantage of learning a caption model from 400 million image, text pairs from the publicly available sources on the internet WIT.

No Environment Reward There have been recent work on methods that use no reward signal from the environment to train for specific tasks. Two notable ones are Curiosity (Pathak et al., 2017) and DIAYN (Eysenbach et al., 2018). Curiosity has only been successful on navigation and video games with discrete action space and DIYAN has mostly been successful on direct state input. Our method however, can learn a subset of manipulation tasks in continuous action space on raw pixels and can be specified easily with goal text.

Utility of large vision language models in RL There has been recent work that leveraged CLIP to do many complex manipulation tasks (Shridhar et al., 2021). They however, have access to labeled expert demonstrations for training their policy. We assume no access to demonstrations or goal images at training or test time.

5 Discussion

In this work, we presented a method for learning a set of object manipulation tasks without access to state of the system by computing reward from pixels conditioned on text goal description alone. Our method doesn't use goal images or demonstrations at either training time or test time. We devised a zero-shot reward model that leverages a language vision model (CLIP) that has been trained on a very large dataset of captioned images on the internet to compute progress (reward) towards a goal text description. We use this zeroshot-reward model to collect data on many tasks to then supervise a language conditioned multi-task policy that can execute new tasks without need of extra training. There are many future directions that can expand the abilities of our reward model such as taking into account pose of objects and state of objects (such as closed door).

Finally we must address the ethical perils inherent in our leverage of models trained on large-scale vision and language datasets. Such datasets are well known to suffer from dataset bias that can cause failure or unintended harm (Buolamwini & Gebru, 2018). While the near-term risks appear to be limited with the robotic applications presently envisioned, practitioners should continuously monitor systems for bias against underrepresented groups and ensure that robotic systems work across all socioeconomic domains. Techniques for bias assessment and debiasing should be employed whenever possible to ensure this remains the case.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017. 8

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018. 2, 8

Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M Blei, and Michael I Jordan. Matching words and pictures. 2003. 2

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pp. 77–91. PMLR, 2018. 8

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018. 5

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. 2019. 2

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 8

Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. 2013. 2

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *arXiv preprint arXiv:1805.11686*, 2018. 1, 5, 8

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019. 2, 8

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. *arXiv preprint arXiv:2007.15543*, 2020. 2, 8

Sergio Guadarrama, Erik Rodner, Kate Saenko, and Trevor Darrell. Understanding object descriptions in robotics by open-vocabulary object retrieval and detection. *The International Journal of Robotics Research*, 35(1-3):265–280, 2016. 2

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 5

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017. 2, 8

Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. *arXiv preprint arXiv:2102.10772*, 2021. 2

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019. 2

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, 2018. 1, 8

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pp. 2661–2670. PMLR, 2017. 2, 8

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017. 5, 8

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2050–2053, 2018. 1, 8

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Learning visual representations using images with captions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2007. 2

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 2, 4

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *ICML*, 2015. 8

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017. 3

Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016. 2, 8

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018. 2, 8

Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020. 2, 8

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. *arXiv preprint arXiv:2109.12098*, 2021. 8

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019. 1, 8

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 2

Li Zhou and Kevin Small. Inverse reinforcement learning with natural language goals. *arXiv preprint arXiv:2008.06924*, 2020. 2, 8