REWARD-INDUCED REPRESENTATION LEARNING

Ayush Jain, Youngwoon Lee, Karl Pertsch
University of Southern California
{ayushj,lee504,pertsch}@usc.edu

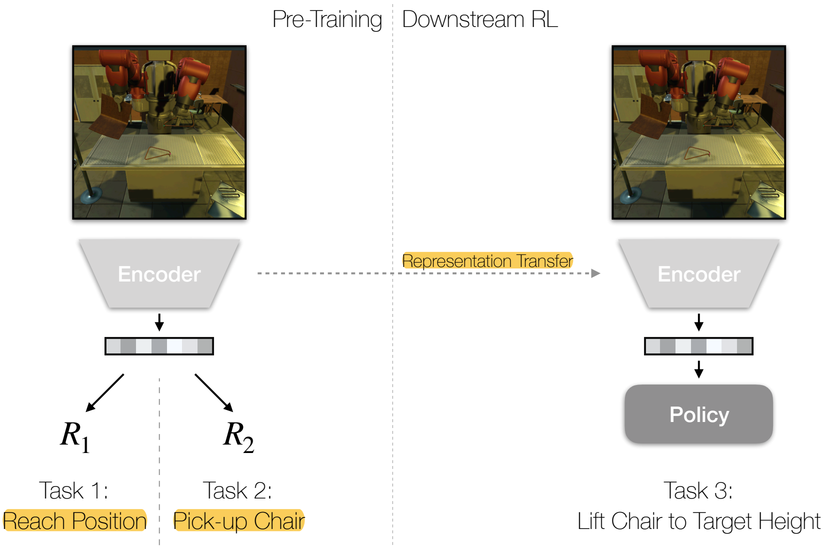


Figure 1: In complex environments we need to learn representations that focus on modeling the important aspects of the environment. We propose to use task information to guide representations as to which parts of the environment are important to model. **Left**: We pre-train a representation by inferring reward values for multiple tasks from a shared representation of the high-dimensional input, encouraging it to focus on parts of the environment that are important for solving the training tasks, e.g. it will focus on the arm and the furniture pieces instead of the background texture. **Right**: We can transfer this representation into a policy for more efficient learning of downstream tasks from the same task distribution that the representation training tasks were drawn from, e.g. tasks focused on robot and furniture pieces like lifting the chair to a certain height.

1 INTRODUCTION

Intelligent agents interacting in the real world need to be able to cope with highly complex environments, processing an immense amount of input information in every second. From a drive through Manhattan to navigating a busy intersection in Tokyo, humans are confronted with that same amount of detailed input information, but have the remarkable ability to condense this raw input data stream into a much compressed representation on which the decision making system operates. In this project we want to enable machine learning agents to build such compressed representations that ignore most of the input data and instead focus on the important aspects, enabling faster learning of downstream tasks.

In recent years, the training of artificial agents for sequential decision making problems has seen remarkable progress, especially in the subfield of deep reinforcement learning (RL) (Mnih & et al. (2015); Levine et al. (2016); Silver et al. (2016); Haarnoja et al. (2018); OpenAI (2018)). However, it has been shown that learning decision policies from high-dimensional input poses a significant challenge to these algorithms, as they need to learn to extract useful information from the input using a typically sparse reward signal. To improve training efficiency it has become common practice to pre-train feature extractors that reduce the input dimensionality while trying to retain as much information as possible (Janner et al. (2019); Lee et al. (2019); Gregor et al. (2019); Oord et al. (2018); Clavera et al. (2019)).

There are two dominant paradigms for the unsupervised pre-training of such feature extractors: *generative* and *discriminative* modeling. The generative approach directly trains a model of the high-dimensional input data via reconstruction or prediction objectives, constraining an intermediate representation within the model to be low-dimensional, e.g. in VAEs or predictive models (Janner et al. (2019); Lee et al. (2019); Gregor et al. (2019)). The discriminative approach instead optimizes a lower-bound on mutual information between the high-dimensional input and the representation using noise-contrastive estimation (Gutmann & Hyvärinen (2010); Oord et al. (2018); Clavera et al. (2019)). By design, such approaches are forced to model *all* information in the input data and cannot discriminate between what is *useful* to model and which input information can be ignored. Therefore, highly expressive models are required for modeling all facets of the environment and as a result the efficiency gains for downstream learning reported in prior work (Janner et al. (2019); Lee et al. (2019); Gregor et al. (2019); Clavera et al. (2019)) only considered relatively clean environments of low detail and complexity. In order to scale RL from high-dimensional observations to more complex environments we need representation learning methods that can focus on the *important* information in the input data.

In this work we propose to use task information for guiding representation learning approaches as to which part of the input information is important to model and which parts can be ignored. This is a natural choice as we aim to solve tasks with the learned representation and we can only decide the usefulness of information with respect to a distribution of target tasks. For example, it is useful to model the fur color of a dog if the tasks are related to classifying different dog breeds, but not so useful if the tasks are about distinguishing different animals.

Concretely, we propose to use a large dataset of trajectories annotated with rewards from a multitude of tasks drawn from the task distribution of interest. We train predictive models over these reward labels for learning a *reward-induced* representation that captures only information that is useful for predicting rewards and therefore useful for solving tasks from this task distribution. In our preliminary experiments we show that such representations facilitate downstream reinforcement learning of novel tasks from within the task distribution and that they are more robust to distractors in the environment than conventional generative representation learning approaches. This work is a first step towards scaling representation learning for decision making to environments of real-world complexity.

2 Related Work

There has been a wide range of works that apply different forms of unsupervised representation learning to improve the sample efficiency of reinforcement learning, both generative/predictive models (Janner et al., 2019; Lee et al., 2019; Gregor et al., 2019) as well as discriminative models (Oord et al., 2018; Clavera et al., 2019). None of these methods can discriminate between useful and distracting information in the input and therefore have only been shown to work on environments that do not match the complexity of the real world.

Achille & Soatto (2017) discusses the separation of learning a minimal, sufficient representation and learning decision making from this representation. They underline that in order to generalize well, a representation should only capture information that is important for the task at hand and ignore all other input information. However, the authors only consider the single-task case while our work specifically focuses on learning a representation that can be used for solving arbitrary downstream tasks drawn from the same task *distribution* that was used for training the representation.

3 Approach

3.1 Preliminaries

We assume access to a pre-training dataset $\mathcal{D}$ of reward-annotated trajectories $\tau = \{s^t, a^t, r^t_{1:K}, s^{t+1}, a^{t+1}, r^{t+1}_{1:K}, \dots\}$, with states s^t, actions a^t and rewards $r^t_{1:K}$ for K tasks drawn from the task distribution $\mathcal{T}$. Note that we do not need to assume complete reward annotations from all K tasks on all $|\mathcal{D}|$ trajectories. Instead, a particular trajectory τ_i can have reward annotations from only a subset $\mathcal{T}' \subseteq \mathcal{T}$. For simplicity, we will assume complete reward annotation in the following, but the proposed model can be trivially extended to the incomplete case. Finally, we do not make assumptions on the exploration policy π_e that was used to collect the pre-training dataset other than that the resulting trajectories need to provide meaningful interactions with the environment.

3.2 Reward-Induced Representations

In this work we propose that reward-information can be used to induce representations that focus on information from the input that is useful for solving tasks from a downstream task distribution $\mathcal{T}$. To learn such a representation we train a reward prediction model $p_\theta(r_{1:K}|x)$ with parameters θ to infer the rewards from K different tasks given the input x. To extract a low-dimensional representation from this model, we factorize the prediction into an encoder $p_\phi(z|x)$ that is shared across tasks and compresses the input into a low-dimensional representation z, and K *reward-heads* $p_{\eta_k}(r_k|z)$ with separate parameters η_k that infer the respective reward from the shared representation:

$$p(r_{1:K}|x) = \prod_{k=1}^{K} \int p_{\eta_k}(r_k|z) \cdot p_\phi(z|x)\ dz. \tag{1}$$

We optimize the parameters $\theta = \{\phi, \eta_1 : K\}$ using a maximum-likelihood objective on the predicted rewards. Assuming a Gaussian output distribution with unit variance we can optimize the MSE loss on the predicted rewards $\hat{r}_k$ to learn the representation. For a single training trajectory from the pre-training dataset this takes the form:

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{k=1}^{K} ||r_k^t - \hat{r}_k^t||^2. \tag{2}$$

The above formulation in (1) represented the single-step inference case. It is however easily extendible to the sequence prediction case, where we predict T future rewards given N conditioning frames:

$$p(r_{1:K}^{1:T}|x^{-N+1:0}) = \prod_{t=1}^{T} \prod_{k=1}^{K} \int p_{\eta_k}(r_k^t|z^t) \cdot p_\phi(z^t|x_{-N+1:t-1})\ dz. \tag{3}$$

In practice we implement this recursive prediction with an RNN-based model and optimize with the same objective as in (2).

3.3 Reinforcement Learning with Pre-Trained Representation

The goal of this work is to show that reward-induced representations are useful to improve the efficiency of downstream skill learning. Following the standard reinforcement learning formalism (Sutton & Barto (2018)) we train policies to optimize the cumulative expected return on downstream tasks: $\arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^{T} R_t\right]$. To use the pre-trained representation we factorize the policy distribution, using the pre-trained encoder to translate inputs into the representation z:

$$\pi(a|x) = \pi'(a|z) \cdot p_\phi(z|x) \tag{4}$$

The optimization of this objective can be performed with any standard RL algorithm, for example value-based approaches (Lillicrap et al., 2015; Clemente et al., 2017; Haarnoja et al., 2018) or policy gradient methods (Schulman et al., 2015; 2017).

4 Experimental Evaluation

With our experimental evaluation we aim to answer the following questions: (1) Are reward-induced representations helpful for improving the sample efficiency of downstream tasks? (2) Are reward-induced representations more robust to visual distractors in the scene?

Because we are operating in deterministic environments we employ a deterministic version of our model. We instantiate the encoder and decoder with simple CNNs, the predictive model with a single-layer LSTM and all reward-heads with 3-layer MLPs. We use the Rectified Adam optimizer (Liu et al., 2019; Kingma & Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for pre-training the representation. For RL training, we use PPO (Schulman et al., 2017) with slight modifications to the default hyperparameters provided in the PyTorch implementation (Kostrikov, 2018).

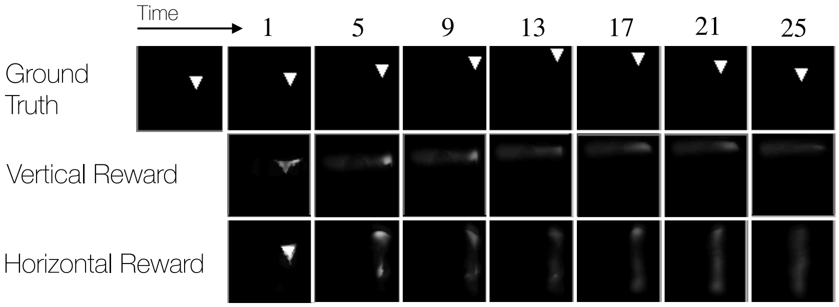


Figure 2: Detached decodings of reward-induced representations. **Top**: Ground truth sequence. **Middle**: Decoding of representation learned with reward proportional to the vertical coordinate. **Bottom**: Same, but with reward proportional to horizontal coordinate. Representations retain information about the position that influences the reward, but no information about the other coordinate, which leads to blurry predictions on the latter axis.

Environment. We perform preliminary evaluations on a simple dataset of geometric shapes bouncing in a quadratic frame of resolution 64×64. While initial position and velocity of all shapes are randomized, the trajectories themselves are deterministic. For simplicity we do not model interactions between the objects. The environment features an agent (circle shape), a target (square) and a variable number of distractor objects (triangles). For pre-training we collect random rollouts annotated with rewards proportional to the x/y position of both agent and target. During RL training we evaluate on a downstream task in which the agent should follow the target, i.e. minimize its L2 distance to the target. We use the following reward (where $\mathbf{p}_{\text{target}}$ and $\mathbf{p}_{\text{agent}}$ denote target's and agent's position):

$$R = 1 - \frac{1}{\sqrt{2}} \cdot \|\mathbf{p}_{\text{target}} - \mathbf{p}_{\text{agent}}\|_2 \tag{5}$$

Baselines. For the downstream RL experiments we compare to the following baselines:

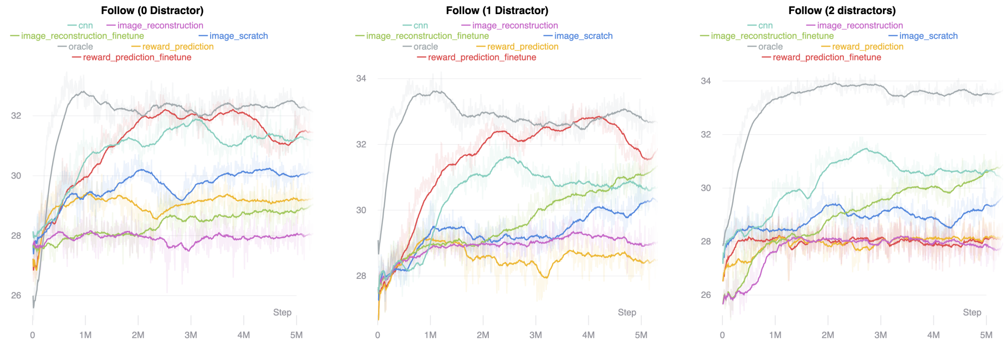
- **cnn** baseline uses a 3-layer CNN with 16 kernels of kernel size 3 and stride 2 to encode an image. ReLU is used as an non-linear activation. Then, the output activation is flattened and passed to two fully connected layers of 64 hidden states to compute an action and a critic.
- **image-scratch** baseline adopts the encoder architecture used in representation learning but we initialize the parameters randomly. The output embedding of size 64 is fed into two 2-layer MLPs of 32 hidden states, one for action and one for critic.
- **image-reconstruction** and **image-reconstruction-finetune** baselines use the learned representation on the image reconstruction task. Given the embedding the action distribution and critic are computed using 2-layer MLPs of 32 hidden states. We freeze the encoder for the baseline (**image-reconstruction**) and finetune the encoder for the baseline (**image-reconstruction-finetune**).
- **reward-prediction** and **reward-prediction-finetune** baselines use the learned representation on the reward prediction task. Given the embedding the action distribution and critic are computed using 2-layer MLPs of 32 hidden states. We freeze the encoder for the baseline (**reward-prediction**) and finetune the encoder for the baseline (**reward-prediction-finetune**).
- **oracle** takes a state representation as input. The state representation consists of (x,y)-coordinates of the agent, target, and distractors. The 2-layer MLPs of 32 hidden states are used to predict both action and critic. This method shows the upper bound of our method.

4.1 Analysis of Learned Representation

In a first effort to analyze the properties of reward-induced representations, we try to elicit what information is captured in the representation by training an image decoder to reconstruct the original image from the representation. Crucially, the gradients from the image decoder are stopped before the

Table 1: Reward regression MSE values for different representations with no, one and two distractors. A-X denotes reward proportional to the agent's horizontal position, T-Y indicates reward proportional to target's vertical position. The reward-induced representation enables more accurate reward regression and is also fairly robust to increasing amounts of noise while the image prediction based representation fails to capture information necessary to regress all rewards accurately.

NUM. DISTRACTORS	NONE				ONE				TWO			
METHOD	A-X	A-Y	T-X	T-Y	A-X	A-Y	T-X	T-Y	A-X	A-Y	T-X	T-Y
IMAGE PREDICTION	0.049	0.051	0.043	**0.005**	0.049	0.049	**0.007**	**0.007**	0.049	0.050	0.036	**0.007**
REWARD-INDUCED	**0.007**	**0.006**	**0.004**	0.004	**0.020**	**0.018**	0.006	0.006	**0.023**	**0.023**	**0.007**	**0.007**



(a) 0 Distractor (b) 1 Distractor (c) 2 Distractors

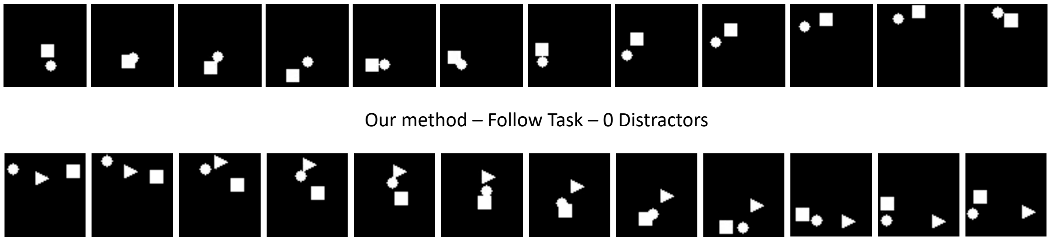
Figure 3: Reward curves on *Follow* tasks with 0, 1, and 2 distractors. Our approach (**reward-prediction-finetune**) achieves comparable results with the **oracle** method in tasks with 0 and 1 distractor.

representation, leaving the ladder unchanged by the probing network's training. The results shown in Fig. 2 for two simple rewards indicate that the reward-induced representation indeed only captures information about the input that are useful for inferring reward as the decoder is not able to infer the position of the object along the axis that has no influence on the reward used for training the representation.

To also quantify how much of the important information is captured in reward-induced representations compared to conventional, image-based representations, we report regression accuracy on position rewards for both agent and target in Tab. 1. We compare values for different numbers of visual distractors in the scene and find that reward-induced representations are better able to capture the important information in the scene across all scenarios. Further, they prove fairly robust to increased noise. Representation learned via image prediction objectives on the other hand are not able to capture all important information, leading to worse regression accuracy, because during training they receive no guidance on what is important to model and what can be ignored.

4.2 REWARD-INDUCED REPRESENTATIONS FOR REINFORCEMENT LEARNING

In Figure 3, we compare the speed of training and convergence across all the baselines discussed above. *Oracle* baseline acts over the most compact state representation containing shape positions and consequently performs the best. *reward-prediction-finetune* comes second both in speed and final convergence for environments with zero and one distractor. *cnn* learns slower than our method, but converges to similar values in the end. Other methods struggle to attain similar performance. This shows that reward prediction leads to a good encoder initialization for downstream tasks, and this is faster than a randomly initialized CNN encoder. It is much better than *image_scratch* which uses a similarly sized encoder as our method. Note that *cnn* performs better than *image_scratch* because of its small sized CNN network suitable for RL. Figure 4 shows qualitative rollouts of *reward-prediction-finetune* upon convergence on the follow task with 0 and 1 distractor.



Our method – Follow Task – 0 Distractors

Our method – Follow Task – 1 Distractor

Figure 4: Rollouts of a policy trained with reinforcement learning by finetuning an encoder pretrained with reward prediction. The agent (circle) successfully follows the target (square) even in the presence of distractors (triangles).

Similar performance cannot be attained with 2 distractors. This can be attributed to the reward prediction possibly overfitting during training. Therefore, the *oracle* method can be seen to perform very well, followed by the standard *cnn* policy. Interestingly, *image_reconstruction_finetune* also performs well on this task since it is able to maintain information about the agent, target and distractors, which can be extracted by the RL policy. However, this is still slower in comparison to the small CNN network based policy. While we can show that our method is more robust than baselines in the case of a single distractor, we hope to further improve our method on multiple distractors with better training schemes.

5 Discussion

We have shown that reward-induced representations are able to improve the learning efficiency of downstream reinforcement learning applications more efficiently than conventional, image-prediction-based representations. Further, we showed that reward-guidance improves the robustness to visual distractors in the scene, an important step towards the applicability of representation learning methods to the real world.

References

Alessandro Achille and Stefano Soatto. A separation principle for control in the age of deep learning. *arXiv preprint arXiv:1711.03321*, 2017.

Ignasi Clavera, Yiming Ding, and Pieter Abbeel. Mutual information maximization for robust plannable representations. *NeurIPS Workshop on Robot Learning*, 2019.

A. V. Clemente, H. N. Castejón, and A. Chandra. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862*, 2017.

Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. *arXiv preprint arXiv:1906.09237*, 2019.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Volodymyr Mnih and et al. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533, 02 2015.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

6 Implementation Training - ToDo

Using above descriptions, please complete the following tasks:

1. Implement the reward-induced representation learning model (see detailed architecture Figure 5). Train the model using the provided dataset.
2. Visualize your training results (learning curves etc) and verify that the model predicts correctly by replicating the results in Figure 2. Note, that you need to add a detached decoder network and train the model on a single reward only to reproduce the experiment.
3. Implement an RL algorithm of your choice to train an agent that can follow a target shape (while ignoring distractor shapes) in the provided environment. Before continuing to the next step, verify your implementation by training a policy that has access to the true state of the environment (i.e. does not need to encode images). This corresponds to the *oracle* baseline in Figure 3.
4. Train an image-based agent that encodes image observations using the pre-trained encoder. Compare its learning curve to that of an agent with the same architecture, but trained from scratch (*image-scratch* baseline in Figure 3).
5. If your implementation is working, (1) the image-based agent *with pre-training* should be able to follow the target shape with up to one distractor, and (2) it should learn faster than the image-based agent trained *from scratch* (but likely slower than the oracle).

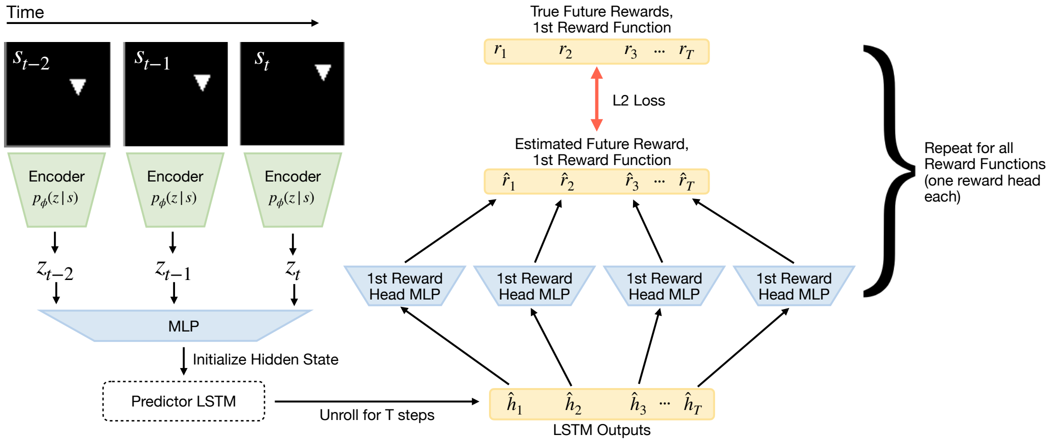

Time
s_{t-2}
s_{t-1}
s_t
Encoder $p_\phi(z|s)$
Encoder $p_\phi(z|s)$
Encoder $p_\phi(z|s)$
z_{t-2}
z_{t-1}
z_t
MLP
Initialize Hidden State
Predictor LSTM
Unroll for T steps
$\hat{h}_1$ $\hat{h}_2$ $\hat{h}_3$ ··· $\hat{h}_T$
LSTM Outputs
1st Reward Head MLP
1st Reward Head MLP
1st Reward Head MLP
1st Reward Head MLP
Estimated Future Reward, 1st Reward Function
$\hat{r}_1$ $\hat{r}_2$ $\hat{r}_3$ ··· $\hat{r}_T$
L2 Loss
True Future Rewards, 1st Reward Function
r_1 r_2 r_3 ··· r_T
Repeat for all Reward Functions (one reward head each)

Figure 5: Architecture of the reward prediction model (used for pre-training the representation z). All MLPs have 3 layers with 32 hidden units. The image encoder uses strided convolutions to reduce the image resolution by a factor of 2 in every layer, until the spatial resolution is 1x1 (i.e. the number of layers is determined by the input resolution). The number of channels gets doubled in every layer starting with 4 channels in the first layer. The final 1x1 feature vector gets mapped to a 64-dimensional observation space using a linear layer. The encoder (green) is transferred to the RL policy, where it is used to encode the image inputs.