Efficient RL via Disentangled Environment and Agent Representations

Kevin Gmelin [*1] Shikhar Bahl [*1] Russell Mendonca [1] Deepak Pathak [1]

Abstract

Agents that are aware of the separation between themselves and their environments can leverage this understanding to form effective representations of visual input. We propose an approach for learning such structured representations for RL algorithms, using *visual knowledge of the agent*, such as its shape or mask, which is often inexpensive to obtain. This is incorporated into the RL objective using a simple auxiliary loss. We show that our method, **S**tructured **E**nvironment-**A**gent **R**epresentations (SEAR), outperforms state-of-the-art model-free approaches over 18 different challenging visual simulation environments spanning 5 different robots.

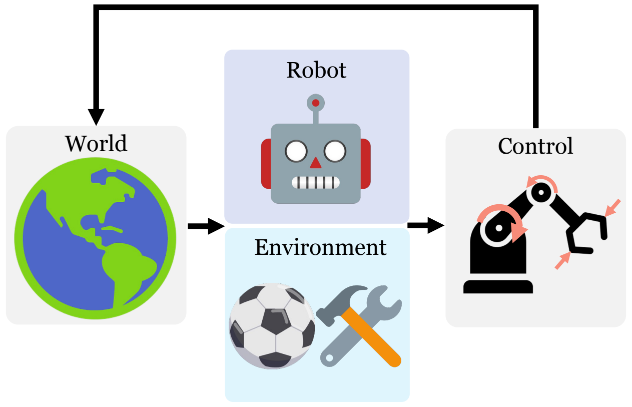

Figure 1: SEAR learns structured representations for visual control, by leveraging knowledge of the robot.

1. Introduction

Proprioception, i.e. the knowledge of one's own self, is heavily used by biological entities enabling them to perform various real-world tasks such as walking, manipulation, and navigation. Awareness of the position and movement of their body, and perceiving the environment as external to themselves, enables forming efficient representations of the observed input (Shapiro, 2010). In contrast, most contemporary methods in visual reinforcement learning (RL) often learn combined representations in an end-to-end manner (Kalashnikov et al., 2021; Arulkumaran et al., 2017; Kalashnikov et al., 2018; Levine et al., 2016; Peters et al., 2010) and require large amounts of data as a result. Inspired by the concept of the interface between the "inner" and "outer" environments, we study the following question: is there a natural way to build a representation that can disentangle a robotic agent from its environment, and does that improve learning efficiency for RL?

We argue that knowledge of the agent allows RL algorithms to focus on more interesting aspects of the visual input, such as objects in the environment. Thus, *by explicitly forcing the learning algorithm to disentangle agent-centric representations, we also implicitly create environment-centric ones*. This allows for visual RL approaches to not only learn faster but makes the representations interpretable. Consider a policy trained on a door-opening task with one type of robot. If it is aware of the agent-environment distinction, then it will be able to adapt a lot faster when deployed on a new robot, or a new task like opening a drawer, since it will not need to relearn visual appearance and it can focus just on dynamics. How can we incorporate this into an RL training setup?

Our solution is simple: *we augment the RL loss with an agent-centric auxiliary loss*. Naively doing so using proprioceptive data like joint angles or end-effector positions does not result in the desired representations, as these are not grounded in the visual observation space, and do not contain information about the robot's appearance. We find that the most grounded forms of supervision are agent *masks*. If some information is known about the agent, it is often quite practical to obtain (even rough) masks for agents, for example, by training a segmentation model.

Our approach, **S**tructured **E**nvironment-**A**gent **R**epresentations (SEAR), formulates the agent-environment representation learning problem as an auxiliary loss to the RL objective. This loss is the sum of the reconstruction loss of the agent mask and the full image as shown in Figure 2.

[*]Equal contribution [1]Carnegie Mellon University. Correspondence to: Kevin Gmelin <kgmelin11@gmail.com>, Shikhar Bahl <sbahl2@andrew.cmu.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

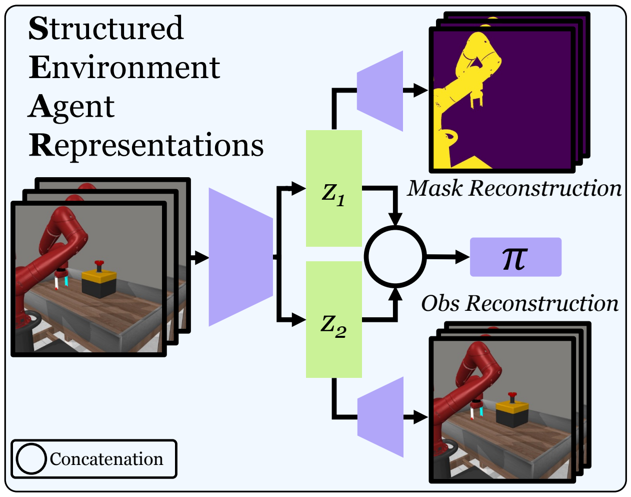

Figure 2: Overview of SEAR. We use the mask of the agent to learn structured representations.

We mathematically formulate the representation learning problem using variational inference and show that it leads to improved control.

Our approach is complementary to the prior works that aim to improve RL sample efficiency by adding auxiliary losses to the representations learned by the policy, drawing inspiration from advances in self-supervised learning in computer vision. This includes using inpainting losses (Pathak et al., 2016; He et al., 2022), contrastive learning (He et al., 2020; Chen et al., 2020; Chen & He, 2021; Grill et al., 2020; Laskin et al., 2020b), large-scale video pre-training (Nair et al., 2022) or simple yet effective image augmentation (Yarats et al., 2021a; Laskin et al., 2020a) to make the network more robust. Such implicit inductive biases allow for useful representations, but they don't take advantage of the fact that there exists a lot of untapped knowledge of the *agent*. Access to such knowledge is remarkably inexpensive as the agent morphology or joints are often known beforehand.

Empirically, we find that SEAR outperforms state-of-the-art model-free approaches for RL on a large suite of 18 different challenging environments, spanning 5 different robots, including the sawyer, franka and adroit-hand robots.

2. Related Work

RL from pixels Model-based approaches (Hafner et al., 2019; 2020; Ebert et al., 2017) learn an efficient latent space dynamics model to learn a policy for visual control tasks. Model-free algorithms such as RAD (Laskin et al., 2020a) and DrQ (Kostrikov et al., 2020) make use of image augmentations to provide additional inductive biases. Some methods have focused on improving the representations learned for visual control through auxillary tasks, such as contrastive learning (Laskin et al., 2020b). Perhaps most similar to our algorithm, SAC-AE (Yarats et al., 2021b) uses an autoencoder, in addition to losses from the critic in SAC (Haarnoja et al., 2018), to train an image encoder used for visual control. However, unlike our algorithm, SAC-AE (Yarats et al., 2021b) only uses a single decoder, and does not attempt to make any explicit distinction between agent and environment. DrQv2 (Yarats et al., 2021a) is a state-of-the-art model-free reinforcement learning algorithm for visual continuous control, using random shift image augmentations and n-step returns for improved sample efficiency.

Agent-Environment Centric Learning Previous approaches have directly structured the representation space using forward or inverse dynamics (Zhang et al., 2019; Watter et al., 2015), but these methods do not scale well to challenging image-based manipulation tasks. Hu et al. (2022) learn a factorized visual dynamics model, using the analytical forward kinematics of the robot and a learned world model. While this enables transfer of the world model to new robots with a similar action space, it doesn't exploit agent knowledge while training. Previous works have also implicitly trained policies aware of robot morphology, using the transferablility of the policies as signal (Yu et al., 2018; Duan et al., 2017; Dasari & Gupta, 2021; Finn et al., 2017). Some methods directly train separate robot and task modules, and attempt to transfer to new combinations of these modules (Neumann et al., 2014; Devin et al., 2017). It is also possible to construct a policy where each node is represented as a joint and each link is an edge (Wang et al., 2018; Huang et al., 2020), allowing for efficient generalization to new morphologies. Decoupling environment and agent learning has also been popular in exploration-based approaches (Parisi et al., 2021; Hu et al., 2022; Bahl et al., 2022; Mendonca et al., 2023). Unlike work which learns self-perception, we assume access to a self-perception module to provide access to robot segmentation masks, and then investigate how such a model can be incorporated to improve policy learning. As we show in Appendix D, such self-perception modules can be easily obtained.

3. Background

Reinforcement Learning Formally, the RL problem is defined by a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{S}_0, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s_{t+1}|s_t, a_t)$ is the unknown state transition function, $\mathcal{S}_0(s)$ is the initial state distribution, $\mathcal{R}(s_t, a_t)$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. An agent acts according to some policy $\pi(a|s)$ and the learning objective is to maximize the expected return, $\mathbb{E}_{s_t,a_t\sim\pi}\left[\sum_t \gamma^t \mathcal{R}(s_t, a_t)\right]$.

Representation Learning and RL Learning effective control policies directly from raw image observations without instrumented setups to detect the states of different objects in the world is quite challenging. One approach to address this is to learn low dimensional representations for control (Zhu et al., 2020; Nair et al., 2018; Yarats et al., 2021b; Laskin et al., 2020a), bringing the setting closer to state-based control. These approaches often model the state at each timestep as a latent z, and learn an encoder q_θ and decoder p_ϕ by maximizing the evidence lower bound on the log likelihood of the image X:

$$\log p(X) \geq \mathbb{E}_{z\sim q_\theta} p_\phi(X|z) - D_{KL}(q_\theta(z|X)||p(z)) \quad (1)$$

4. Structured Latents for Control

4.1. Learning Structured Latents

In the approaches described above, the learned latent z models the entire image, consisting of both the robot and the objects in the environment. Is there a way we can learn more structured latent representations, given access to visual information pertaining to the robot X_R? In addition to learning variable z which encodes the entire image, we also model z_R, which corresponds to salient information in X_R for control. z_R can be thought of as a processed version of z which only contains *agent*-relevant information, as opposed to information of the entire scene. To build the full dependency graph between variables X, X_R, z, z_R, we state the following desired property:

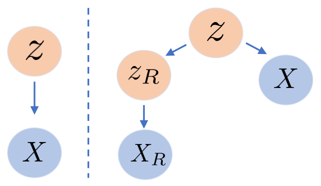

Figure 3: Graphical Models for prior approaches (left), vs SEAR (right).

Proposition 4.1. *If z effectively encodes X, then z_R and X should be conditionally independent given z, i.e* $p(z_R, X|z) = p(z_R|z).p(X|z)$

We enforce this variable dependency by implementing the graphical model in Figure 3. The joint probability distribution resulting from the model is :

$$p(X, X_R, z, z_R) = p(z).p(z_R|z).p(X|z).p(X_R|z_R) \quad (2)$$

In order to learn z, z_R, we maximize $J = \log p(X, X_R)$, and learn a variational approximation $q_\theta(z, z_R|X)$ to the posterior, which leads to the following lower bound (derivation in Appendix A.1):

$$\begin{aligned}\mathcal{L} = &\mathop{\mathbb{E}}_{z,z_R\sim q}[\log p(X|z)] + \mathop{\mathbb{E}}_{z,z_R\sim q}[\log p(X_R|z_R)] \\ &- D_{KL}(q(z, z_R|X))||(p(z, z_R))\end{aligned} \quad (3)$$

Breaking each term down, we first want to maximize $\mathbb{E}_{z\sim q_\theta} \log p(X|z)$. This is the standard reconstruction loss of the Variational Autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014). The second term, $\mathbb{E}_{z\sim q_\theta} \log p(X_R|z_R)$ reconstructs the provided robot visual information X_R from z_R, and the final term regularizes the posterior distribution just like in regular VAEs.

4.2. Effectiveness for Control

Given these structured latent representations, do they enable better sample efficiency and faster training for policies? We propose the following thought experiment as a starting point. Under the graphical model described in Proposition 4.1, consider a value function $V(z)$ that can, without loss of generality, be written as:

$$V(z) = V_R(z_R) + V_C(z) \quad (4)$$

V_R is a function only dependent on the agent (the robot in our case) and V_C is a coupling term for the value function, for the case where the agent is directly interacting with the environment. We argue that this formulation can represent a large class of value functions, especially under our disentangled representation space.

Our main insight is that in contact rich tasks, there are many uncertainties in modeling the environment. Furthermore, the first step in many such tasks is control of the robot to move it to some target location before it can then engage in contact. Thus, for a large percentage of $t \in (1, T)$, there will be a large emphasis on the first term of Equation 4, which is the value function dependent on the agent directly. By explicitly modelling z_R, we expect that it will be easier for the agent to learn basic robot control. After the agent can perform basic control of the robot, and capture some reward only relevant to the motion of the robot, it can then move on to manipulating the environment, which is better captured with the second term of Equation 4.

Overall, we hypothesize that modelling z_R enables agents to be much more efficient in learning policies, which is supported by our experimental analysis. We provide more details and analysis on this in Appendix A.2, and showcase this intuition in a toy experiment in Appendix A.3.

4.3. Visually Grounded Agent Representations

Now that we have seen how to learn agent-centric representations and that they are effective for learning policies, how do we obtain the agent-relevant observations X_R required for representation learning? While proprioceptive data such as end-effector positions or joint angles are easily accessible, they are not visually grounded, and may not convey the full information needed to model how the robot appears in the image observations. Thus, we try to predict the visual robot observation in the image. One approach for this is to use a segmentation model of the agent to obtain a robot mask. This is reasonable to obtain for robots since the shape and appearance of the robot does not change across tasks, and obtaining the mask model is a one-time cost. Furthermore, they do not need to be fully accurate, and we show that our method can still learn effective control policies when using noisy or approximate masks.

Given an image of the full scene X, we set X_R to be a segmentation of the robot, M, where $M_{i,j} = 1$ for every pixel that is occupied by the agent and 0 otherwise. Using a visual encoder $q_\theta(.)$, we get $y = q_\theta(X)$, which is then split into 2 vectors to obtain z_R and z. The encoder we use is similar to that of Yarats et al. (2021a). z_R is used as input to a decoder, $P_\phi(M|z_R)$ which tries to predict the robot mask. This decoder is trained using a binary cross-entropy loss, as shown in equation 5. Posing the agent-centric reconstruction as a classification problem allows for more efficient learning.

$$\mathcal{L}_{mask} = M \log P_\phi(M|z_R) + (1-M) \log (1 - P_\phi(M|z_R)) \tag{5}$$

Following Equation 3, z is used to approximate $p(X|z)$, by using a neural network decoder P_ψ, which tries to reconstruct the input image, X. This is trained using a mean squared error reconstruction loss, as shown in equation 6.

$$\mathcal{L}_{recon} = \|X - P_\psi(z)\|_2^2 \tag{6}$$

We empirically found that setting the last term of Equation 3 to be very low or 0 had much better results.

While we use DrQv2 from Yarats et al. (2021a) as our base reinforcement learning algorithm, it is possible to apply SEAR to any other reinforcement learning algorithm that utilizes a visual encoder, both for on or off-policy methods. We apply the same random shift to both the robot masks and input image observations as part of data augmentation. The mask decoder, reconstruction decoder, critic, and encoder are trained concurrently using a joint loss function, shown in Equation 7, with coefficients weighing the relative importance of the three component losses.

$$\mathcal{L} = \mathcal{L}_{critic} + c_1\mathcal{L}_{recon} + c_2\mathcal{L}_{mask} \tag{7}$$

The overall architecture is shown in figure 2. As with the DrQv2 algorithm, we do not let actor losses backpropagate into the encoder. Overall, SEAR introduces 4 additional hyperparameters to a reinforcement learning algorithm: two learning rates corresponding to two decoders, and two coefficients for the reconstruction and mask losses. To simplify hyperparameter tuning, we used the same learning rate as the critic for the two decoders. Our approach can be seen in Algorithm 1.

Algorithm 1 SEAR: Structured Environment and Agent Representations for Control

```
1: for t = 1 ... T do
2:     Collect transition $(x_t, m_t, a_t, R(x_t, a_t), x_{t+n})$
3:     $\mathcal{D} \leftarrow \mathcal{D} \cup (x_t, m_t, a_t, R(x_t, a_t), x_{t+1})$
4:     UPDATECRITICANDDECODERS($\mathcal{D}$)
5:     UPDATEACTOR (Yarats et al., 2021a)
6: end for
7: function UPDATECRITICANDDECODERS($\mathcal{D}$)
8:     $(x_t, M_t, a_t, r_{t:t+n-1}, x_{t+n}) \sim \mathcal{D}$
9:     Sample augmentation $A_1$
10:    $z_t \leftarrow q_\theta(A_1(x_t))$
11:    $[z_t^1; z_t^2] \leftarrow z_t$
12:    $\mathcal{L}_{mask} \leftarrow \mathcal{L}_{\text{ent}}(P_\phi(z_t^1), A_1(M_t))$
13:    $\mathcal{L}_{recon} \leftarrow ||P_\psi(z_t^2) - A_1(x_t))||_2^2$
14:    Compute $\mathcal{L}_{critic}$ (Yarats et al., 2021a)
15:    $\mathcal{L}_{total} \leftarrow \mathcal{L}_{critic} + c_1\mathcal{L}_{recon} + c_2\mathcal{L}_{mask}$
16:    Update $\theta_{enc}, \theta_{critic}, \theta_{mask}, \theta_{recon}$ using $\mathcal{L}_{total}$
17: end function
```

5. Experiments

In our experiments we seek to test the effectiveness of our structured representations in the following settings : 1) RL for manipulation with robot arms 2) RL for dexterous manipulation 3) Control in visually distracting environments 4) Transfer learning/finetuning. We also demonstrate some preliminary results in 5) multi-task learning. In this section, we describe our experimental setup, evaluations, baselines and results.

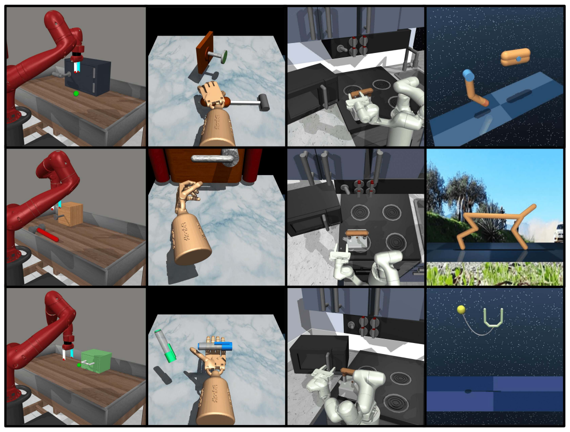

Figure 4: Environments used in the evaluation of SEAR. From left to right: Meta-World, Hand Manipulation Suite, Franka Kitchen, Distracting Control Suite

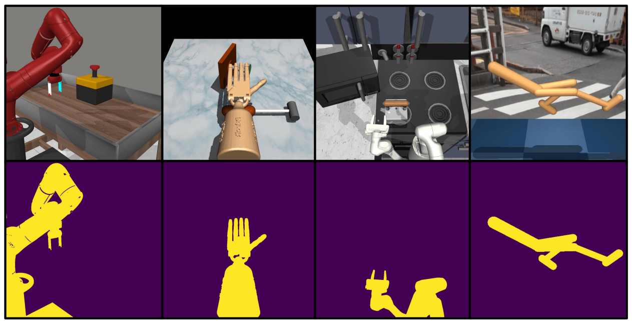

Figure 5: Environment RGB images with their corresponding robot segmentation masks.

Environments: We use the following Mujoco-simulated (Todorov et al., 2012) environment suites:

- `Meta-World` (Yu et al., 2020) - Table-top manipulation tasks performed by a Sawyer robot arm.
- `Franka Kitchen` (Gupta et al., 2019) - Manipulating objects in a realistic kitchen with a Franka arm.
- `Hand Manipulation Suite` (Rajeswaran et al., 2017) - Manipulating objects with an Adroit hand.
- `Distracting Control Suite` (Stone et al., 2021) - A variant of the DM Control suite (Tassa et al., 2018) with distractions added.

Note that we used implementations of the Hand Manipulation Suite and Franka Kitchen from the D4RL benchmark (Fu et al., 2020). Images of these four environments are shown in figure 4.

Setup: We obtain agent masks from the simulator directly. Ground truth segmentation masks were generated using Mujoco's rendering API, which has a flag for rendering segmented images. Pixel values from these segmentation renders correspond to geometric IDs. We then create binary masks based on geometric IDs known to correspond to the robot for a given environment. Example masks from each suite are shown in Fig. 5. Note that one could also use a segmentation model for these. The agent receives an RGB image as input, and a mask as supervision for the representation learning (all of which are 84x84). For each environment, a frame stack of 3 was used. For Meta-World and Distracting Control tasks, an action repeat of 2 was used, whereas an action repeat of 1 was used for Hand Manipulation Suite and Franka Kitchen tasks. In the Franka Kitchen setup, we used a sparse reward of 1 for each of the individual tasks. For the multi-task version, we add the sparse reward from each subtask achieved.

We compare to the following baselines in our experiments -

- `DrQ` - DDPG (Lillicrap et al., 2015) with image augmentations, shown to be state of the art in many visual RL settings. We use the code and setup from Yarats et al. (2021a), and implement SEAR on top of this base.
- `DrQ-AE` - Uses the same encoder as SEAR, but without the mask decoder. Has a single decoder operating on the entire latent vector in order to reconstruct the original input image. This is a version of SAC-AE (Yarats et al., 2021b). We run this to test if agent-environment disentangling is helping or if any form of reconstruction will have the same effect.
- `DrQ-MAE` - Reconstructing randomly masked patches in the images. Such methods have shown to be robust in self-supervised learning and control settings (Xiao et al., 2022; Radosavovic et al., 2022; He et al., 2022).
- `CURL` - State-of-the-art self-supervised visual learning for RL approach (Laskin et al., 2020b), which has been shown to be a useful auxiliary loss. Performs momentum contrastive learning (He et al., 2020) on the input images during the RL loop.

5.1. Analysis and Ablations

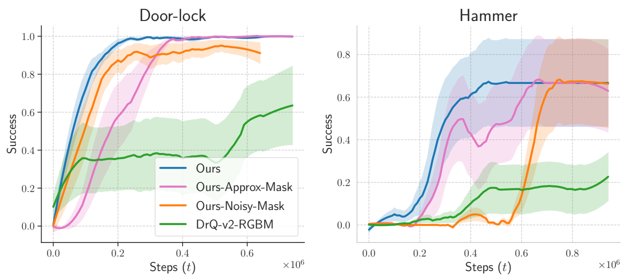

Figure 6: Analysis of robustness of SEAR with noisy and approximate robot masks.

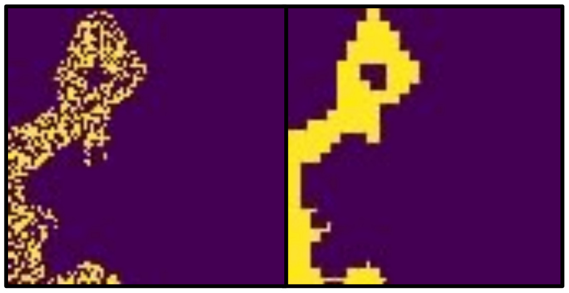

Figure 7: Noisy Mask (left) and Approximate Mask (right). Implementation described in Appendix C.6

Robustness to inaccuracies in robot masks: In real world scenarios, we do not have direct access to robot segmentation masks and may need to either have a trained

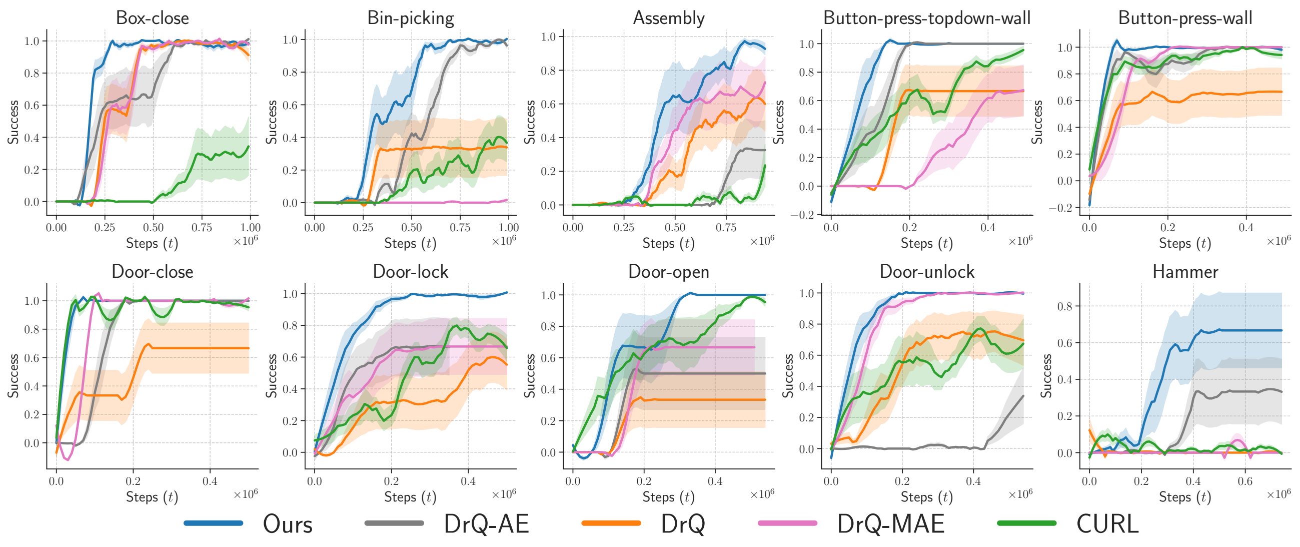

Figure 8: **Meta-World**: Success rate for different methods on 10 different tasks from the Meta-World suite. SEAR outperforms or matches prior approaches on all tasks.

model or use robot proprioception data. While SEAR does use additional information, which is the robot mask data, we argue that this information is not difficult to obtain or approximate. To test this, we run multiple ablations of our model and analyze these. Firstly, we confirm that adding even an approximate amount of agent information helps in the case of learning. This is done via two experiments shown in Figure 6. Agents were trained on two separate Meta-World tasks. We artificially introduce noise into the masks to evaluate the robustness of our approach (shown in orange). Another option is to use proprioceptive data to get an approximate mask. We mimic this by generating a large patch of pixels around the joints to get some mask that has a somewhat similar shape to the robot. This curve is shown in magenta in Figure 6. With both, we see that the performance does not decrease by much, almost being similar to SEAR with a perfect mask, even when a large amount of noise is added. Such masks can be seen in Figure 7. This experiment allows us to conclude that when training in the real world, a segmentation model of the robot can be used, even if it has inaccuracies. In Appendix D, we further qualitatively show that a simple masking model fine-tuned on 100 images (including those collected on our robots, and internet images) can give good robot masks for many out-of-distribution images.

Implicit Representations: Concluding that incorporating the robot information is important, we analyze other ways to learn z_R from the robot mask, for example in an implicit manner. Thus, we run a simple ablation (green) with `DrQ-RGBM`. This approach uses the DrQ-v2 algorithm trained on observations consisting of RGB images concatenated with robot masks. We see that this performance on both environments in Figure 6 is worse than that of our approach. We also see that SEAR is robust to hyperparameters such as c_1 and c_2, the loss coefficients for the reconstruction and mask (Appendix E). In Appendix F, we further show that there is a performance drop if the mask loss is added as an auxillary loss without splitting the latent vector.

5.2. Continuous Control Experiments

For the continuous control experiments, we evaluate SEAR on various challenging environments, tasks and benchmarks, where the agent varies greatly in morphology, ranging from a 7-DoF robot arm to a hand or an embodied two dimensional walker, all in MuJoCo (Todorov et al., 2012). We firstly evaluated our agent on 10 different visually challenging tasks from the Meta-World benchmark (Yu et al., 2020), as well as on three hand manipulation suite tasks, three Franka Kitchen tasks, and two Distracting Control tasks. We measure and report either episode success or total reward, at test time.

Meta-World We compared our agents performance, measured as episode success rate, to baselines on single-task manipulation tasks available in the Meta-World benchmark. Due to compute and time constraints, we only ran experiments on 10 tasks, which we chose because they seemed like the most contact-rich. The results are shown in figure 8. We observed greater sample efficiency on many of the tasks. These are all visually diverse and relatively challenging settings. Furthermore, on three of the tasks, SEAR achieved the highest episode reward by the end of training. On button-press-wall and door-close, SEAR did not outperform all the baselines, but it still matched the best ones. We found the

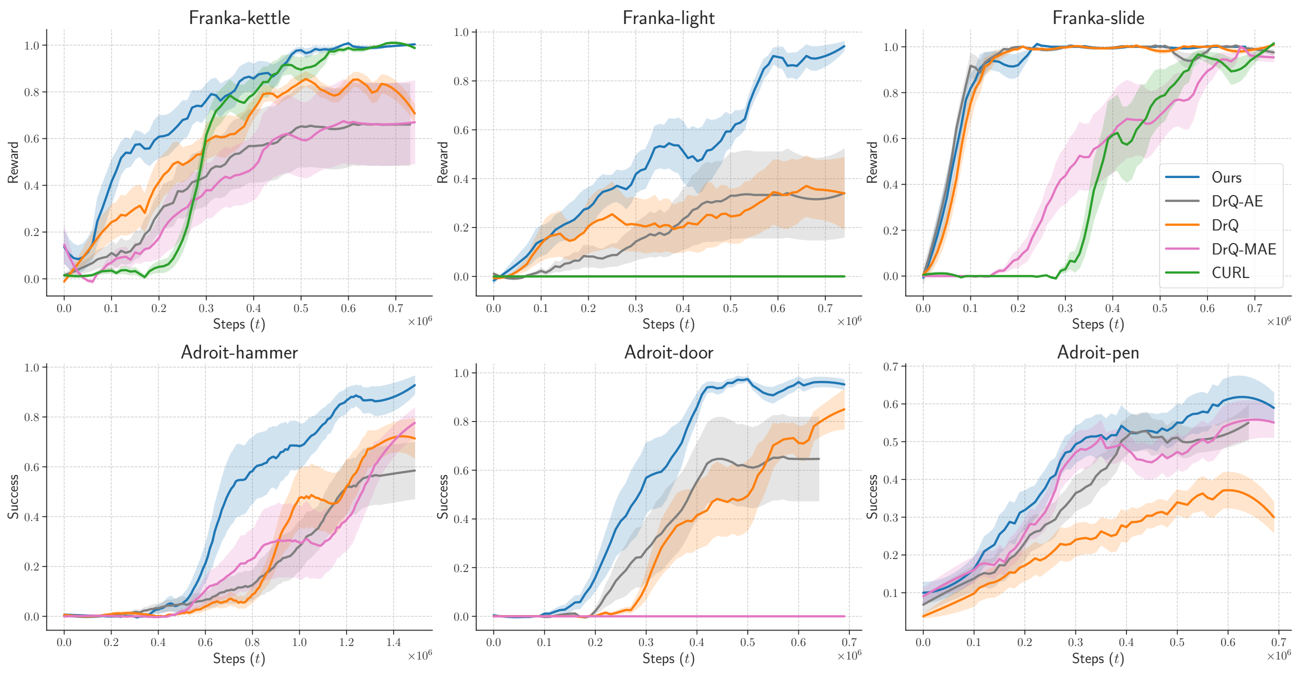

Figure 9: **Franka Kitchen & Adroit**: Success for different methods on 3 different tasks from the FrankaKitchen suite (top) and from the Adroit-hand suite (bottom). SEAR outperforms or matches prior approaches on all tasks.

auto-encoding baseline `DrQ-AE` to be a strong one, while contrastive learning (`CURL`) had difficulty. How does SEAR scale to different types of robots, for example, a Franka?

Franka Kitchen We evaluated our agent on the Franka Kitchen benchmark, a diverse and visually challenging environment, where a task is defined by a set of objects in the scene that have to be moved to pre-specified goal locations. We looked at three different, single item-tasks: kettle, light, and slider. A reward of 1 is given for successfully moving a task item to its pre-specified goal location. Results, given as total episode reward, are shown in figure 9. SEAR beat all baselines on light, and matched the best baseline on the other two tasks. Note that the best baseline was different between these two tasks (kettle and slide), suggesting that while there is high inter-task variance in the performance of the various algorithms, SEAR tends to match or surpass the performance of the best baseline for a given task.

Adroit Hand Manipulation We evaluated episode success rate of our agent on three different tasks within the Adroit Hand Manipulation Suite: Pen, Hammer, and Door. Results are shown in figure 9. SEAR outperformed baselines on two of the tasks, and matched the best baseline on the other task. We also ran experiments on the Relocate task, but all of the agents failed to learn, and thus we have not included it. We found that `CURL` was unable to solve any of the Adroit Hand Manipulation tasks, even with > 1.5M environment steps.

Distracting Control If our hypothesis of SEAR learning environment-agent disentangling representations was correct, we would see a strong performance in cases where control completely disentangles environment from agent. Thus we conducted an experiment on two challenging environments from the Distracting Control environments: (Stone et al., 2021) the ball-in-cup catch and walker-walk, where each episode, a random image is shown in the background. Total episode reward results are shown in Fig 10. By the end of training, SEAR achieved a higher total episode reward on both tasks compared to all other baselines. This suggests that SEAR learns useful disentangled representations when controlling a robot in visually distracting environments.

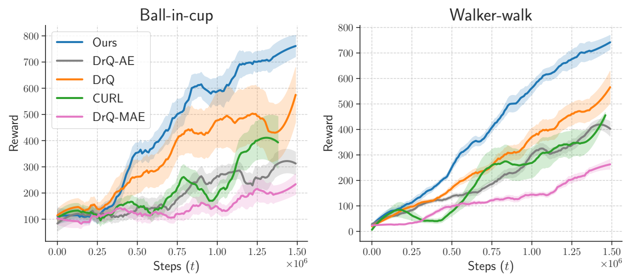

Figure 10: **Distracting Control:** Reward for different methods, SEAR outperforms all the baselines.

Transfer Learning In transfer learning environments, while the task may be different, the agent is often the same.

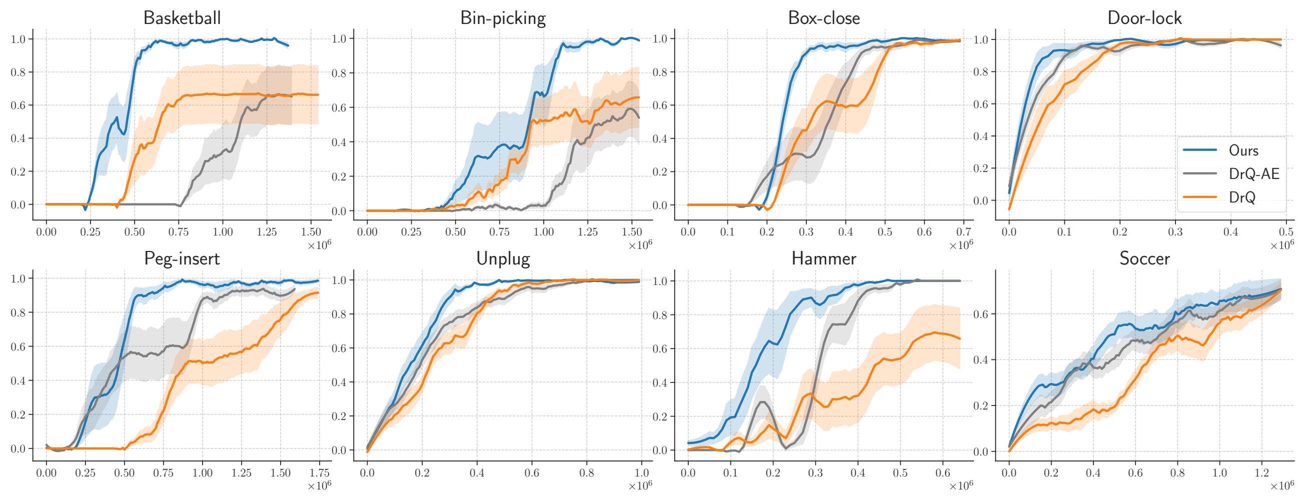

Figure 11: **Transfer**: Success for finetuning a multi-task policy pretrained on the Meta-World MT-10 suite to 8 different unseen tasks. SEAR transfers representations much better than prior methods.

This leads us to expect that making an agent-environment distinction in representation learning will lead to improved performance during transfer learning. To test this, we pre-trained agents on a multi-task setting (MT10 suite), and then fine-tuned on new Meta-World tasks not part of the original MT10 tasks. We also fine-tuned on Peg-Insert, which the agents failed to learn in the multi-task setting (Potentially affected by a poor camera angle). Results for transfer learning are shown in figure 11. SEAR outperformed the baselines on many of the tasks, and matched the best baseline on the rest. Overall, the results suggest that SEAR learns a representation that performs well in transfer learning settings. We leave it to future work to investigate to what extent transfer learning performance is attributable to better transferability vs the better performance that SEAR has shown in single-task learning.

Visualizing Feature Maps To gain more insight into the difference in representations learned by SEAR, we examine the activation maps of the encoder on one task each from three of our robotics environments. This activation map is the ReLU activations after the last encoder convolutional layer. Within the encoder, such activation maps would pass though an additional pooling and projection layer to form the latent vector. These activation maps are resized to the input image size, normalized to [0,1], and then blended with the original image after being mapped through a color map. We examine these activations since it is easier to visualize their spatial correlation with the input image, compared to visualizing the latent vector. Figure 12 contains a few example maps, where we can see that SEAR has more activity in regions around the robot in some maps and a lot more activity around objects in other maps. This is particularly salient in the Franka-Kitchen and Meta-World environments. In Appendix G, we present a comparison of all 32 channels for both SEAR and `DrQ`. While they don't show full disentanglement (since the linear projection can map them in arbitrary combinations), we found it interesting that activations from the SEAR encoder tend to have more filters that fire on either the robot or the environment (Figures 18 and 20), or focus on more diverse parts of the environment (kettle, burners, cabinet or microwave in Figure 19) compared to DrQ filters which only fire on the kettle. We hope to study this type of disentanglement in more detail in future work.

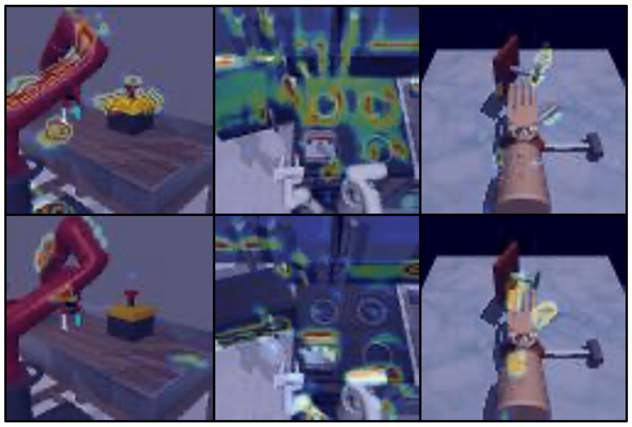

Figure 12: SEAR activation maps for select channels from the last convolution layer in the encoder of SEAR.

Multi-Task Learning Multi-task environments often use the same agent across multiple tasks. Thus, it is expected that making a distinction between agent and environment features will help multi-task learning as similar agent features can be shared across different tasks. To test this hypothesis, we utilized two small-scale multi-task environment

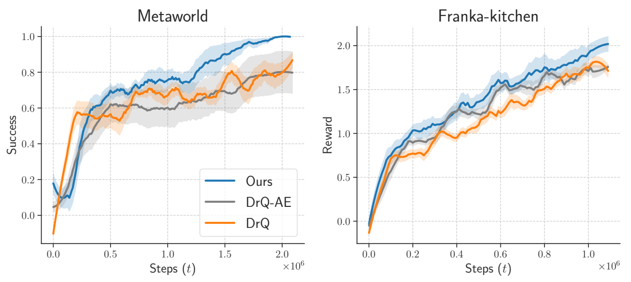

Figure 13: **Small-Scale Multi-task settings:** Success for different approaches for joint training on multiple tasks from an environment suite.

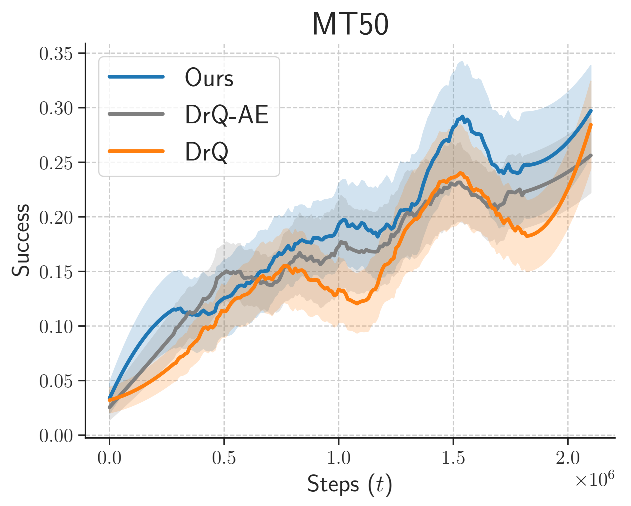

Figure 14: **Larger-Scale Multi-task setting:** Success for different approaches for joint training on the Meta-World MT50 set of tasks.

setups: a Meta-World setup with three tasks, and the Franka Kitchen environment with multiple task objects and randomized camera locations (shown in Appendix H). For each of these, the only input is the raw image. The three Meta-World tasks, button-press-topdown-wall, door-open, and hammer, were selected due to being three visually distinct tasks from our initial set of 10 contact-rich tasks. Results are shown in figure 13. SEAR outperforms baselines in the Meta-World setup and matches performance in the Franka-Kitchen environment. We further evaluate SEAR in a larger multi-task setup with the Meta-World MT50 set of tasks, with results shown in figure 14. On this larger set of tasks, SEAR matched but did not outperform the baselines. These results are preliminary, and we leave a full investigation into applying and adapting SEAR to multi-task environments as future work.

6. Discussion and Limitations

In this paper, we hypothesize that building representations which create a distinction between the agent and environment leads to more effective RL. We do so by formulating a representation learning method that can incorporate agent information directly and efficiently. We construct a learning algorithm, SEAR, that encourages representations to learn the agent-environment split by *reconstructing the mask* of the agent. We argue that this form of supervision is a very weak assumption, and in most cases is readily available (even in an approximate form). We show strong results against state of the art approaches in visual RL, on multiple different types of agents (robot arms, hands or even walkers), and in many different and challenging continuous control environments. In our future work, we hope to build agent-centric representations that can allow for building more efficient visual dynamics model, and empower exploration. Additionally, we would like to investigate disentangled representations for multi-task learning, and real-world robotics (a current limitation of SEAR).

Acknowledgement We would like to thank Alexander C. Li and Murtaza Dalal for fruitful discussions.

This work is supported by Sony Faculty Research Award and NSF IIS-2024594.

References

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. 1

Bahl, S., Gupta, A., and Pathak, D. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022. 2

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/chen20j.html`. 2

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021. 2

Dasari, S. and Gupta, A. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pp. 2071–2084. PMLR, 2021. 2

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017. 2

Duan, Y., Andrychowicz, M., Stadie, B., Ho, O. J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *NIPS*, 2017. 2

Ebert, F., Finn, C., Lee, A. X., and Levine, S. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017. 2

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. *CoRL*, 2017. 2

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 5

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 2

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 5

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 2

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 2

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 2

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *ICCV*, 2017. 16

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 5

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022. 2, 5

Hu, E. S., Huang, K., Rybkin, O., and Jayaraman, D. Know thyself: Transferable visual control policies through robot-awareness. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. 2

Huang, W., Mordatch, I., and Pathak, D. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020. 2

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018. 1

Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021. 1

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. 2

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020a. 2, 3

Laskin, M., Srinivas, A., and Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020b. URL `https://proceedings.mlr.press/v119/laskin20a.html`. 2, 5

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *JMLR*, 2016. 1

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 5

Mendonca, R., Bahl, S., and Pathak, D. Alan: Autonomously exploring robotic agents in the real world. *arXiv preprint arXiv:2302.06604*, 2023. 2

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *NeurIPS*, pp. 9191–9200, 2018. 3

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 2

Neumann, G., Daniel, C., Paraschos, A., Kupcsik, A., and Peters, J. Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8:62, 2014. 2

Parisi, S., Dean, V., Pathak, D., and Gupta, A. Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, 34: 20516–20530, 2021. 2

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

Peters, J., Mülling, K., and Altün, Y. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pp. 1607–1612. AAAI Press, 2010. 1

Radosavovic, I., Xiao, T., James, S., Abbeel, P., Malik, J., and Darrell, T. Real-world robot learning with masked visual pre-training. *arXiv preprint arXiv:2210.03109*, 2022. 5

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. 5

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014. 3

Shapiro, L. *Embodied cognition*. Routledge, 2010. 1

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021. 5, 7

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 5

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *IROS*, 2012. 5, 6

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. 2018. 2

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015. 2

Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 5

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a. 2, 4, 5, 14

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021b. 2, 3, 5

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. *RSS*, 2018. 2

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*, pp. 1094–1100. PMLR, 2020. 5, 6

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pp. 7444–7453. PMLR, 2019. 2

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., and Levine, S. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020. 3

A. Theoretical Justification

A.1. Representation Objective

We describe how we obtain the training objective for the representation space from the graphical model shown in Figure 3. In order to learn z, z_R, we maximize $J = \log p(X, X_R)$, and learn a variational approximation $q_\theta(z, z_R|X)$ to the posterior. We have :

$$\begin{aligned}
\log p(X, X_R) &= \log \int_{z,z_R} p(z).p(z_R|z).p(X|z).p(X_R|z_R) \\
&\geq \int_{z,z_R} \log(p(z, z_R).p(X|z).p(X_R|z_R)) \\
&= \mathop{\mathbb{E}}_{z,z_R\sim q}\left[\log\left(\frac{p(z, z_R).p(X|z).p(X_R|z_R)}{q(z, z_R|X)}\right)\right] \\
&= \mathop{\mathbb{E}}_{z,z_R\sim q}[\log p(X|z)] + \mathop{\mathbb{E}}_{z,z_R\sim q}[\log p(X_R|z_R)] - D_{KL}(q(z, z_R|X))||(p(z, z_R))
\end{aligned}$$

A.2. RL Performance

Consider the general value function described in Section 4:

$$V(\boldsymbol{z}_t) = V_R(\boldsymbol{z}_t^R) + V_C(\boldsymbol{z_t}) \tag{8}$$

Here z_t is the concatenation of $\boldsymbol{z}_t^R$ and $\boldsymbol{z}_t^E$. We can think of the following

$$\begin{aligned}
\boldsymbol{z}_t^R &\sim \mathcal{N}(\mu_R, \sigma_R) \\
\boldsymbol{z}_t^E &\sim \mathcal{N}(\mu_E, \sigma_E)
\end{aligned}$$

Let us assume two cases, one where there is environment interaction and one where there is none. In the case where there is very little or no interaction, we can think in this case that σ_R will be small, as there is less noise in the robot information, as these features are more salient for the value function. Assuming that the value function can be written as a quadratic function:

$$\begin{aligned}
V(\boldsymbol{z}_t) &= [\boldsymbol{z}_t^R, \boldsymbol{z}_t^E]^T V[\boldsymbol{z}_t^R, \boldsymbol{z}_t^E] \\
V(\boldsymbol{z}_t) &= [\boldsymbol{z}_t^R, \boldsymbol{z}_t^E]^T [V_1, V_2][\boldsymbol{z}_t^R, \boldsymbol{z}_t^E] \\
V(\boldsymbol{z}_t) &= (\boldsymbol{z}_t^R)^T V_1 \boldsymbol{z}_t^R + (\boldsymbol{z}_t^E)^T V_1 \boldsymbol{z}_t^E + 2(\boldsymbol{z}_t^R)^T V_2 \boldsymbol{z}_t^E
\end{aligned}$$

When fitting these parameters, due to the high noise in $\boldsymbol{z}_t^E$, we find that singular values of V_1 are much higher than those of V_2.

Now let us look at the observation X. This can be thought of as:

$$X = Q\boldsymbol{z}_t + \epsilon \tag{9}$$

Where Q is some matrix in $SO(3)$ and $\epsilon \sim \mathcal{N}(0, 1)$. When learning $\boldsymbol{z}_t$ from these, in the case of an *autoencoder*, we will fit $\hat{\boldsymbol{z}}$ to the top principal components of $\boldsymbol{z}_t^E$, as $\sigma_E > \sigma_R$. However, in the case where we are explicitly reconstructing X_R, the model will fit $\hat{\boldsymbol{z}}$ to a mix of top principal components of $\boldsymbol{z}_t^R$ and $\boldsymbol{z}_t^E$, which means it is better at recovering the true $\boldsymbol{z}_t$.

A.3. Toy Experiment

To showcase the intuition that it can be easier to learn the value function/policy if the latent representation directly decouples the agent and environment state, we construct the following toy experiment. We create a 2D continuous point mass environment with a changing obstacle position and a randomly sampled goal and initialization position. The ground truth reward is made up of an attraction term (exponential distance to goal) and a repulsion term (exponential distance to obstacle). The agent does not observe the ground truth state, but a high dimensional projection of a noisy latent state as mentioned in equation 9. Furthermore, as in Appendix A.2, $\boldsymbol{z}_t^R$ and $\boldsymbol{z}_t^E$ are drawn from normal distributions, where μ_R is the position of the agent and μ_E contains the position of the goal and the position and size of the obstacle.

When running SEAR, the agent additionally has access to the mask : $X_R = Q_2 z_R$.

We collect data in this environment to learn $\hat{z}$ from X and $\hat{z}_R$ from X_R, which learn the overall structure of the data and the agent-relevant structure respectively. We use this to train a predictive reward model: $V(\hat{z}, \hat{z}_R)$, which is later used for MPC. We test the performance of SEAR which models $(\hat{z}, \hat{z}_R)$, to a baseline which directly models $\hat{z}$, using final distance to goal as the metric, for different relative values of σ_R and σ_E, presenting the results in Table 1.

Table 1: Final distance to goal for varying relative values of agent and environment observation noise

	$V(\hat{z})$	$V(\hat{z}, \hat{z}_R)$
$\sigma_E > \sigma_R$	1.35	1.09
$\sigma_E = \sigma_R$	1.09	1.08
$\sigma_E < \sigma_R$	1.40	1.36

Table 1 shows the distance to goal achieved for different noise levels by using $V(\hat{z})$ or $V(\hat{z}, \hat{z}_R)$ for planning. We see that when the noise in the environment observation is high, SEAR provides much more effective control, and that the performance is roughly equal in all other cases.

B. Limitations and Broader Impacts

SEAR has a couple of different limitations. First, it's performance on learning representations for multi-task learning is not fully clear. We showed preliminary results on two small-scale multi-task settings and the Meta-World MT50 setting, but further investigation is required. SEAR has not yet been tested on real-world environments.

SEAR was only tested with the DrQv2 algorithm, and so while SEAR can be deployed with any reinforcement learning algorithm with an image encoder, future work still needs to explore the performance of using SEAR with other algorithms.

SEAR does not provide any benefit over other approaches in cases where the robot is not visible in the input image. Thus, future work will need to investigate how to incorporate such visual observations, which may be common in cases where, for example, the robot has a first-person view.

Another limitation of SEAR is that the representations learned by SEAR may not be closely aligned with human interests. This becomes more relevant as visual control algorithms are deployed in greater numbers to perform a wider variety of tasks, especially for tasks with a direct impact on humans.

Even if the representations learned are relatively well-aligned with the intentions of a designer using this algorithm, there are no safe-guards against a designer purposefully using SEAR to learn visual control tasks that can cause harm to humans.

SEAR could also be used to aid the deployment of more general-purpose robots, which may affect people's employment and overall economic conditions.

C. Implementation Details

Here, we present additional details relevant to our implementation of SEAR.

C.1. Code

The code of ours can be found at `https://sear-rl.github.io`

C.2. Hyperparameters

For SEAR, we used the DrQv2 algorithm (Yarats et al., 2021a) with a modified encoder, and added additional mask and reconstruction decoders. A list of hyperparameters can be found in Table 2. Most hyperparameters have the same value as in (Yarats et al., 2021a).

Compared to the original replay buffer size of 1e6, we used a smaller replay buffer size of 2.5e5. This change was made mainly to reduce RAM usage for each experiment. For instance, a buffer of size 1e6 with a frame stack of 3 corresponds to 3e6 rgb images of size (84, 84). Given 3 bytes per pixel, this gives us a total RAM usage of about 63.5 GB. This problem could be alleviated through tricks such as only storing each image once, instead of duplicating the image 3 times for a frame stack of 3. However, we did not see any performance decrease during preliminary experiments when reducing the replay buffer. We used this smaller replay buffer size as well when evaluating baselines.

Another difference in value between hyperparameters common to both DrQv2 and SEAR is action repeat. On the Franka Kitchen and Hand Manipulation Suite environments, we set the action repeat to 1. We did not experiment with varying the action repeat, but we did use the same action repeat in our comparison baselines.

SEAR has a couple of extra hyperparameters that are not present in DrQv2. One such hyperparameter is the learning rate for the two decoders. For simplicity, we ended up using the same learning rate as the actor and critic. The other new hyperparameters are the coefficients controlling the contribution of the reconstruction and mask decoders to the total loss, relative to the critic loss. These correspond to coefficients c_1 and c_2 in equation 7. We observed that the cross entropy mask loss tended to converge to a value about four times greater than the reconstruction loss, so we set the reconstruction loss scaling coefficient to be four times the mask loss scaling coefficient. This kept the contribution of the decoders relatively equal. From the ablation shown in 17, SEAR is relatively insensitive to the choice for c_1.

C.3. Architecture

Our architecture differs from (Yarats et al., 2021a) with the addition of two decoders, and a modification to the encoder. The architectures used for the actor and critic are the same.

Similar to (Yarats et al., 2021a), our encoder architecture contains a series of four convolutions with ReLU's between them. Each convolution has a kernel size of 3, and every convolution except the first one has a stride of 1. The first convolution has a stride of 2. Every convolution has 32 feature maps. This transforms a stack of 84x84 rgb images into a 32x35x35 set of activation maps. After the convolutions, we add a 2D average pooling layer with kernel size of 4 and stride of 4. This helps reduce the overall dimensionality down, giving us 32x8x8 features. We flatten this set of feature maps, and pass this through a linear layer, projecting the activations to a desired latent dimension. The encoder outputs the output from this linear layer.

The decoder architecture, used for both the mask and reconstruction decoders, starts with a linear layer with ReLU activation. This maps the latent vectors to a 2048 dimensional vector, which we reshape into a 32x8x8 set of feature maps. We pass these feature maps through 5 transpose convolutions. The first transpose convolution in the decoder is set up to mirror the average pooling layer in the encoder. It has a kernel size and stride of 4. We used 16 channels for this convolution, as our original encoder implementation had no linear projection, and so each half of the latent vector passed to each decoder only came from 16 of the 32 channels output by the average pooling layer. After adding a linear projection to the encoder, each half of the latent now comes from a combination of all 32 channels output by the average pooling layer, but we kept using 16 channels in the first layer of our decoder. Future work could change the first transpose convolution to use 32 channels, although we do not expect such a change to make any significant impact on the overall performance of SEAR. The last four transpose convolutions mirror the 4 convolutions in the encoder. They have 32 channels, except for the last convolution, which has the same number of channels as the stack of rgb images for the reconstruction decoder, or the same number of channels as the stack of masks for the mask decoder. They have a kernel size of 3, and a stride of 1, except for the last transpose convolution which has a stride of 2. The output from the transpose convolutions is passed through an output activation, which is the identity for the reconstruction decoder and a sigmoid for the mask decoder.

C.4. Training Details

We train each model using a mix of RTX6000, A5000, A6000 or 2080Ti GPUs. Each run takes about 3-7 hours of training on these.

Table 2: Hyperparameter Settings

Hyperparameter	Value
Replay Buffer Size	2.5e5
Frame Stack	3
Action Repeat	2 - Meta-World; Distracting Control
	1 - Kitchen; Hand Manipulation
Seed Frames	4000
Exploration Steps	2000
N-Step Return	3
Batch Size	256
Discount	0.99
Optimizer	Adam
Actor, Critic Learning Rate	1e-4
Decoder Learning Rate	1e-4
Agent Update Frequency	2
Critic Q-function Soft-Update Rate τ	0.01
Actor, Critic Feature Dimensions	50
Latent Dimensions	4096 - Single-task
	512 - Multi-task, Transfer
Exploration StdDev. Schedule	linear(1.0,0.1,2e6) - Meta-World MT
	linear(1.0,0.1,5e5) - otherwise
Exploration StdDev Clip	0.3
Observation Render Size	(84, 84)
Reconstruction Loss Scaling Coefficient (c_1)	0.01
Mask Loss Scaling Coefficient (c_2)	0.0025
Camera Name/ID	”Corner” - Meta-World
	”Fixed” - Hand Manipulation; Kitchen
	0 - Distracting Control
Evaluation Frequency	1e4
Evaluation Episodes	10

C.5. Robot Mask Preprocessing

For segmentation masks, single pixel wide artifacts were observed to render around the edges of some non-robot objects. At a resolution of 84x84, some robot features were also on the order of a single pixel wide, which made filtering at such a resolution more difficult. We observed that the artifacts remained one-pixel wide, even as the resolution was increased. Thus, to filter out these artifacts, segmentation masks were rendered at a resolution of (252, 252), a morphological opening was applied, and then the masks were downsampled to a resolution of (84, 84) before being passed to the agent.

C.6. Noisy and Approximate Mask Generation

Noisy masks were generated by taking the clean, filtered robot mask, and randomly setting robot labels to non-robot labels with a user-specified probability. To generate an approximate mask, we take the original robot mask, downsample it, and then upsample it back to its original size. We then apply gaussian blurring and threshold the image to get a new mask.

D. Real-World Robot Segmentation Model

In order to more directly test applicability to real robots, we train a segmentation model for a real Franka Panda robot. For this setup, we finetune a Mask-RCNN model (He et al., 2017) on around 100 images of our own robot (Franka Panda) as well as a few internet images, both of which we manually label. Figure 15 shows images of the masks obtained by running our model on out-of-distribution robot images as well as new viewpoints. When paired with the results from 5.1, which show that SEAR can work with noisy and approximate robot masks, we have hope that SEAR may also be applicable to training in real-world settings.

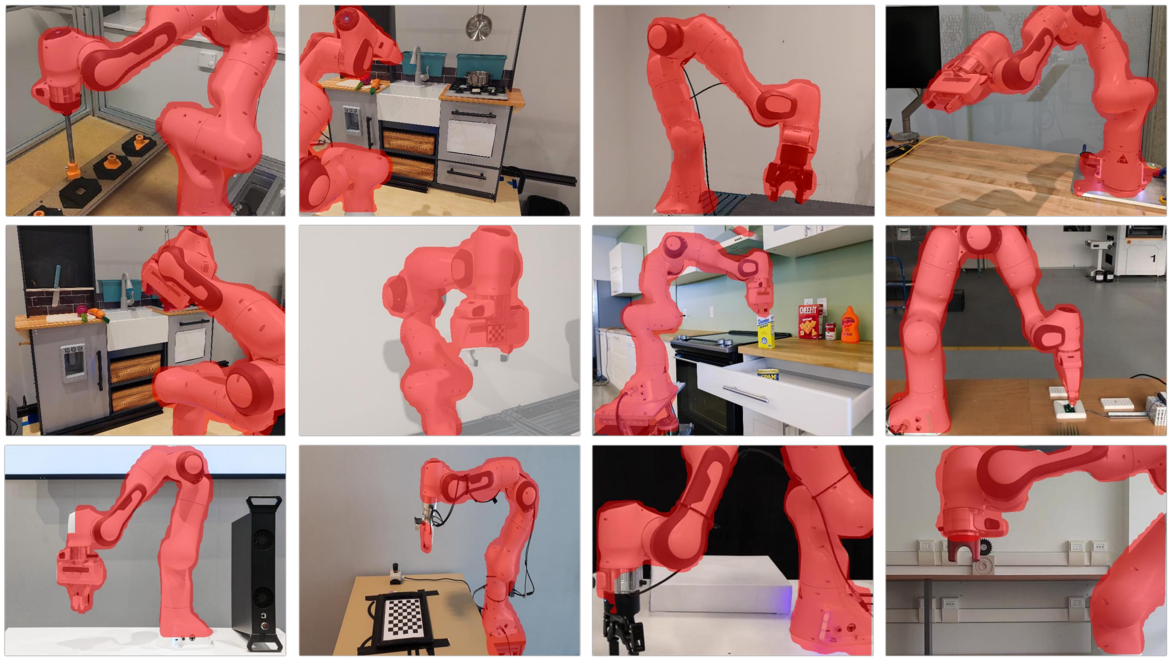

Figure 15: Segmentation masks predicted on out-of-distribution images by a fine-tuned Mask-RCNN model (He et al., 2017).

E. Decoder Ablations

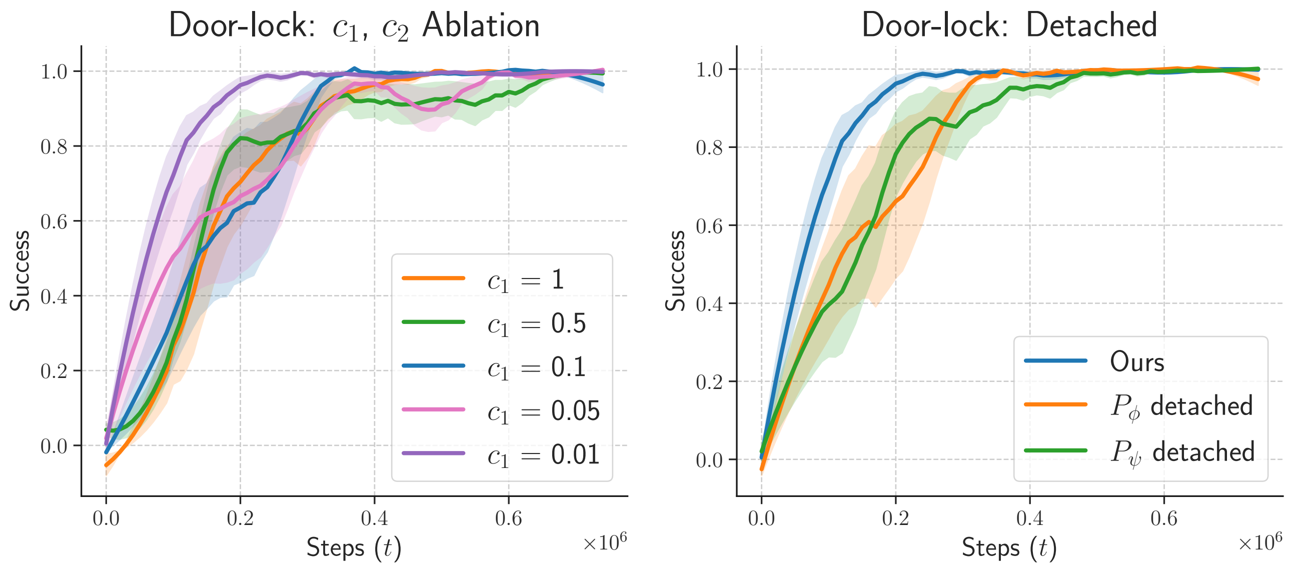

Figure 16: Ablation of hyperparameters of SEAR such as the coefficients of the reconstruction loss: c_1, c_2.

F. Auxillary Mask Loss Without Latent Separation

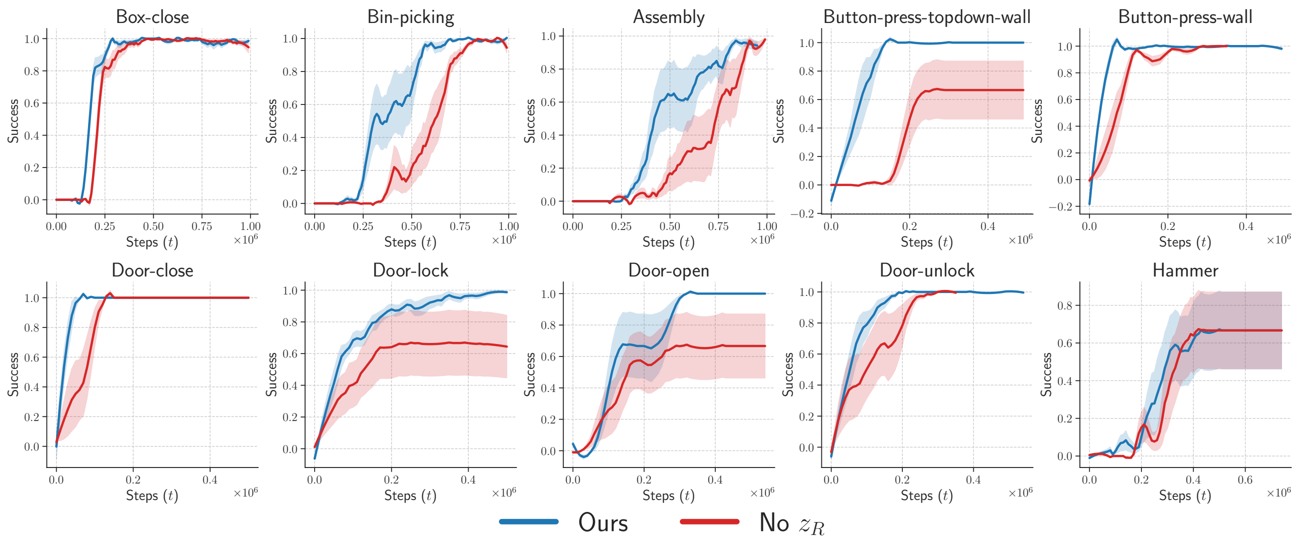

Figure 17: Ablation comparing SEAR to a model where you add an auxillary mask loss but do not split the latent vector.

G. Encoder Activation Maps

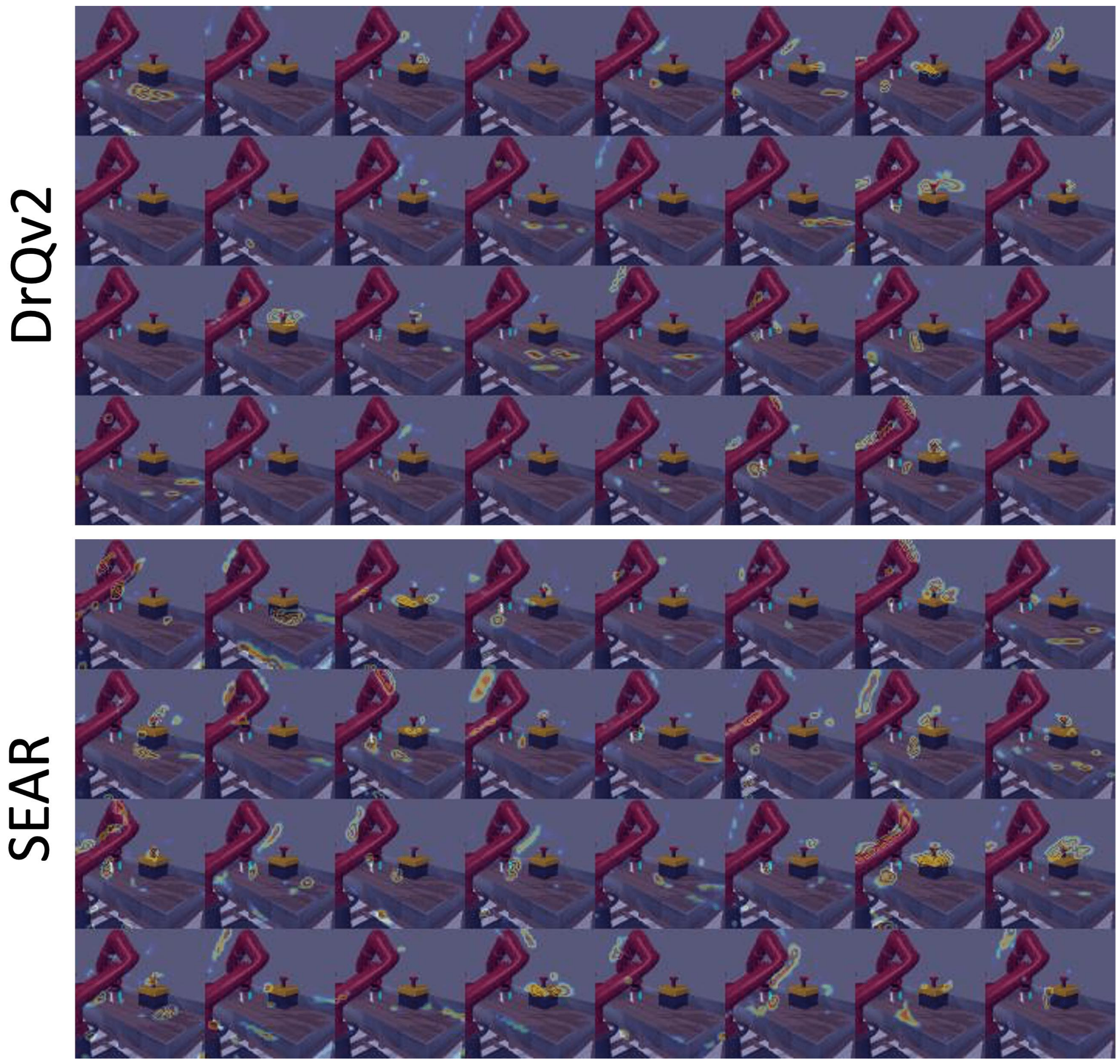

Figure 18: Activation maps from the fourth convolution of the encoder, resized and overlaid on top of the corresponding input image.

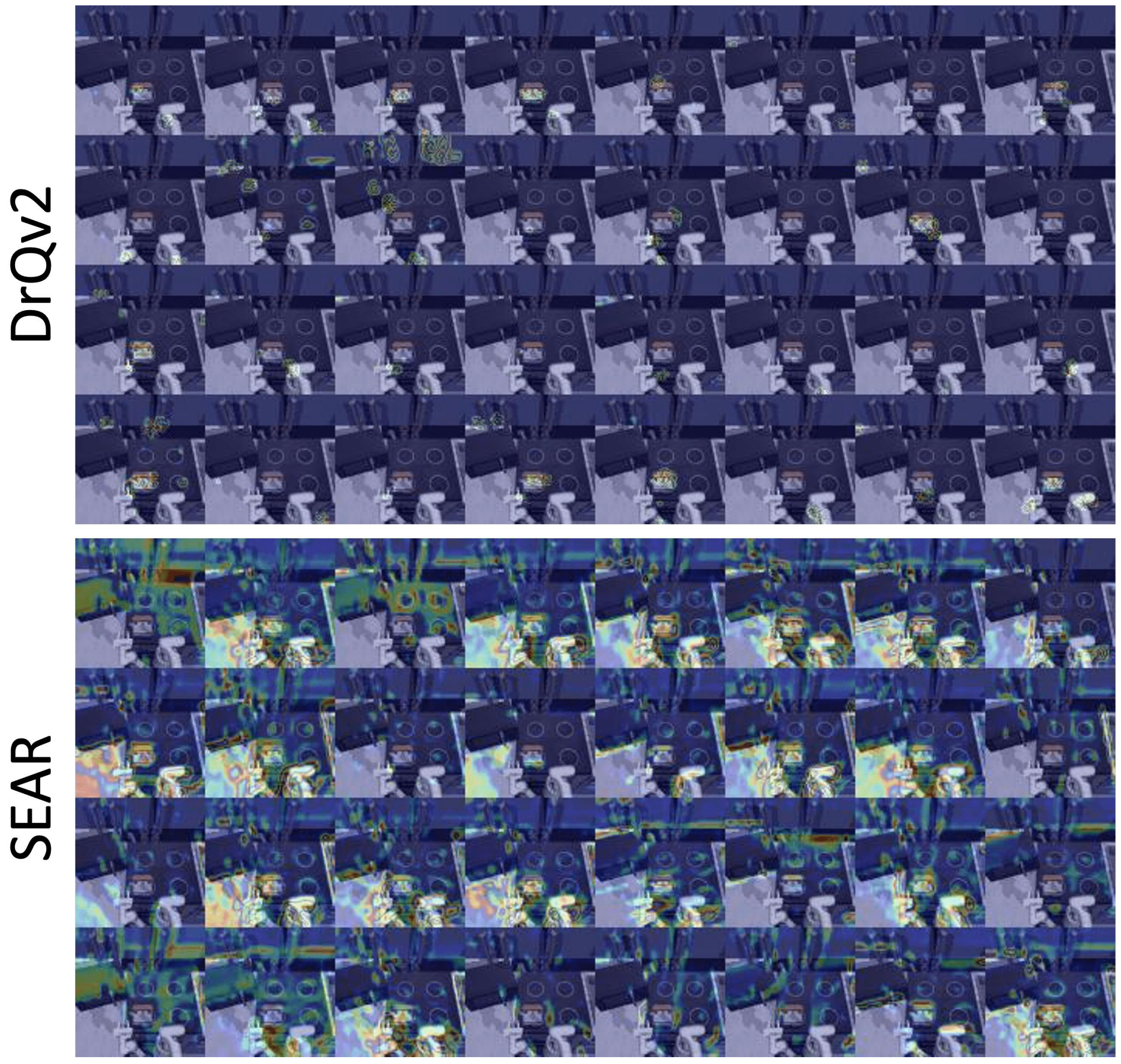

Figure 19: Activation maps from the fourth convolution of the encoder, resized and overlaid on top of the corresponding input image.

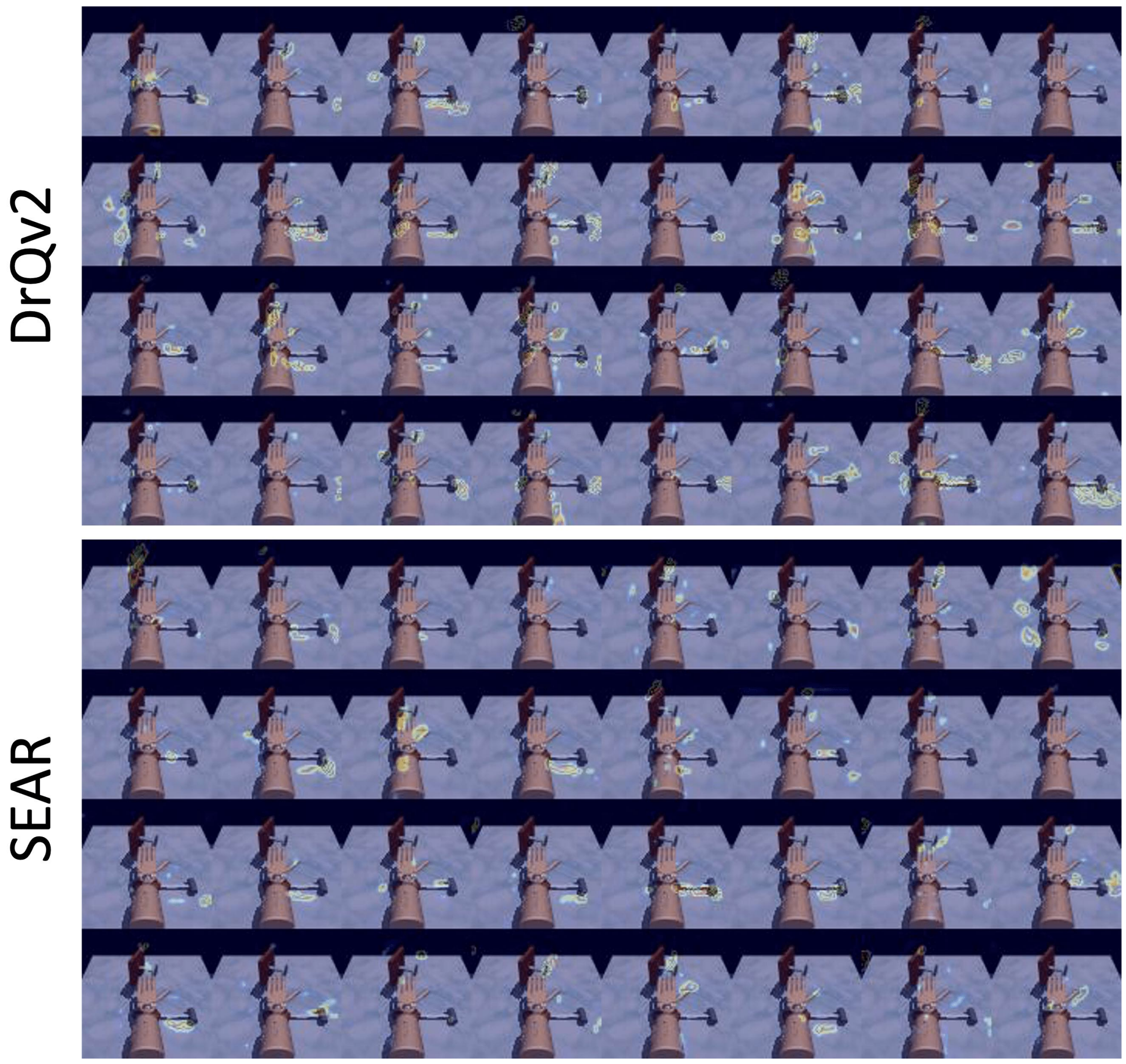

Figure 20: Activation maps from the fourth convolution of the encoder, resized and overlaid on top of the corresponding input image.

H. Multi-Task Franka Kitchen Camera Locations

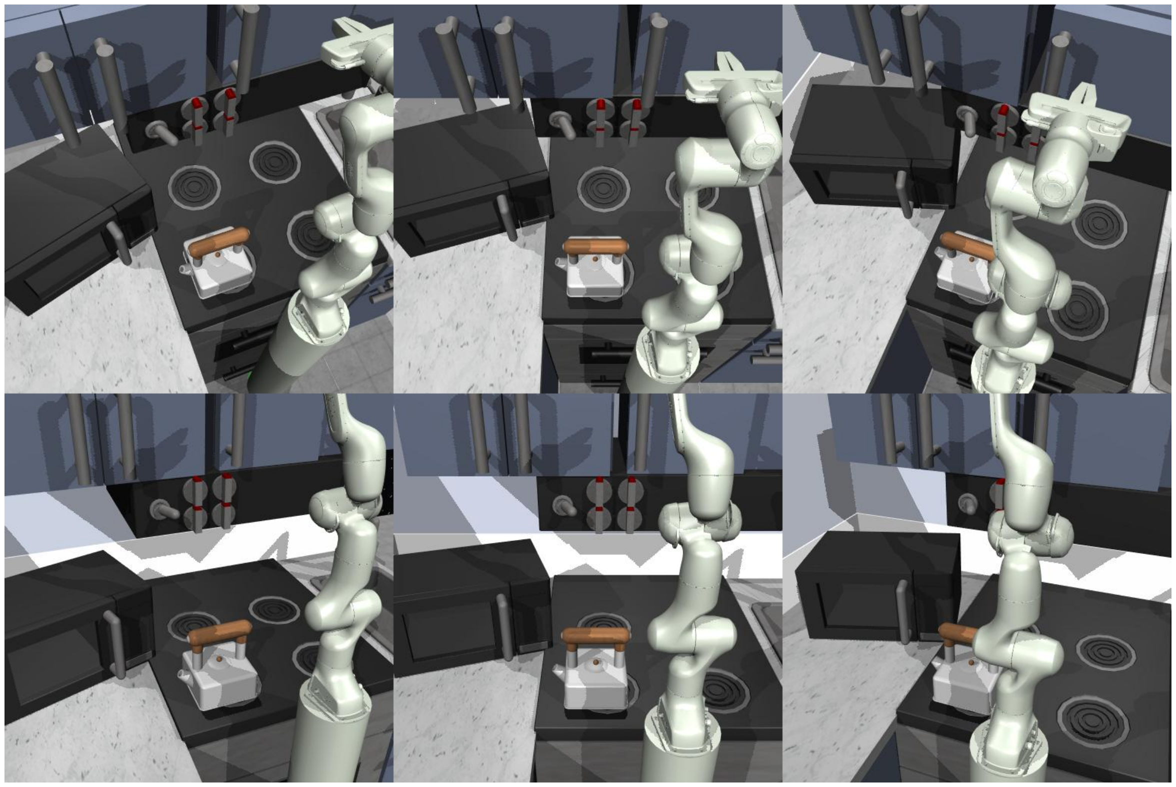

Figure 21: Set of 6 camera angles randomly selected from each episode in the Franka Kitchen multi-task setup