Multi-view Disentanglement for Reinforcement Learning with Multiple Cameras

Mhairi Dunion
mhairi.dunion@ed.ac.uk
University of Edinburgh

Stefano V. Albrecht
s.albrecht@ed.ac.uk
University of Edinburgh

Abstract

The performance of image-based Reinforcement Learning (RL) agents can vary depending on the position of the camera used to capture the images. Training on multiple cameras simultaneously, including a first-person egocentric camera, can leverage information from different camera perspectives to improve the performance of RL. However, hardware constraints may limit the availability of multiple cameras in real-world deployment. Additionally, cameras may become damaged in the real-world preventing access to all cameras that were used during training. To overcome these hardware constraints, we propose Multi-View Disentanglement (MVD), which uses multiple cameras to learn a policy that is robust to a reduction in the number of cameras to generalise to any single camera from the training set. Our approach is a self-supervised auxiliary task for RL that learns a disentangled representation from multiple cameras, with a shared representation that is aligned across all cameras to allow generalisation to a single camera, and a private representation that is camera-specific. We show experimentally that an RL agent trained on a single third-person camera is unable to learn an optimal policy in many control tasks; but, our approach, benefiting from multiple cameras during training, is able to solve the task using only the same single third-person camera.

1 Introduction

The ability of a Reinforcement Learning (RL) agent to learn an optimal policy on robotic control tasks from images depends on the position of the camera available during training. Often, a static third-person camera pointing towards the scene (e.g. Figure 1b) is not sufficient to learn an optimal policy. A first-person egocentric camera on the robot's end-effector (e.g. Figure 1a) has been shown to be necessary to learn an optimal policy in many tasks (Hsu et al., 2022). Approaches that leverage both first-person and third-person cameras simultaneously have been shown to improve the performance of RL algorithms (Hsu et al., 2022; Jangir et al., 2022; Barati & Chen, 2019). Whilst it is possible to create multiple camera views in simulation to improve training, access to multiple cameras in the real-world may be restricted due to hardware limitations. Therefore, it is desirable for an RL agent to be able to generalise from multiple cameras to a single camera for successful deployment. Even when multiple cameras are available in the real-world, cameras may fail or become faulty during deployment, so a robust RL agent should continue to perform optimally with only a subset of the cameras that were available during training. We propose to address these hardware limitations with an approach that leverages multiple cameras

(a) first-person (b) third-person

Figure 1: First-person and third-person camera views for MetaWorld Soccer task.

during training to learn a policy that successfully performs control tasks with multiple cameras as well as achieving robustness to a reduction in the available cameras.

We use multiple cameras to learn a disentangled representation that allows robustness to a reduction in cameras to generalise to any single camera from the training set. Our approach, called *Multi-View Disentanglement* (MVD), is a self-supervised auxiliary task to learn such a disentangled representation and can be used with existing RL algorithms. The learned representation is disentangled into a shared representation that is similar across all cameras, and a private representation that contains information only available to an individual camera. The shared representation gives the RL policy a consistent representation that can be relied upon regardless of the camera. The private representation allows information only available to a single camera to be used during training to improve policy learning as some camera views make it easier to discern important features than others.

We evaluate our approach on robotic control tasks using a Panda robot (Gallouédec et al., 2021; Hsu et al., 2022), and several MetaWorld tasks (Yu et al., 2020) using a Sawyer robot. We show experimentally that an RL agent often cannot learn an optimal policy when training on a third-person camera alone, and that an approach combining multiple cameras during training (Hsu et al., 2022) does not generalise to a single camera. Our results also show that our approach, MVD, is able to learn an optimal policy from multiple cameras and achieve zero-shot generalisation to successfully solve the task using any of the cameras individually in many tasks.

2 Related work

Robotic control with multiple cameras. Prior work uses multiple cameras to improve performance on robotic control tasks. Hsu et al. (2022) use both first-person and third-person cameras together with a variational information bottleneck to regularise the third-person representation, Jangir et al. (2022) use transformers with cross-view attention, Barati & Chen (2019) train multiple workers with different views and combine features for each worker weighted by Q-values, and Driess et al. (2022) use Neural Radiance Fields to learn a representation from multiple images to improve the performance of RL. These approaches combine camera representations to learn a policy that is dependent on all available cameras, and so cannot generalise if one of the training cameras is no longer available. Acar et al. (2023) trains a teacher RL policy on multiple cameras augmented with human demonstrations. They use imitation learning to learn a student single-camera policy to output an action similar to the teacher multi-camera policy. Shang & Ryoo (2021) also use imitation learning where demonstrations are a third-person view from a human or robot, which are used alongside first-person views of the egocentric robot to learn a disentangled representation for imitation learning. Our approach does not require training a separate teacher policy or collecting suitable demonstrations, and instead learns a disentangled representation online that allows generalisation to a single camera.

Multi-view representation learning. Multi-view representation learning approaches use multiple sources/views of a shared context. This can consist of multiple camera views of the same scene (as is the case in our work) as well as combining multi-modal inputs. Li et al. (2019) categorise approaches into representation alignment and representation fusion. Representation alignment includes minimising the distance between representations of different views (Feng et al., 2014; Li et al., 2003), maximising similarity between views (Bachman et al., 2019; Frome et al., 2013) and maximising correlation of variables across views (Andrew et al., 2013). Representation fusion combines the representations from different views into a single representation for downstream tasks (Geng et al., 2022; Xie et al., 2021; Karpathy & Fei-Fei, 2014). Multi-view disentanglement separates the learned representation into shared and private parts. The shared representation is aligned across views, while the private representation is view-specific. Several approaches have been proposed to achieve multi-view disentanglement in the supervised and unsupervised learning literature (Jain et al., 2023; Ke et al., 2023; Xu et al., 2021; Ke et al., 2021; Xin et al., 2021; Hosoya, 2019; Gonzalez-Garcia et al., 2018; Ye et al., 2016). Our work is aligned with the multi-view disentanglement literature, but we consider multi-view disentanglement for RL and use the temporal data available in RL.

Self-supervised auxiliary tasks in RL. Disentangled representations have been used to improve RL generalisation for a single camera (Dunion et al., 2023a;b; Higgins et al., 2017). Other approaches learn representations using mutual information (Lee et al., 2020; Garcin et al., 2024), regression targets (McInroe et al., 2024) and similarity constraints (Agarwal et al., 2021; Mazoure et al., 2020; van den Oord et al., 2018). CURL (Laskin et al., 2020) uses contrastive learning to maximise the similarity between representations of the same image with different augmentations to improve sample efficiency. However, Li et al. (2022) find that self-supervised learning frameworks with augmented images have limited impact on RL performance compared to image augmentation alone. While we also use contrastive learning with images, we consider camera views rather than augmentations.

3 Preliminaries

Reinforcement learning. We assume the agent is acting in a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(x_{t+1}|x_t, a_t)$ is the probability of next state $x_{t+1} \in \mathcal{S}$ given action $a_t \in \mathcal{A}$ is taken in state $x_t \in \mathcal{S}$ at time t, $R(x_t, a_t)$ is the reward function giving reward r_t after taking action a_t in state x_t, and $\gamma \in [0, 1)$ is the discount factor. The goal of an RL agent is to learn a policy π to maximise the discounted return, $\max_\pi \mathbb{E}_{P,\pi}[\sum_{t=0}^{\infty}[\gamma^t R(x_t, a_t)]]$. In this work, we focus on RL from image pixels, where the agent observation $\mathbf{o}_t \in \mathbb{R}^{c\times h\times w}$ at timestep t is an RGB image, a high-dimensional observation of the underlying state, where c is the channels, and h and w are the image height and width respectively. An observation can consist of multiple consecutive image frames where required for the task. The agent learns a lower-dimensional latent representation $\mathbf{z}_t$ and the policy π is now a function of the learned representation, $\pi(\mathbf{a}_t|\mathbf{z}_t)$. We consider N different camera views of the same scene; we use C to denote the set of cameras and $c_i \in C$ is a single camera. We use superscript to identify the observation $\mathbf{o}_t^{c_i}$ or representation $\mathbf{z}_t^{c_i}$ for camera $c_i \in C$.

Contrastive learning. We use contrastive learning to learn disentangled representations in a self-supervised way. Given a query $\mathbf{q}$ (sometimes also referred to as an anchor), contrastive learning aims to maximise the similarity between $\mathbf{q}$ and a positive key $\mathbf{k}^+$ while minimising the similarity with each negative key $\mathbf{k}^-$. Many approaches use the dot product $\mathbf{q}^T\mathbf{k}$ to measure the similarity between vectors $\mathbf{q}$ and $\mathbf{k}$ (Chen et al., 2020; He et al., 2020; Wu et al., 2018) or the bilinear product $\mathbf{q}^T W\mathbf{k}$ where W is a learnable weight matrix (Laskin et al., 2020; van den Oord et al., 2018).

We use the InfoNCE loss (van den Oord et al., 2018) with normalised dot product similarity measure, also known as cosine similarity, to measure the distance between vectors. This loss function was also used for the SimCLR (Chen et al., 2020) and MoCo (He et al., 2020) methods. Let $\text{sim}(\mathbf{q}, \mathbf{k})$ denote the cosine similarity between two vectors $\mathbf{q}$ and $\mathbf{k}$, given by:

$$\text{sim}(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T\mathbf{k}}{\|\mathbf{q}\|\,\|\mathbf{k}\|}. \tag{1}$$

Then the InfoNCE loss with query $\mathbf{q}$, positive key $\mathbf{k}^+$ and M negative keys $\{\mathbf{k}_i^-\}_{i=1}^M$ is given by:

$$\mathcal{L}^{\text{InfoNCE}}(\mathbf{q}, \mathbf{k}^+, \{\mathbf{k}_i^-\}_{i=1}^M) = \log\left[\frac{\exp\left(\text{sim}(\mathbf{q}^T, \mathbf{k}^+)/\tau\right)}{\exp\left(\text{sim}(\mathbf{q}^T, \mathbf{k}^+)/\tau\right) + \sum_{i=1}^M \exp\left(\text{sim}(\mathbf{q}^T, \mathbf{k}_i^-)/\tau\right)}\right] \tag{2}$$

where τ is a temperature parameter.

4 Multi-view disentanglement for RL

We propose *Multi-View Disentanglement* (MVD) to learn a disentangled representation from multiple cameras that is robust to a reduction in available cameras. Our approach learns separate shared $\mathbf{s}_t^{c_i}$ and private $\mathbf{p}_t^{c_i}$ representations for each camera $c_i \in C$, where C denotes the set of cameras and t is the timestep. The shared representation $\mathbf{s}_t^{c_i}$ is trained to be similar across all cameras $\forall c_i \in C$,

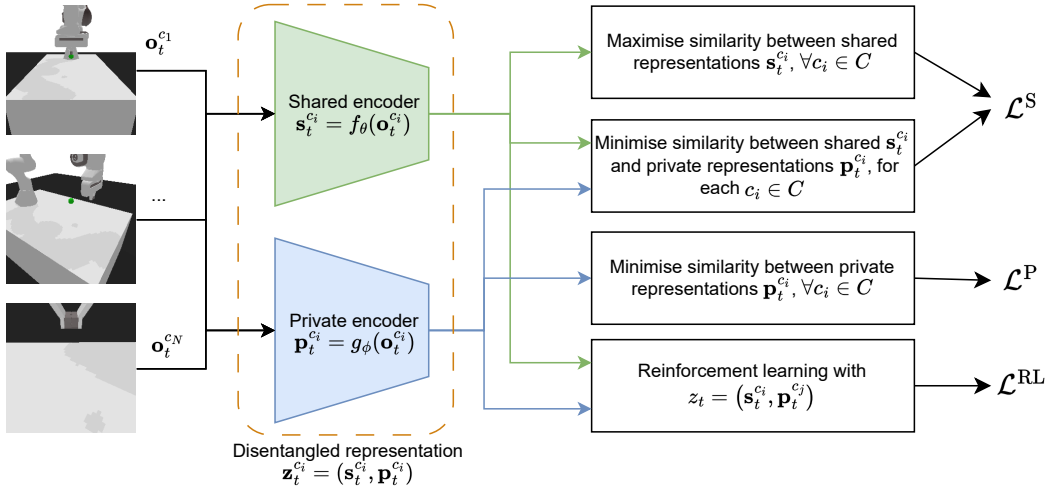


Figure 2: Multi-view Disentanglement (MVD) architecture. Each camera image is used to generate a shared and private representation. The shared auxiliary loss $\mathcal{L}^{\mathrm{S}}$ uses these representations to maximise similarity between shared representations and minimise similarity between shared and private representations. The private auxiliary loss $\mathcal{L}^{\mathrm{P}}$ minimises similarity between private representations.

while the private representation $\mathbf{p}_t^{c_i}$ encodes information available only to a specific camera $c_i \in C$. Both representations together are used to condition the RL policy $\pi(\mathbf{a}_t|\mathbf{z}_t)$ with $\mathbf{z}_t = (\mathbf{s}_t^{c_i}, \mathbf{p}_t^{c_j})$, where the shared part allows for camera generalisation and the private part allows the policy to leverage extra information available to a single camera to improve training. We define two contrastive learning objectives that together give an auxiliary task for multiple camera views that can be applied to existing RL algorithms. We first provide an overview of the approach in Section 4.1, then discuss the MVD auxiliary task in Section 4.2. Finally, we explain how the disentangled representation is used as input to the RL policy in Section 4.3.

4.1 Overview of approach

An outline of our approach is depicted in Figure 2. MVD consists of two separate encoders, which both learn a lower-dimensional latent representation of the same size for each camera $c_i \in C$. There is one encoder for the shared representation, $\mathbf{s}_t^{c_i} = f_\theta(\mathbf{o}_t^{c_i})$, where f_θ is an encoder parameterised by θ. There is a separate encoder for the private representation, $\mathbf{p}_t^{c_i} = g_\phi(\mathbf{o}_t^{c_i})$, where g_ϕ is an encoder parameterised by ϕ. Both encoders take the same observation image pixels of a single camera, $\mathbf{o}_t^{c_i}$, as input and encoder parameters are shared across all cameras. We use a contrastive learning approach to maximise similarity between the shared representations, $\mathbf{s}_t^{c_i}$, of all cameras $c_i \in C$ while minimising similarity between the shared $\mathbf{s}_t^{c_i}$ and private $\mathbf{p}_t^{c_i}$ representations for each camera $c_i \in C$ to achieve disentanglement. An additional contrastive loss is used to minimise similarity between private representations $\mathbf{p}_t^{c_i}$ for all $c_i \in C$ to ensure the policy cannot rely solely on the private representation. The contrastive learning details are provided in Section 4.2. Both the shared and private representations are used as input to the RL algorithm, which will be described in Section 4.3.

4.2 Multi-view disentanglement

Shared contrastive loss. The goal of the shared contrastive loss is to disentangle the shared $\mathbf{s}_t^{c_i}$ and private $\mathbf{p}_t^{c_i}$ representations for each camera $c_i \in C$ by minimising the similarity between these representations, while also maximising the similarity between the shared representation $\mathbf{s}_t^{c_i}$ for all cameras $\forall c_i \in C$ to achieve alignment. These two objectives are combined into one contrastive loss by defining suitable positive and negative keys. We use the InfoNCE loss in Equation 2 with the cosine similarity measure in Equation 1. For each calculation of the InfoNCE loss, one camera

$c_q \in C$ is selected as the query camera and another as the positive camera $c_+ \in C$ where $c_q \neq c_+$. The shared representation for the query camera c_q is the query for the contrastive loss, $\mathbf{q} = \mathbf{s}_t^{c_q}$. The shared representation of the positive camera c_+ is the positive key, $\mathbf{k}^+ = \mathbf{s}_t^{c_+}$. The negative keys consist of two sets: (1) the positive keys of all other queries in the batch $\{\mathbf{s}_{t'}^{c_+}\}_{t' \neq t}$, and (2) the private representations for all N cameras $\{\mathbf{p}_t^{c_n}\}_{n=1}^N$. The positive key encourages the shared representations to be similar across cameras, while the negative keys encourage (1) shared representations to be different for different timesteps to capture task-relevant information, and (2) shared and private representations to be dissimilar to disentangle them. The query, positive and negative keys are used as the input to the InfoNCE loss:

$$\mathcal{L}^{\mathrm{S}} = \sum_{c_q \neq c_+} \mathcal{L}^{\mathrm{InfoNCE}}(\mathbf{s}_t^{c_q}, \mathbf{s}_t^{c_+}, \{\mathbf{s}_{t'}^{c_+}\}_{t' \neq t} \cup \{\mathbf{p}_t^{c_n}\}_{n=1}^N) \tag{3}$$

where we sum the loss over each camera in a single update step.

Private contrastive loss. To prevent the shared representation from collapsing and the agent relying solely on the private representation, we minimise the similarity between the private representations. We use the InfoNCE loss with cosine similarity again to achieve this. The query is the private representation for a given camera at timestep t, $\mathbf{q} = \mathbf{p}_t^{c_q}$ for $c_q \in C$. The positive key is the private representation for the same camera at the next timestep $t+1$, such that $\mathbf{k}^+ = \mathbf{p}_{t+1}^{c_q}$. The negative keys are the private representations for all other cameras at timestep t, given by $\{\mathbf{p}_t^{c_n}\}_{\forall n \neq q}$. The positive keys are needed to keep the representation bounded and are chosen to encourage temporal consistency in the private representations at consecutive timesteps, while the negative keys encourage the private representation for each camera to be dissimilar. This prevents the agent from encoding information available to all cameras in the private representation. The query, positive and negative keys are used as the input to the InfoNCE loss:

$$\mathcal{L}^{\mathrm{P}} = \sum_{c_q} \mathcal{L}^{\mathrm{InfoNCE}}(\mathbf{p}_t^{c_q}, \mathbf{p}_{t+1}^{c_q}, \{\mathbf{p}_t^{c_n}\}_{\forall n \neq q}) \tag{4}$$

MVD loss. Since the similarity measure is normalised, we combine the shared and private contrastive losses by summing them, giving the MVD loss:

$$\mathcal{L}^{\mathrm{MVD}} = \mathcal{L}^{\mathrm{S}} + \mathcal{L}^{\mathrm{P}} \tag{5}$$

4.3 Reinforcement learning

We use both the shared and private representations to create the representation $\mathbf{z}_t$ as input to the RL algorithm (for both value and policy where applicable) because the shared representation alone does not benefit from the features that are easier to discern or only available to one camera during training. One shared representation $\mathbf{s}_t^{c_i}$ and private representation $\mathbf{p}_t^{c_j}$ are randomly sampled for each update step, such that $\mathbf{z}_t = \left(\mathbf{s}_t^{c_i}, \mathbf{p}_t^{c_j}\right)$ with $c_i, c_j \in C$. The shared and private representation do not have to come from the same camera due to the consistency across shared representations. We randomly sample the representations at each update step because combining all representations would result in a policy that can learn to rely on all cameras and would therefore fail to generalise if one camera is unavailable. Both the RL and MVD losses are used to update both shared and private encoders. Combining the RL loss with the MVD loss gives the loss used in training at every update step:

$$\mathcal{L} = \mathcal{L}^{\mathrm{RL}} + \mathcal{L}^{\mathrm{MVD}} \tag{6}$$

where $\mathcal{L}^{\mathrm{RL}}$ is the RL loss for the chosen RL algorithm.

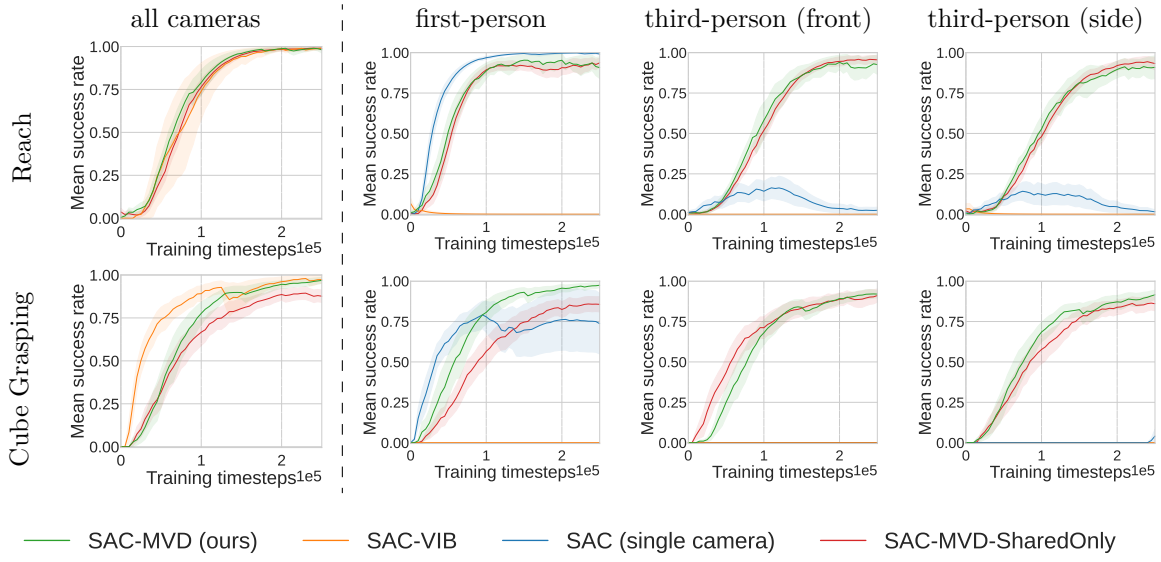


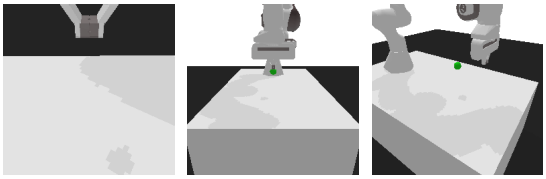
Figure 4: Results for Panda tasks showing success rate for evaluation on all cameras (left of dashed line) compared with success rate on each of the individual cameras (right). Success rate is averaged over 20 evaluation episodes for 5 seeds. The shaded region is standard deviation.

5 Experimental results

Our experiments evaluate performance on multiple cameras as well as zero-shot generalisation to any single camera from the training set. We evaluate our approach on robotic control tasks with a Panda robot using the Reach task from Panda Gym (Gallouédec et al., 2021) and a cube grasping task from Hsu et al. (2022). We also evaluate on four MetaWorld (Yu et al., 2020) tasks using a Sawyer robot. Our results show that MVD learns a policy that can solve the task with multiple cameras and generalise to a single camera. In many experiments, our approach is able to learn an optimal policy for a single camera where an agent trained directly on that camera alone is unable to learn at all.

5.1 Experimental setup

To demonstrate the broad applicability of our approach, we apply MVD to two RL algorithms and different numbers of cameras. For the Panda tasks we use SAC (Haarnoja et al., 2018) with a decoder to aid learning from images and three cameras (depicted in Figure 3). For the MetaWorld tasks we use DrQ (Yarats et al., 2021a), which uses image augmentations to improve sample efficiency on these more difficult tasks, and two cameras (depicted in Figure 1). In all tasks, one of the cameras is a first-person camera on the end-effector, while the remaining views are static third-person cameras. Implementation details and hyperparameters are provided in Appendix B.1. Our code is available at github.com/uoe-agents/MVD.



(a) first-person (b) third-person (front) (c) third-person (side)

Figure 3: Camera views used for Panda tasks.

In the MetaWorld tasks only, we also concatenate the representation with proprioceptive state information to improve training, similar to prior work using MetaWorld from images (Hsu et al., 2022). The proprioceptive state consists of the 3D end-effector position and 1D gripper width, and

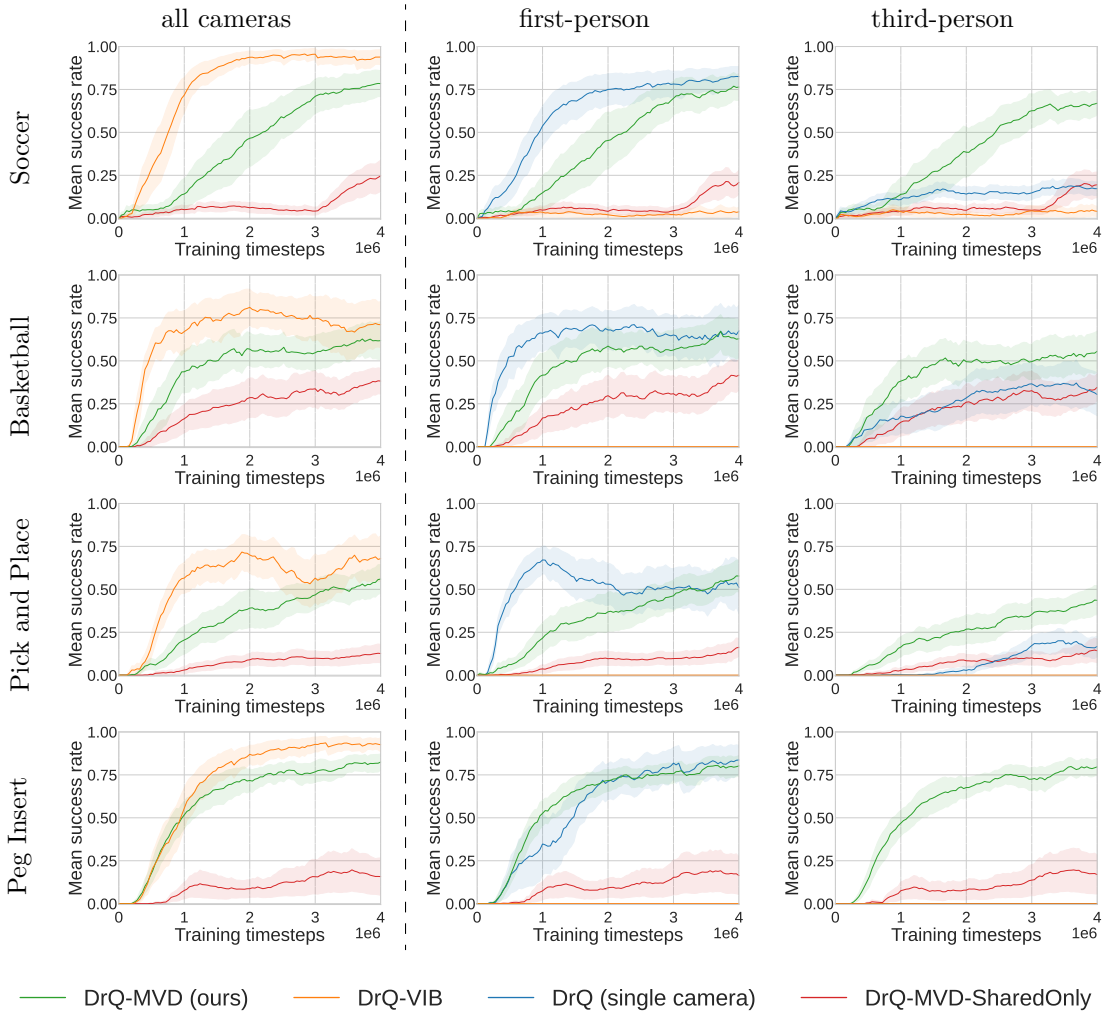


Figure 5: Results for MetaWorld tasks showing success rate for evaluation on all cameras (left of dashed line) compared with success rate on each of the individual cameras (right). Success rate is averaged over 20 evaluation episodes for 5 seeds. The shaded region is standard deviation.

is used for baselines as well as our method. However, we also demonstrate that our approach does not depend on proprioceptive state information by excluding it on the Panda tasks.

To demonstrate the importance of using multiple cameras in training, we compare to the same base RL algorithm trained only on a single camera for each available camera separately. We also demonstrate the importance of our disentanglement approach for generalisation over representation fusion approaches by comparing to VIB (Hsu et al., 2022), an approach that combines camera views with a variational information bottleneck on the third-person camera, for which we use the same base RL algorithm as MVD in each task. Finally, we include an ablation of our method that has only the shared representation with the corresponding loss to maximise similarity between the shared representation for all cameras ('MVD-SharedOnly'). This ablation is used to demonstrate the importance of the private representation during training.

5.2 Generalisation results

The results for Panda tasks in Figure 4 and MetaWorld in Figure 5, show the task success rate of evaluation episodes completed at intervals throughout training. For MVD and VIB, which both

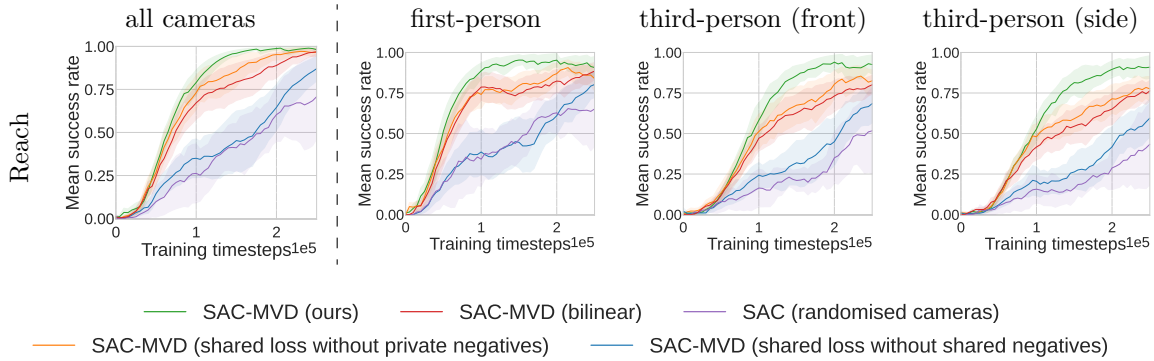


Figure 6: Results of ablation experiments for Panda Reach showing success rate for evaluation on all cameras (left of dashed line) compared with success rate on each of the individual cameras (right). Success rate is averaged over 20 evaluation episodes for 5 seeds. The shaded region is standard deviation. See text for details of each ablation.

train on all cameras, the success rate for evaluation episodes on all cameras is provided on the left of the vertical dashed line for each task. The success rate for each individual camera in the training set is provided on the right of the dashed line, representing the zero-shot generalisation performance of MVD and VIB while also showing the performance of the base RL algorithm that learns only with that individual camera.

As expected, the VIB baseline achieves optimal performance when evaluated on all three cameras but is unable to generalise to any individual camera. The base RL algorithm trained directly on each single camera is able to learn an optimal policy only for the first-person camera but fails to learn on the third-person cameras. In contrast, MVD achieves similar performance to VIB when evaluated on all cameras, albeit with lower sample efficiency in some tasks, but also achieves zero-shot generalisation to each camera, even when the base RL algorithm is unable to learn directly from that camera alone. MVD is the only method to attain consistent performance when evaluated on each camera individually.

The ablation of MVD with the shared representation only (MVD-SharedOnly) demonstrates the importance of the private representation in MVD. In all the tasks except Panda Reach, the shared representation alone achieves lower performance than MVD. This may be because the first-person camera is easier to learn from than the third-person cameras (as evidenced by the single camera baseline). Encouraging the shared representations to be similar for all cameras, without the flexibility of a separate private representation, may prevent important features from a single camera being used during training if the agent is not yet able to extract similar features from the other cameras.

5.3 Ablation study

In this section, we conduct more detailed analysis of MVD for the Panda Reach and MetaWorld Soccer tasks with ablation experiments to better understand the components of MVD. We also provide some analysis of the learned representations in Appendix C.

We consider the MVD shared loss (Equation 3), which uses two types of negative samples: shared representations from the same camera at different timesteps, and private representations for all cameras at the same timestep. We assess the impact of each type of negative sample by measuring performance when one type is removed, resulting in two MVD ablations: *MVD shared loss without shared negatives* and *MVD shared loss without private negatives*. The results in Figures 6 and 7 show that removing either of the negative samples reduces the performance of MVD, but the shared negatives have a much greater impact on performance than the private negatives. This may be due to the other constraints in the MVD shared and private losses indirectly encouraging disentanglement by encouraging shared representations to be similar and private representations to

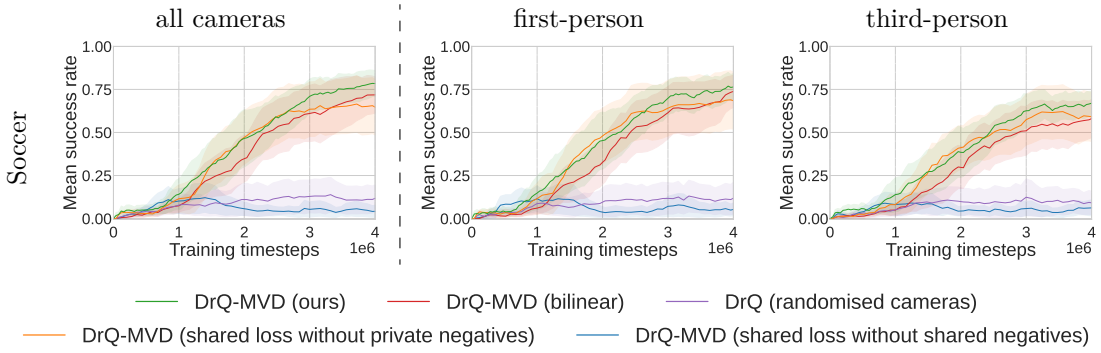


Figure 7: Results of ablation experiments for MetaWorld Soccer showing success rate for evaluation on all cameras (left of dashed line) compared with success rate on each of the individual cameras (right). Success rate is averaged over 20 evaluation episodes for 5 seeds. The shaded region is standard deviation. See text for details of each ablation.

be dissimilar, limiting the additional improvement gained by explicitly doing so through the use of private negatives in the shared loss.

To understand the impact of the similarity measure for InfoNCE, we compare the normalised dot product similarity measure used for MVD to an alternative *bilinear* similarity measure. The results, also in Figures 6 and 7, show that while the method is reasonably robust to the choice of similarity measure, the dot product outperforms the bilinear similarity measure, particularly in Panda Reach. This may be due to the learnable weight matrix in the bilinear similarity measure making it a more lenient constraint as the agent can learn to give less weight to some dimensions than others.

Finally, we consider the same training setup as MVD but without the MVD loss. We train the base algorithm with *randomised cameras* to replicate the MVD training setup by randomly sampling a camera representation as input to the RL loss at each timestep (as described in Section 4.3) but without the MVD loss to structure the learned representation. As in the previous section, the base algorithm is SAC for Panda Reach and DrQ for MetaWorld Soccer. The results in Figures 6 and 7 show that randomising the cameras alone is not enough to learn to solve the more difficult MetaWorld Soccer task. While camera randomisation does improve on the single camera baseline (see Figure 4) for the easier Panda Reach task, it achieves a much lower success rate than MVD.

6 Conclusion and future work

We demonstrated that camera perspective impacts the ability of an RL agent to learn an optimal policy. The impact can be mitigated by training with multiple cameras in simulation, but hardware constraints may prevent an RL agent from always relying on access to all of these cameras in the real-world. We propose Multi-View Disentanglement (MVD), an auxiliary task for RL algorithms to learn disentangled representations with a shared representation, which is aligned across all cameras, and a private representation, which is camera-specific. Our experiments showed that an RL agent trained only on a single third-person camera cannot learn an optimal policy in many control tasks, whereas MVD, benefiting from multiple cameras in training, achieves robustness to a reduction in cameras to solve the task with the same single third-person camera.

Future work could leverage progress on feature-level disentanglement, either within RL (Dunion et al., 2023a;b; Higgins et al., 2017) or in the multi-view disentanglement literature (Qiu et al., 2023; Hsieh et al., 2018), to further extend this work to disentangle individual features within the shared and private representations. Future work could also consider extending to a larger number of cameras to learn a representation that generalises out-of-distribution for sim2real transfer when real-world camera views do not perfectly match simulation.

References

Cihan Acar, Kuluhan Binici, Alp Tekirdag, and Yan Wu. Visual-policy learning through multi-camera view to single-camera view knowledge distillation for robot manipulation tasks. *IEEE Robotics and Automation Letters*, 9:691–698, 2023.

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1247–1255. PMLR, 2013.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

Elaheh Barati and Xuewen Chen. An actor-critic-attention mechanism for deep reinforcement learning in multi-view environments. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, 2019.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences*, 121(2), 2024.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Conditional mutual information for disentangled representations in reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023a.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Temporal disentanglement of representations for improved generalisation in reinforcement learning. In *International Conference on Learning Representations*, 2023b.

Fangxiang Feng, Xiaojie Wang, and Ruifan Li. Cross-modal retrieval with correspondence autoencoder. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MULTIMEDIA '14, pp. 7–16. Association for Computing Machinery, 2014.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc' Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NeurIPS 2013)*, 2013.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

Samuel Garcin, James Doran, Shangmin Guo, Christopher G. Lucas, and Stefano V. Albrecht. DRED: Zero-shot transfer in reinforcement learning via data-regularised environment design. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, 2024.

Xinyang Geng, Hao Liu, Lisa Lee, Dale Schuurams, Sergey Levine, and P. Abbeel. Multimodal masked autoencoders learn transferable representations. *ArXiv*, abs/2205.14204, 2022.

Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Offpolicy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735. IEEE Computer Society, 2020.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1480–1490. PMLR, 2017.

Sara Hooker, D. Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Haruo Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 2506–2513. International Joint Conferences on Artificial Intelligence Organization, 2019.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Kyle Hsu, Moo Jin Kim, Rafael Rafailov, Jiajun Wu, and Chelsea Finn. Vision-based manipulators need to also see from their hands. In *International Conference on Learning Representations (ICLR 2022)*, 2022.

Nihal Jain, Praneetha Vaddamanu, Paridhi Maheshwari, Vishwa Vinay, and Kuldeep Kulkarni. Self-supervised multi-view disentanglement for expansion of visual collections. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, pp. 841–849. Association for Computing Machinery, 2023.

Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, 7:3046–3053, 2022.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:664–676, 2014.

Guanzhou Ke, Zhiyong Hong, Zhiqiang Zeng, Zeyi Liu, Yangjie Sun, and Yannan Xie. Conan: Contrastive fusion networks for multi-view clustering. In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 653–660, 2021.

Guanzhou Ke, Yang Yu, Guoqing Chao, Xiaoli Wang, Chenyang Xu, and Shengfeng He. Disentangling multi-view representations be- yond inductive bias. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23)*, 2023.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch. *arXiv:2009.07896*, 2020.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 2020.

Kuang-Huei Lee, Ian Fischer, Anthony Z. Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in rl. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Dongge Li, Nevenka Dimitrova, Mingkun Li, and Ishwar K. Sethi. Multimedia content processing through cross-modal association. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, pp. 604–611. Association for Computing Machinery, 2003. ISBN 1581137222.

Xiang Li, Jinghuan Shang, Srijan Das, and Michael S Ryoo. Does self-supervised learning really improve reinforcement learning from pixels? In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

Yingming Li, Ming Yang, and Zhongfei Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge & Data Engineering*, 31:1863–1883, 2019.

Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Trevor McInroe, Lukas Schäfer, and Stefano V. Albrecht. Multi-horizon representations with hierarchical forward models for reinforcement learning. *Transactions on Machine Learning Research (TMLR)*, 2024.

Lin Qiu, Vernon M. Chinchilli, and Lin Lin. Interpretable representation learning from temporal multi-view data. In *Proceedings of The 14th Asian Conference on Machine Learning*, volume 189 of *Proceedings of Machine Learning Research*, pp. 864–879. PMLR, 2023.

Jinghuan Shang and Michael S. Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

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

Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Yuan Xie, Bingqian Lin, Yanyun Qu, Cuihua Li, Wensheng Zhang, Lizhuang Ma, Yonggang Wen, and Dacheng Tao. Joint deep multi-view learning for image clustering. *IEEE Transactions on Knowledge and Data Engineering*, 33(11):3594–3606, 2021.

Bowen Xin, Shan Zeng, and Xiuying Wang. Self-supervised deep correlational multi-view clustering. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

Jie Xu, Yazhou Ren, Huayi Tang, Xiaorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. Multi-vae: Learning disentangled view-common and view-peculiar visual representations for multi-view clustering. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9214–9223, 2021.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021b.

Tengqi Ye, Tianchun Wang, Kevin McGuinness, Yu Guo, and Cathal Gurrin. Learning multiple views with orthogonal denoising autoencoders. In *Conference on Multimedia Modeling*, 2016.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020.

A Extended background

We use SAC (Haarnoja et al., 2018) and DrQ (Yarats et al., 2021a) as the base RL algorithms for the MVD auxiliary task.

SAC is an actor-critic RL algorithm for continuous control. SAC maximises the expected return and entropy of the policy π. The critic Q minimises the loss:

$$L_Q = \mathbb{E}_{(\mathbf{o}_t,\mathbf{a}_t,\mathbf{o}_{t+1},r_t)\sim\mathcal{D}}\left[\left(Q(\mathbf{o}_t,\mathbf{a}_t) - r_t - \gamma\bar{V}(\mathbf{o}_{t+1})\right)\right)^2\right] \tag{7}$$

where $\mathbf{o}_t$ is the image observation and $\mathbf{a}_t$ is the action at time t. The actor π is trained by minimising the loss:

$$L_\pi = \mathbb{E}_{\mathbf{o}_t\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi}\left[\alpha_{\text{SAC}}\log(\pi(\mathbf{a}_t \mid \mathbf{o}_t)) - \min_{i=1,2}\bar{Q}_i(\mathbf{o}_t,\mathbf{a}_t)\right]\right] \tag{8}$$

where $\bar{Q}$ is exponential moving average of the Q network parameters. We augment SAC with a decoder, trained with an image reconstruction loss, to improve learning from images. The encoder and decoder details are provided in Appendix B.1.

DrQ is a data augmentation approach for robust learning from image pixels without the need for a decoder. DrQ adds padding and random crop augmentations to the image observations and averages over the target Q-value for each augmentation in the critic update as well as averaging over the augmentations for the Q function itself. The actor π uses unaugmented images and applies the SAC policy loss in Equation 8.

B Implementation details

B.1 MVD implementation

Our codebase is built on top of the public and open-source generalisation benchmark code provided by Hansen & Wang (2021), and uses the official DrQ implementation by Yarats et al. (2021a). A public and open-source implementation of MVD is available at github.com/uoe-agents/MVD.

Encoders. Both the shared encoder f_θ and private encoder g_ϕ consist of 4 convolutional layers, each with a 3×3 kernel size and 32 channels. The first layer has a stride of 2, all other layers have a stride of 1. There is a ReLU activation between each of the convolutional layers. The convolutional layers are followed by a linear layer, normalisation, then a tanh activation function. The output size (i.e. size of each representation) is 50. The encoder weights are shared between the actor π and critic Q.

Decoder. Where we use SAC as the base RL algorithm, we also include a decoder trained with an image reconstruction loss as this improves the sample efficiency of SAC with images (e.g. Yarats et al. (2021b)). The decoder is not used for DrQ. The first layer of the decoder is a fully-connected layer, which is followed by 4 deconvolutional layers, each with a 3×3 kernel size and 32 channels. Each deconvolutional layer has a stride of 1, except the last, which has a stride of 2. The reconstruction loss, which is used to update both the encoder and the decoder, is the mean squared error between the input image and the reconstructed image.

Actor and critic. Both the actor π and critic Q networks are multilayer perceptrons that each consist of two layers and have a hidden dimension of 1024. There is a ReLU activation after each layer except the last layer.

Hyperparameters. Table 1 shows the hyperparameters for all tasks for both MVD and baselines as they use the same base RL algorithm.

Hyperparameter	Value
Replay buffer capacity	100000
Initial steps before training begins	1000
Stacked frames	3 for MetaWorld, 1 for Panda
Action repeat	2 for MetaWorld, 1 for Panda
Batch size	128
Discount factor	0.99
Optimizer	Adam
Learning rate (actor, critic and encoder)	1e-3
α_{SAC} learning rate	1e-4
Q function soft-update rate	0.01
Actor update frequency	2
Actor log stddev bounds	$[-10, 2]$
Initial temperature	0.1
Image size	$3 \times 84 \times 84$
InfoNCE temperature	0.1

Table 1: Hyperparameter values.

C Representation analysis

We use saliency maps to show that our learned representations reflect shared and private features in the camera images. The attribution method we use is Integrated Gradients (Sundararajan et al., 2017) with SmoothGrad-Squared (Hooker et al., 2018) to reduce visual noise. However, while features attribution methods such as Integrated Gradients are commonly used in the literature, very recently Bilodeau et al. (2024) has shown that such feature attributions methods can fail to improve on random guessing for inferring model behaviour. We provide our representation analysis results for completeness.

C.1 Saliency map results

Using the learned encoders f_θ and g_ϕ, we calculate the attributions for each image pixel on each representation feature. We also calculate the attributions for each dimension in the representations on the output of the learned policy π. The pixel attributions are weighted by the corresponding policy attributions, and normalised for each camera to be in $[0, 1]$, to visualise the attention of the RL policy based on the features in the shared and private representations separately.

The resulting saliency maps in Figure 8 show that the shared and private representations focus on different features. The shared representation consistently focuses on features that are visible in all cameras, such as goal position for both Panda Reach and MetaWorld Soccer. The private representation focuses on camera-specific features that are not clearly visible in all cameras. In Panda Reach, the private representation for the first-person camera highlights the end-effector, which may be harder to extract from the third-person cameras; while the third-person front camera highlights the table edge, which is not visible in the first-person camera. In MetaWorld Soccer, the private representation for the first-person camera highlights the right edge of the goal post, which may help the agent guide the ball into the goal but is obscured by the robot in the third-person camera. The shared and private saliency maps are shown on different scales for readability. Comparing the maximum attribution on each of the scales shows that the attributions for the shared representations are higher than the corresponding private representations. This means that the policy focuses more on the shared representation by the end of training since the most important features for the task are visible in all cameras, such as the goal positions. However, the private representation is still required during training as evidenced by the ablation experiment in Section 5.2.

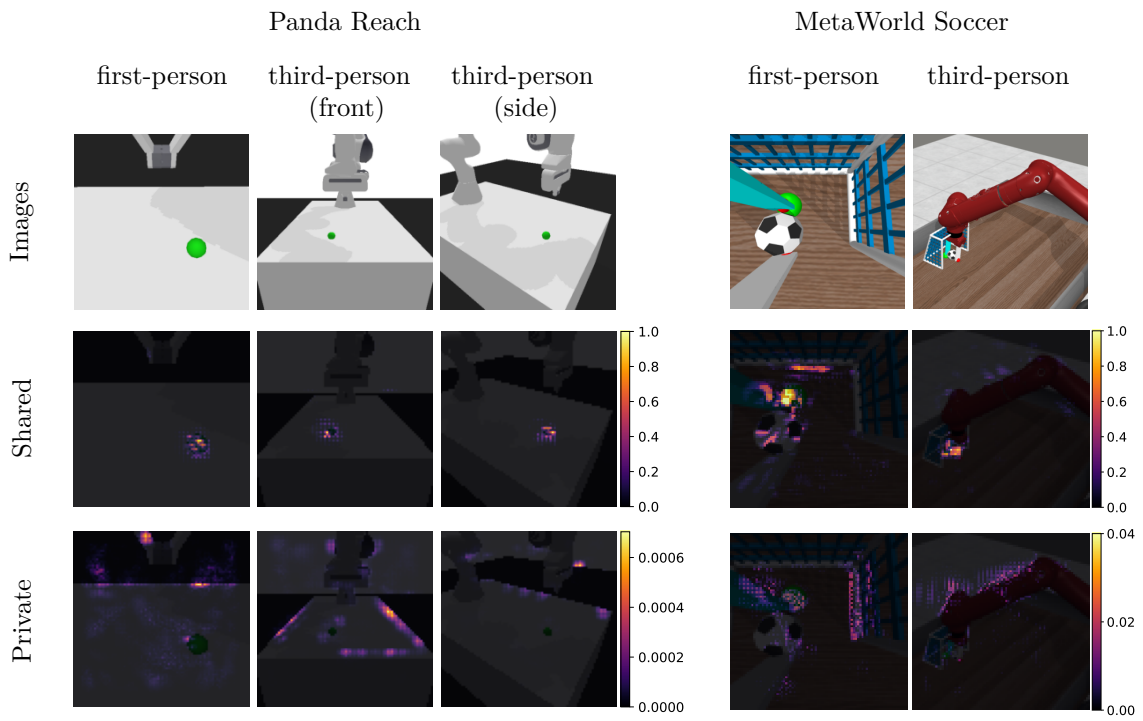


Figure 8: Policy saliency maps for MVD on the Panda Reach and MetaWorld Soccer tasks. The first row shows the original images used to calculate the attributions for each camera, the second and third rows show the saliency maps for the shared and private representations, respectively. Brighter pixels correspond to higher attributions.

C.2 Implementation details

We use the Integrated Gradients implementation from the open-source Captum library (Kokhlikyan et al., 2020) to calculate the attributions for the saliency maps in Figure 8. We use an input image for each camera (depicted in the first row of Figure 8) and an all black image as the baseline. We used SmoothGrad-Squared to reduce visual noise, which adds Gaussian noise to n copies of the input image, calculates the Integrated Gradient attributions for each of these noisy images, and returns the mean squared of the attributions across the noisy images. We use the implementation of SmoothGrad-Squared provided by Captum with the default $n = 5$.

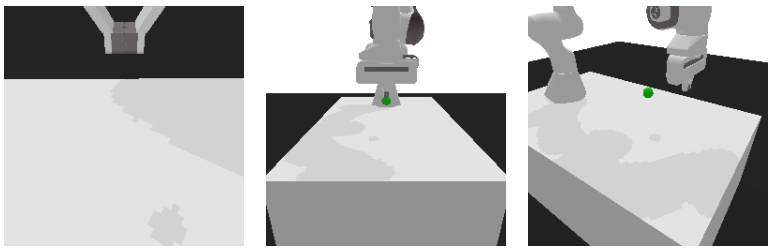
For the input image $\mathbf{o}_t^{c_i}$ for each camera $c_i \in C$, we calculated the image pixel attributions for each dimension of the output for both the shared encoder f_θ and the private encoder g_ϕ. For the corresponding representation $\mathbf{z}_t^{c_i} = (\mathbf{s}_t^{c_i}, \mathbf{p}_t^{c_i})$, we calculated the attribution of each dimension of both the shared and private representation on the output of the policy network $\pi(\mathbf{a}_t|\mathbf{z}_t)$. The pixel attributions for each dimension of the shared and private representation are weighted by the corresponding attributions from the policy network. The absolute value of the attributions are summed over all dimensions for each of the shared and private representations. The resulting pixel attributions allow us to visualise the attention of the policy network to the pixels based on whether those pixels were used by the shared or private representation. The attributions are re-scaled be in $[0, 1]$ for each camera for easier comparison across cameras and different representations. These normalised attributions are overlayed onto the input image to create the saliency maps. The shared and private saliency maps are shown on a different scale to improve readability of the private attributions since they are much smaller than the shared attributions.

D Environment details

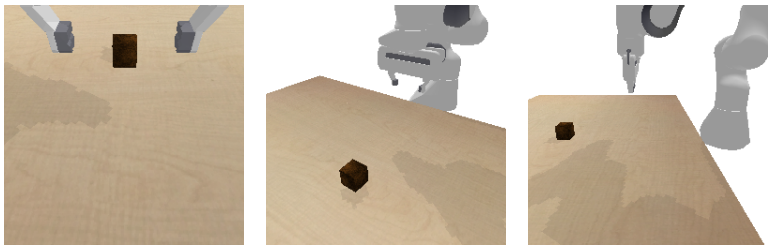
In this section, we provide a description and images of each task used in our experiments.

D.1 Panda tasks

We evaluate performance using tasks with the simulated Franka Emika Panda robotic arm instantiated in the PyBullet physics engine (Coumans & Bai, 2016–2019). For both tasks, we use three camera views as the observations generated with PyBullet, depicted in Figure 9.



(a) Panda Reach



(b) Panda Cube Grasping

Figure 9: Images showing each camera view used for the Panda tasks.

Panda Reach. We use the Reach task from Panda Gym (Gallouédec et al., 2021), where the goal is for the robot to place it's end-effector at a target position. We use the dense reward setting in which the agent receives a reward at each timestep based on the distance to the goal. The action space consists of the 3D end-effector position and 1D gripper control. The position of the end-effector and goal are randomly initialised at the start of each episode.

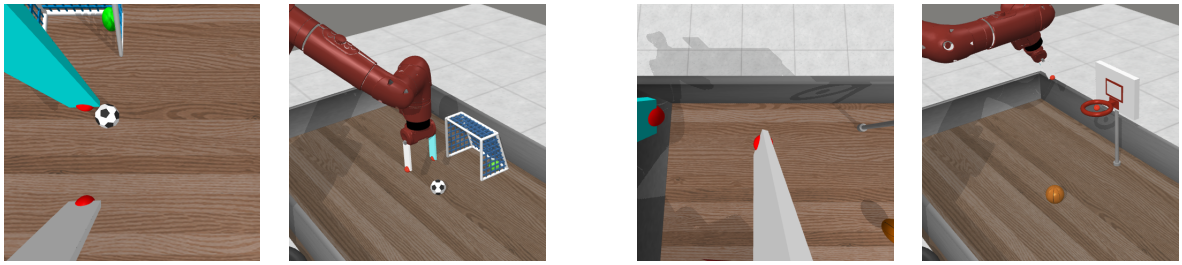
Panda Cube Grasping. The Cube Grasping task was proposed by Hsu et al. (2022). The goal is for the robot to grasp and pick up a cube. The reward at each timestep is based on the distance to the cube plus additional rewards if the agent is gripping or lifting the cube. The action space consists of the 3D end-effector position and 1D gripper control. The cube and end-effector position are randomly initialised at the start of each episode.

D.2 MetaWorld tasks

We evaluate performance on four tasks from the MetaWorld benchmark suite (Yu et al., 2020). All tasks use the simulated Sawyer robotic arm and are implemented in the MuJoCo physics engine (Todorov et al., 2012). We use the 'Multi-Task 1' setup which randomises the goal position for every episode within a single task. The action space consists of the 3D change in end-effector position and 1D gripper control. The goal of each task is described in Table 2. For all tasks, we render camera images from first-person and third-person perspectives for the observations, depicted in Figure 10.

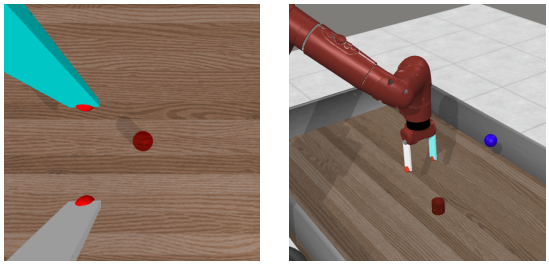
Task	Goal
Soccer	Guide the ball in to the goal
Basketball	Pick up the basketball and place into the hoop
Pick and Place	Move the object to the goal position
Peg Insert	Pick up and insert a peg sideways into the hole

Table 2: Description of MetaWorld tasks.

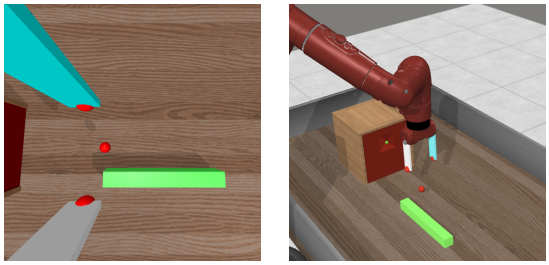


(a) Soccer

(b) Basketball



(c) Pick and Place



(d) Peg Insert

Figure 10: Images showing each camera view used for the MetaWorld tasks.