Self-Supervised Keypoint Learning

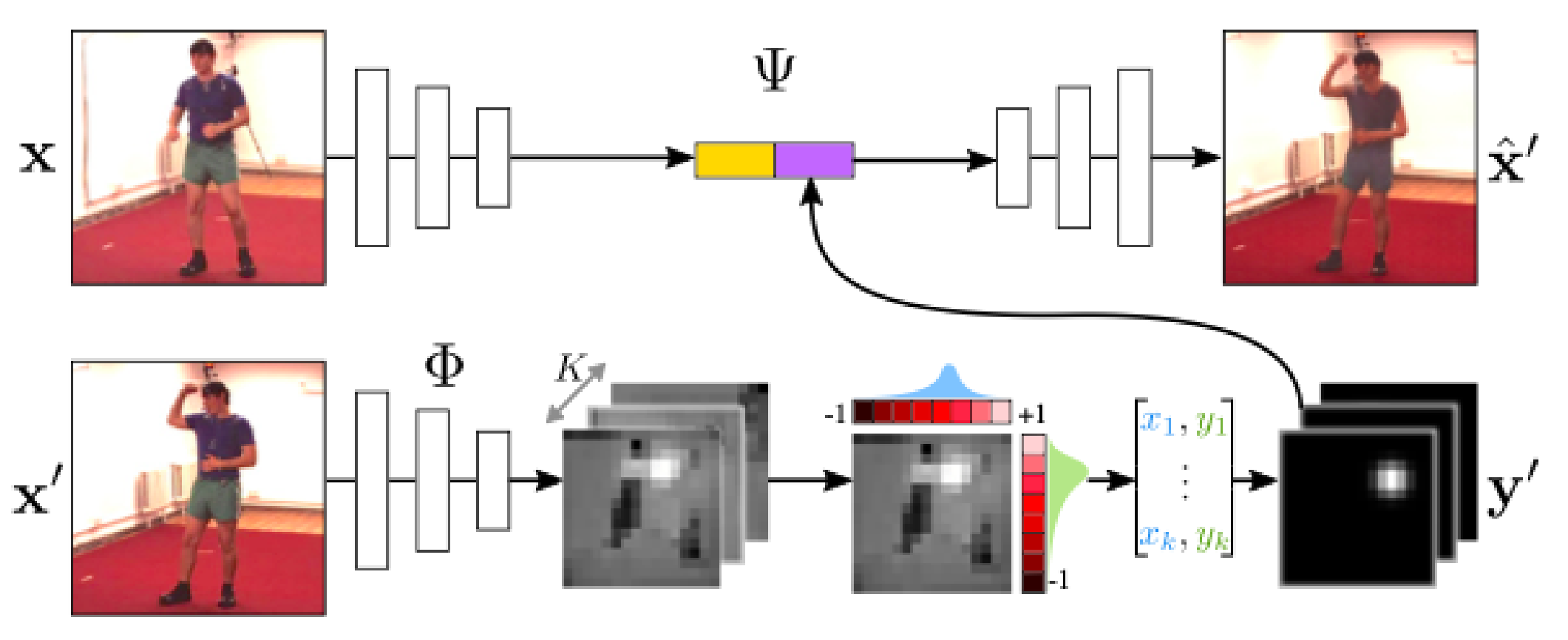
Andreas Lindhardt Plesner (s174505), Bjørn Jerram Dahl (s174480), Frederik Möbius Rygaard (s164222)

Abstract

This project investigates the applications of using self-supervised key point learning in Reinforcement Learning (RL) tasks. The implemented key point learning was based on [1] which utilizes Convolutional Neural Networks and auto encoding to convert images in an input batch to heat maps focusing on object landmarks in the images. The idea is to use key point learning in RL tasks to see if training and performance of such tasks can be improved significantly; here the cart-pole swing-up task from DeepMind's Control Suite is used [3]. The key point learning was implemented in SAC+AE [4]. Testing the method on the cart-pole data using only the original SAC+AE model yielded good results, but when adding key point learning, the results became significantly worse. In general, the addition of key points to SAC didn't improve the model but in fact made the model perform significantly worse; which may be caused by non optimal hyperparameters. Thus further analysis of the integration of key point learning to SAC should be made.

Key point learning and its Architecture

Given a source image $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{H \times W \times C}$, the objective is to learn a target image, $\mathbf{x}' \in \mathcal{X}$, where $\mathbf{x}$ and $\mathbf{x}'$ are extracted as frames e.g. in a video sequence. To do this, a function, $\Phi(\mathbf{x}) = \mathbf{y} \in \mathcal{Y} \subset \mathcal{X}$ is learned unsupervised such that $\mathbf{y}$ captures K object landmarks as described in [1]. To learn Φ unsupervised, a generator function, $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$, given by $\Psi(x, y) = x'$, is learned. Both Ψ and Φ are learned at the same time [1]. An overview of the architecture has been shown in **Architecture in key point learning**. For Φ at first an image encoder is applied to the source image. Based on this K score maps, $S_u(\mathbf{x}, k)$, are generated corresponding to the channels of a $\mathbb{R}^{H \times W \times K}$ tensor generated by the encoder. Hereafter, the discrete marginal probability distributions of each score map are determined using Softmax. Each set of probability distributions is then condensed to a point using the expected value – these points are the learned key points. Finally each map is replaced with a function which is proportional to the Gaussian distribution with hyper parameter, σ.Thus $\mathbf{y} = \Phi(\mathbf{x}) \in \mathbb{R}^{H \times W \times K}$ encodes Gaussian-like heat maps around the location of K maxima.



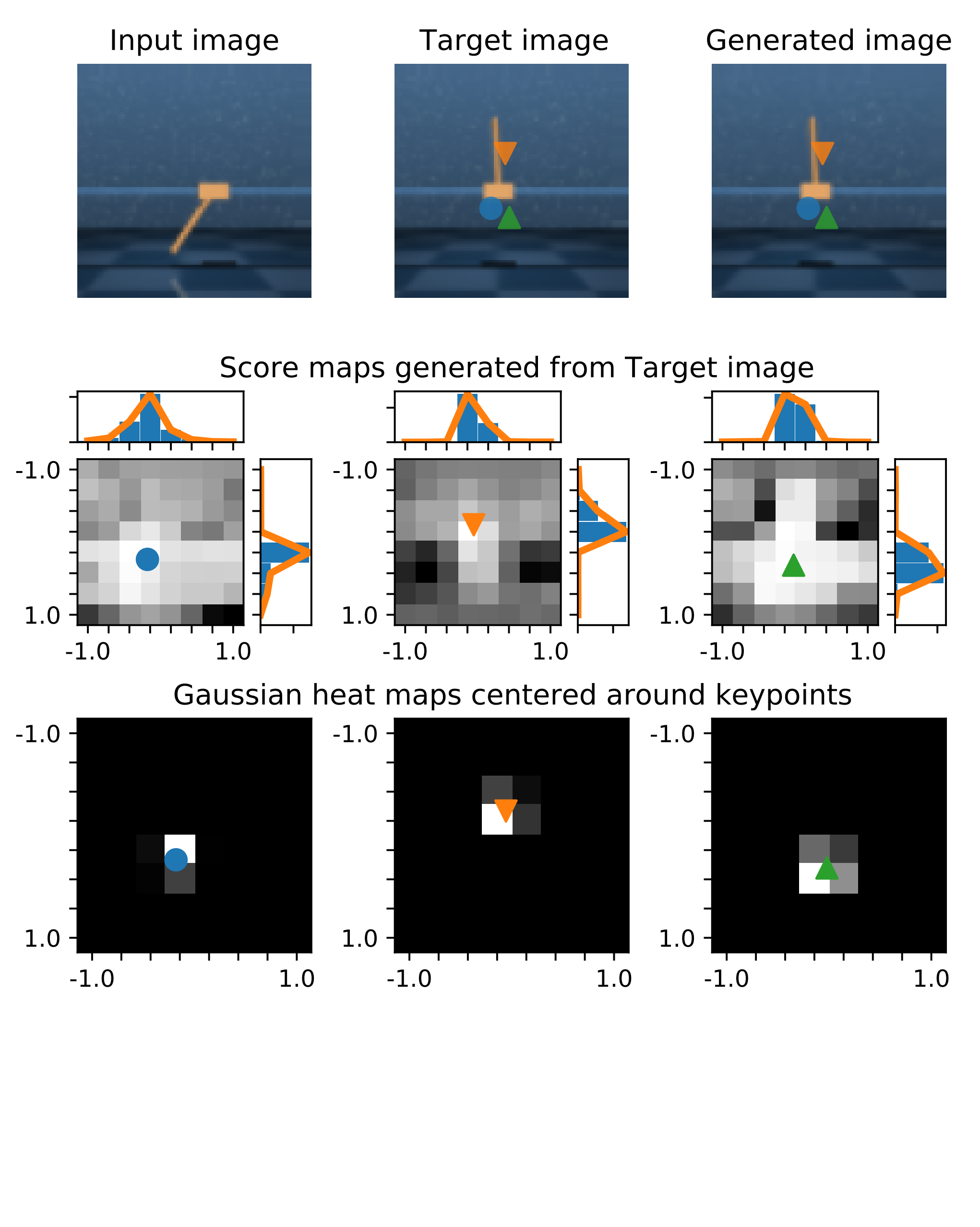
Architecture in key point learning. Visual representation of the IMM model; a detailed description can be found above. IMM refers to the entire model which gives a generated image based on an input and target image. KeyNet gives the score maps, key points, and heat maps. GenNet generates an image using an encoded input image and heat maps from the target image. Image is from [1].

The generator function $\Psi(\mathbf{x}, \mathbf{y}) = \hat{\mathbf{x}}'$ is learned such that the reconstruction error, $\mathcal{L}(\mathbf{x}', \hat{\mathbf{x}}') = \mathrm{MSE}(\mathbf{x}', \hat{\mathbf{x}}')$ is minimised. MSE is the Mean Squared Error loss function. $\mathcal{L}(\mathbf{x}', \hat{\mathbf{x}}')$ was minimised using the ADAM optimiser with a learning rate of 10^{-3}.

In order to increase the sample efficiency, data is augmented using random cropping. Furthermore, a replay buffer is used such that the policy also learns from past experiences; i.e. off-policy learning.

Results from key point learning

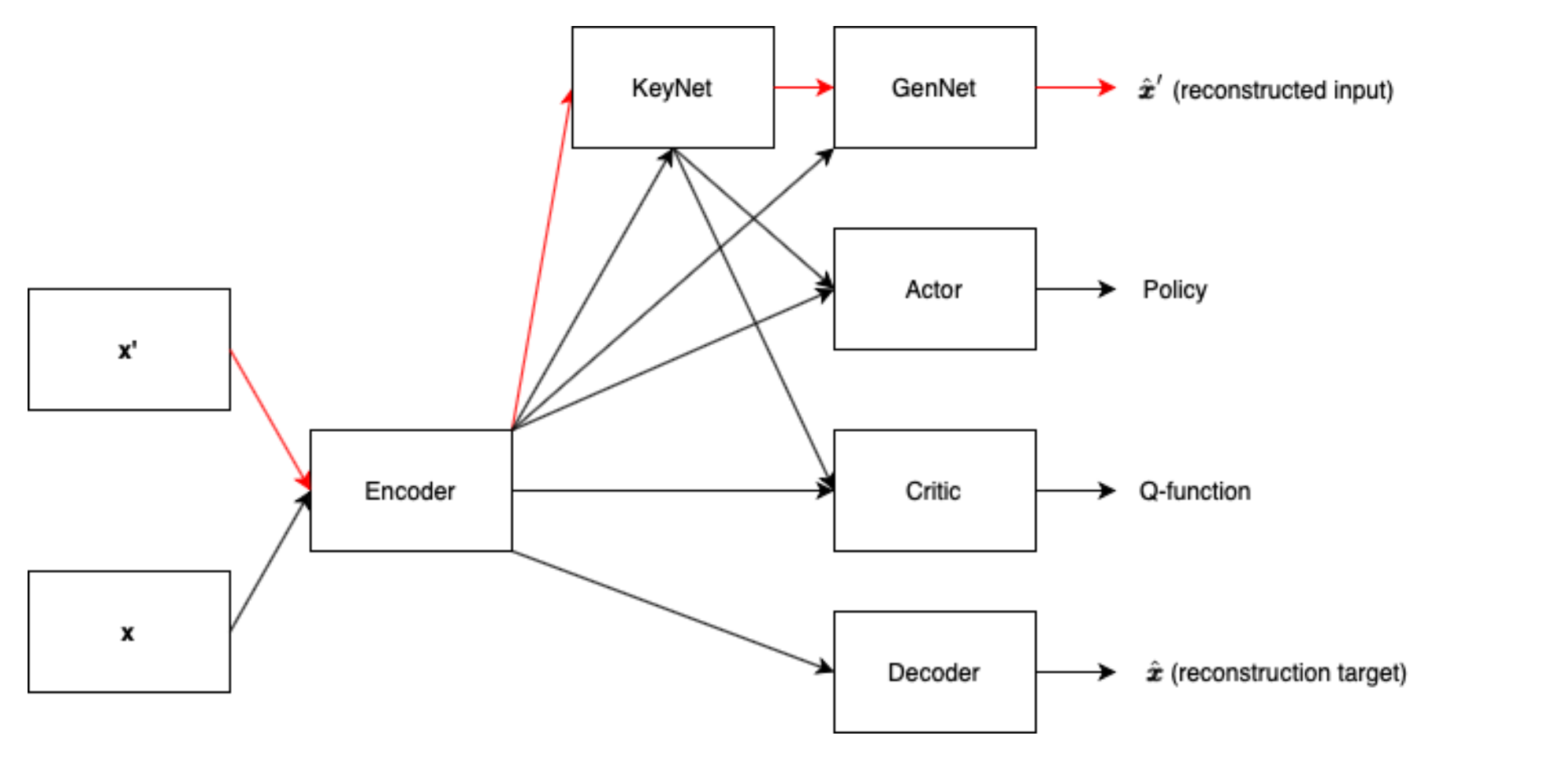
The following shows the key point learning results when training on a replay buffer containing RGB image frames from the cart-pole swing-up task. This data set is generated ahead of time and such these results purely showcase the IMM model's ability to learn relevant key points for cart-pole images.



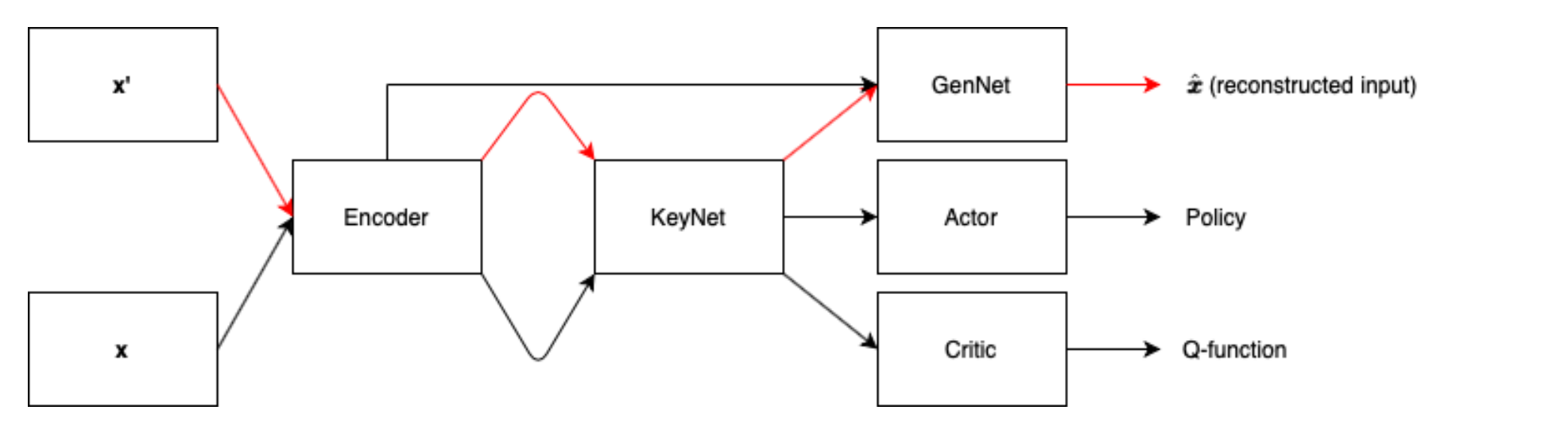
IMM example. Figure showing an example of an input, target, and generated image set along with corresponding examples of the score maps and heat maps for the three learned key points. First row from left to right: Input image, target image with key points from KeyNet, and generated image with target image's key points from KeyNet. Second row: Each image with histograms corresponds to one of the target image's key points; the marker shows which. The histograms show the marginal distributions of the corresponding score map from which the location of the key point is found using a weighted mean. Third row: gaussian heat maps generated by computing a gaussian distribution with center at the key point and standard deviation σ. σ is set to 0.1 as in [1] and the heat maps, therefore, only consist of few pixels with values noticeably above zero. The heat maps are, therefore, not included in later examples.

Soft-Actor-Critic and Autoencoders

In order to test the effect of using key point learning in Reinforcement Learning, the IMM model is integrated into the SAC+AE:pixel model [4]; pixel is omitted from here on. SAC+AE consists of a Soft Actor and Critic (SAC) method to learn the policy iteration, and an autoencoder (AE) to learn the latent states of the observations given to the SAC to find the optimal policy. SAC+AE code is based on the implemented Github version [4]. We modified the architecture slightly according to the structure described in [2]. We then added the IMM model to this version. Firstly, we try adding the IMM model to the SAC+AE model as shown in figure **SAC+AE with key points**; this model is denoted as SAC+AE+IMM. Secondly, we try completely replacing the AE with IMM in the SAC+AE model. This is shown in figure **SAC with key points**; this model is denoted as SAC+IMM.



SAC+AE with key points. A given observation, **x**, is first encoded and then fed to KeyNet. The encoded observation along with the result from KeyNet is then given to Actor and Critic. The AE is trained as in [4] and the IMM (KeyNet and GenNet) is trained similarly. The black arrows denote results originating from the input image and red arrows denote results originating from the target image.



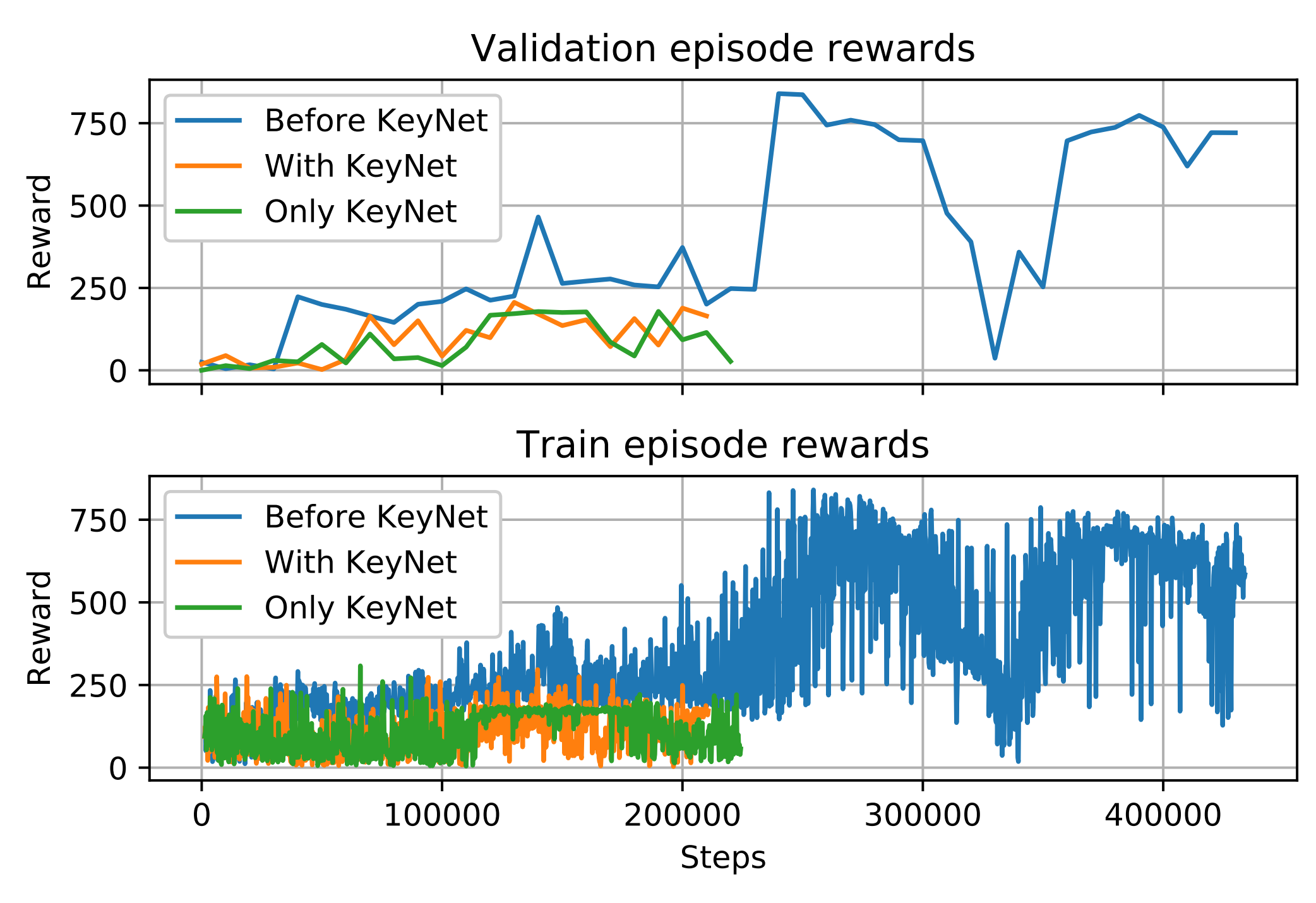
SAC with key points. A given observation, **x**, is first encoded and then fed to KeyNet. The IMM is trained as in SAC+AE+IMM. Black and red arrows are as in figure **SAC+AE with key points**.

SAC experiment results

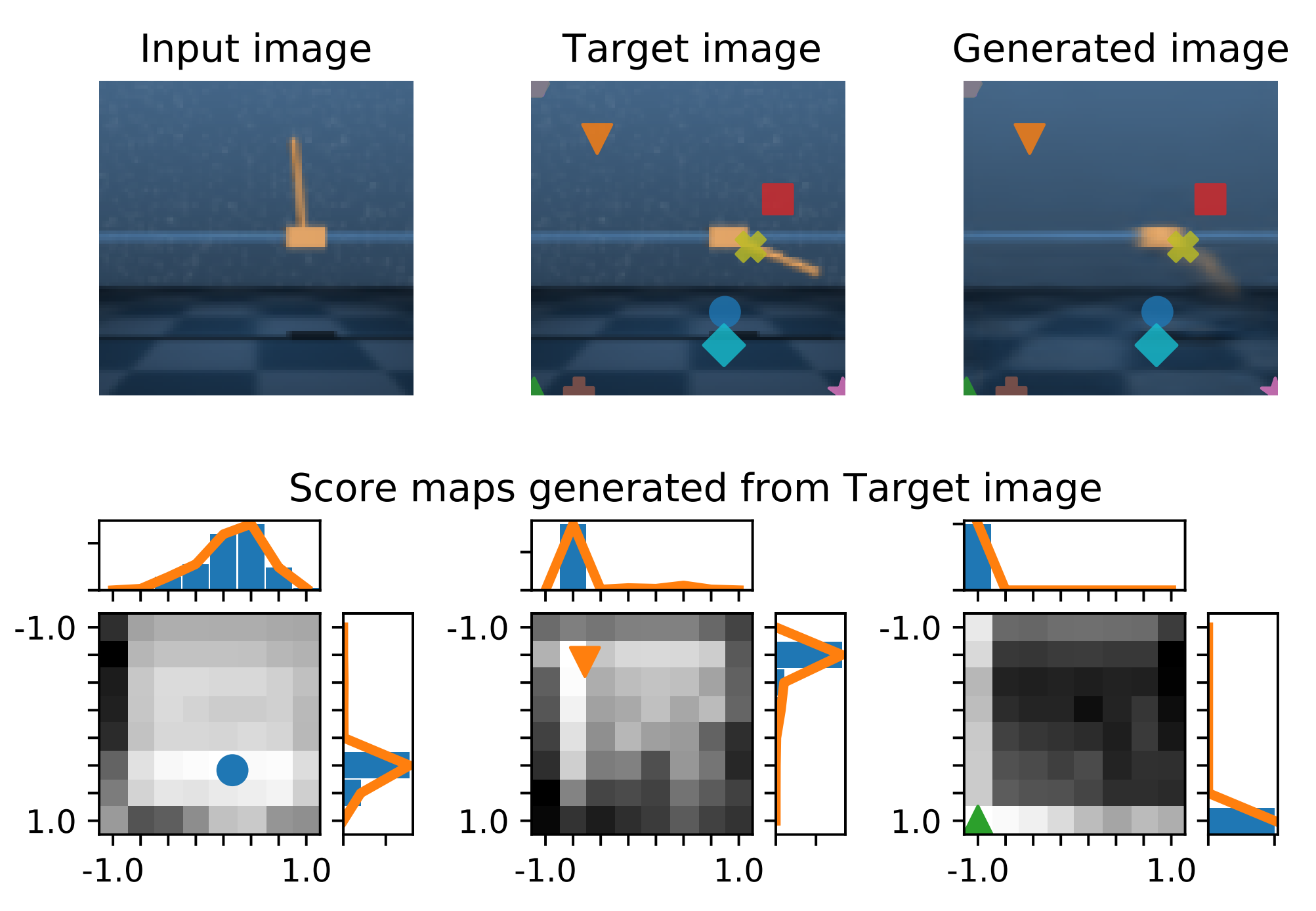
The following section presents results obtained during three experiments. Firstly, training the SAC+AE model as described in [4]. Secondly, adding IMM from [1] to the SAC+AE code; the architecture can be seen in figure **SAC+AE with key points**. Thirdly, training the SAC model using only IMM - hence removing the AE; the architecture can be seen in figure **SAC with key points**.

We also recreate figure **IMM example** using the resulting IMM from the second experiment. This can be seen in figure **IMM example from SAC+AE+IMM**. A similar figure for IMM from the third experiment is omitted since it is similar to the figure for the IMM from the second experiment.

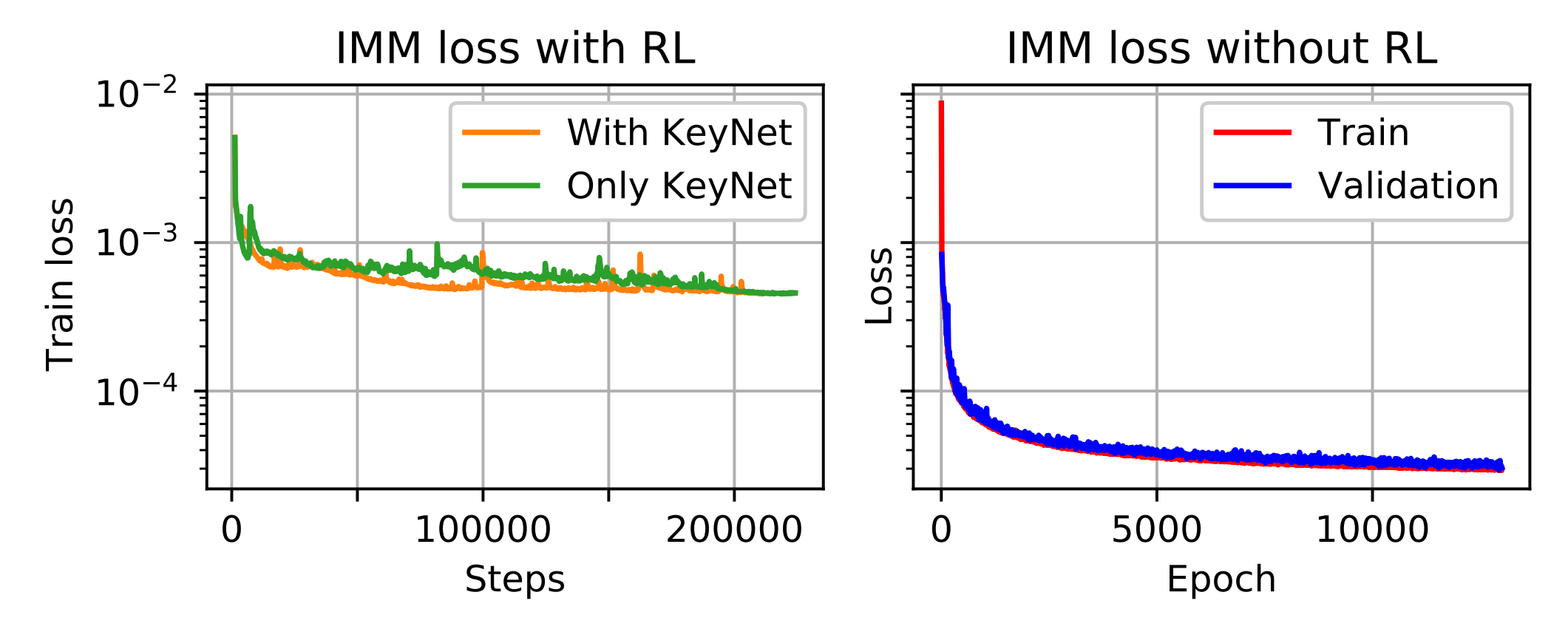
The IMM model is trained differently when trained along with SAC than when trained separately. When training the IMM on its own, a single input and a single target RGB image is fed into the model. When training the model during RL a stack of three input and a stack of three target RGB images are fed into the model. The key points are, therefore, not related to a single image, but, three similar images. A stack of three images are closely related in time.



Train and validation episode rewards during training of different variations of SAC using DeepMind Control Suite's CartPole environment [3] The blue *Before KeyNet* lines show the results of training the original SAC+AE, the orange *With KeyNet* lines show the results of training SAC+AE with the KeyNet implemented as in the **SAC+AE with KeyPoints** figure, and finally the green *Only KeyNet* lines show the results of training SAC with the KeyNet implemented, however, where the latent representation obtained from the image encoder is only provided to the KeyNet.



IMM example from SAC+AE+IMM. Figure showing an example of the first image in an input, target, and generated image set along with corresponding examples of the score maps for three of the key points. Heat maps have not been shown. Detailed description can be found in figure **IMM example**



Comparison of IMM loss when trained alone or during RL. Left) IMM loss obtained by training SAC+AE+IMM (the orange *With KeyNet* line) and SAC+IMM (the green *Only KeyNet* line) using DeepMind Control Suite's CartPole environment [3]. Here, observations correspond to stacks of 3 concurrent RGB image frames, wherefore, the learned keypoints relate to three similar images. Right) Train and validation loss obtained by training IMM on its own on RGB image frames from a CartPole replay buffer (i.e. no reinforcement learning is involved during training of IMM).

Conclusion and Future Work

In this poster the application of key point learning to RL tasks has been investigated on DeepMind Control Suite's cart-pole swing-up task. The implementation of key point learning and its direct application to RGB image frames from the cart-pole environment showed promising results, where the generated image using IMM was relatively similar to the target image. However, when integrating IMM with SAC the results were less promising. In fact the results were significantly worse compared to the original implementation of SAC+AE. Given the promising results of the reconstruction using key points, this indicates that integration of IMM in SAC should be investigated further in order to improve the results.

For future work, the hyperparameters of SAC+AE+IMM and SAC+IMM should be investigated further. We have already made a few tests where the learning rate was decreased by a factor of five (from 10^{-3} to $2 \cdot 10^{-4}$). These tests yielded episode rewards which were more stable and on average higher than the results presented in this poster – indicating that tuning of hyperparameters is likely to improve performance.

Note also that as of now SAC+AE is trained on twice as many steps compared to the the other two models, and therefore increasing the number of episodes might also yield better results. An alternative way is to change the implementation of IMM in SAC such that IMM is trained in parallel to SAC or pretrained on a pregenerated replay buffer.

References

[1] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. *CoRR*, abs/1806.07823, 2018.

[2] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. 2020.

[3] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. 2018.

[4] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *CoRR*, abs/1910.01741, 2019.