Zero-shot stitching in Reinforcement Learning using Relative Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper we investigate the use of a recent method called "relative representations" to enable zero-shot model stitching in visual RL between encoders and policies trained on the CarRacing environment, which does not require additional training. Our experiments show that the relative representation framework can be effectively applied to the RL realm to obtain compositionality and therefore zero-shot stitching across agents with multiple variation factors: i) random seed for the training; ii) environment style (background color); iii) training algorithm used (PPO and DDQN).

1 Introduction

Model stitching is a technique that has been explored in standard deep learning but, to the best of our knowledge, it is not applied to reinforcement learning (RL). It is standard practice in the Reinforcement Learning (RL) field to repeatedly train agents from scratch in an end-to-end fashion: both the feature extractor module and the policy. Ideally, in scenarios where multiple tasks are performed within the same environment, it would be advantageous to freeze a policy trained on a certain task and swap its encoder with another could allow the policy to perform on unseen tasks or environment variations, such as driving a car on different weather conditions, all without retraining. Moved by this idea, we decided to exploit relative representations [Moschella et al., 2023], a method for zero-shot communication between latent spaces. We demonstrate that it is possible to perform zero-shot stitching between encoders and policies by using relative representations. This enables the reuse of neural components and could be an initial step towards model stitching across different tasks. We conduct our preliminary experiments using a modified version of the CarRacing environment (Figure 1. This modified environment has a discrete action space and allows for background color changes. First, we train our end-to-end agents, which are comprised of a standard CNN-based feature extractor and a policy. Empirical results then demonstrate that by utilizing relative representations, a policy that was trained jointly with an encoder in an environment with a specific background color, could seamlessly be combined with an encoder that was trained on a different background color in a zero-shot manner, with little to no performance loss.

2 Preliminaries

2.1 Reinforcement Learning from images

We formulate image-based control as a Markov Decision Process (MDP) [Bellman, 1957]. As opposed to the common practice [Mnih et al., 2013], we do not approximate the current state of the system by stacking three consecutive prior observations, but adopt a simplified approach and use a single frame. With this in mind, such MDP can be described as a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$, where $\mathcal{X}$

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

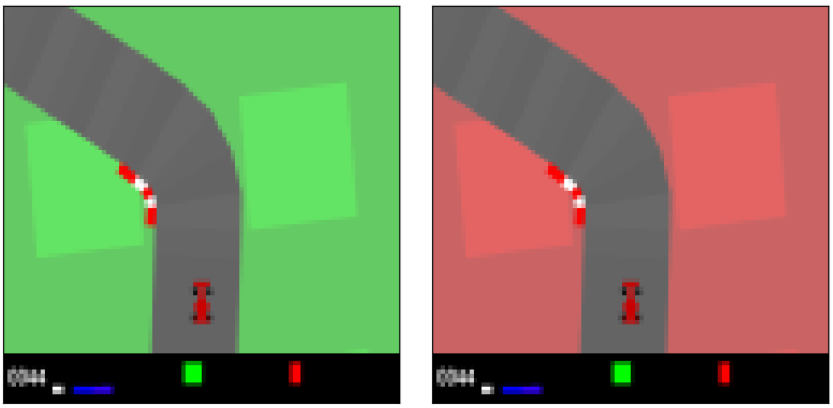

Figure 1: The modified version of the Car Racing environment, where we can change the color of the background.

is the state space (a three-stack of image observations), $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$ is the transition function that defines a probability distribution over the next state given the current state and action, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to [-100, 1]$ is the reward function, $\gamma \in [0, 1)$ is a discount factor, and $d_0 \in \Delta(\mathcal{X})$ is the distribution of the initial state x_0. The goal is to find a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ that maximizes the expected discounted sum of rewards:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] , \tag{1}$$

where $x_0 \sim d_0$, and for all t, we have $a_t \sim \pi(\cdot|x_t)$, $x_{t+1} \sim \mathcal{P}(\cdot|x_t, a_t)$, and $r_t = \mathcal{R}(x_t, a_t)$.

2.2 Model stitching

Model stitching refers to the process of combining different neural networks to create a novel model. It involves integrating parts from multiple networks or ensuring compatibility between latent spaces for tasks such as verifying similarity. Several studies have explored this concept and introduced techniques to facilitate model stitching. One approach is the utilization of trainable stitching layers, which have been previously introduced in literature [Lenc and Vedaldi, 2015, Bansal et al., 2021, Csiszarik et al., 2021]. These stitching layers enable combining specific network components or facilitate comparisons between latent spaces, effectively integrating different network parts. Alternatively, certain works have proposed methods that generate directly compatible and reusable network components without relying on explicit stitching layers [Gygli et al., 2021, Yaman et al., 2022]. These approaches aim to produce network components that can be readily combined without the need for additional stitching layers. An important concept in this field is that of relative representations [Moschella et al., 2023, Norelli et al., 2022]. Relative representations enable zero-shot stitching between distinct neural networks trained on semantically similar datasets. This approach assumes the use of decoders trained on relative representations, which facilitates the effective reuse of models. Leveraging relative representations makes it possible to stitch together different networks trained on semantically similar data, even in scenarios where the networks were not originally designed to be combined.

2.3 Relative Representations

Relative representation, as proposed by [Moschella et al., 2023], offers a simplified framework for facilitating communication among diverse latent spaces in neural models. This approach introduces an alternative perspective by shifting from an absolute coordinate system to a relative one, with respect to a predefined set of anchor samples. It involves representing samples in the latent space of neural networks by the distances each point z in the space $Z \subset \mathbb{R}^d$ has with the anchor set $A_Z \subset Z$. When cosine similarity is chosen as distance function, it can be formalized as:

$$Z_{rel} = Z_{abs} \cdot A_Z^T \tag{2}$$

Here, $Z_{rel} \in \mathbb{R}^{n \times k}$, $A_Z \in \mathbb{R}^{k \times d}$, and $Z_{abs} \in \mathbb{R}^{n \times d}$, where $n = |Z|$ and $k = |A_Z|$. To ensure consistency, both the samples in Z and A_Z are rescaled to unit norm, i.e., $z = \frac{z}{|z|}$ for all $z \in Z$ and $a = \frac{a}{|a|}$ for all $a \in A_Z$.

By employing relative representation in a latent space, the inherent structure of the space is emphasized, enabling compatibility between seemingly dissimilar spaces due to extrinsic factors. In this work, we apply relative representation to remove the following extrinsic factors: i) the background color of the observations; ii) the training algorithm for the policy; the random seed used for training.

3 Method

For our methods, we frame agents in Visual RL as composed of an encoder and a policy module. We suggest employing relative representations to enable the reuse of encoders and policies trained on different variations of the environment. Reuse is performed via zero-shot model stitching. To achieve this, encoders and policies must be trained using relative representations.

Therefore, given two agents, denoted as A_i and A_j, we define the encoders for agents A_i and A_j by $E^{(i)}(x)$, and $E^{(j)}(x)$, respectively. An encoder takes an observation x as input and produces a representation $z^{(i)}$:

$$z^{(i)} = E^{(i)}(x)\,. \tag{3}$$

Similarly, the policies can be represented by $\pi^{(i)}(z)$ and $\pi^{(j)}(z)$. A policy takes the representation $z^{(i)}$ produced by the encoder and generates an action $a^{(i)}$:

$$a^{(i)} \sim \pi^{(i)}(z^{(i)})\,. \tag{4}$$

To stitch the agents together, we can use the representation $z^{(i)}$ from agent A_i as input to the policy of agent A_j, resulting in a stitched agent:

$$\mathrm{A}_s = \pi^{(j)}(z^{(i)})\,. \tag{5}$$

Throughout this paper, we will use encoders and policies trained using PPO, unless otherwise specified.

4 Experiments

In this section we compare the stitching performance between agents trained using absolute and relative representations.
Then, we perform an analysis of the latent spaces, comparing the absolute encoders to the relative ones, showing that relative encoders produce the same features regardless of the background color of the input observation.

4.1 Policy Stitching Evaluation

After training different agents on different background colors in the modified version of the Car Racing environment, we freeze the encoders and policies of each agent and evaluate their performance on different tracks, stitching for every combination of the background colors. Namely, the colors are: *green*, *red*, *blue*, *yellow*. Tests are performed for every color and over 5 track seeds, corresponding to different tracks. Each cell shows the average return and the standard deviation obtained along the 5 tracks. To be more precise, the score recorded is the maximum return obtained during the run, meaning that if the agent starts taking penalties because it goes offroad, we do not lower the score. Table 1 compares the scores obtained when evaluating agents with relative and absolute stitching. Although suffering from higher variance, the scores obtained using relative representations remain consistent among every encoder-policy stitching test. With absolute representation stitching, instead, policies are unable to interpret the representations coming from different encoders, therefore being unable to drive.

Please note that for this test we trained every agents using the same seed, meaning that models' weights had the same initialization. As we will see in the next section, this is very helpful for agents using absolute representations, because they are able to learn representations starting from the same network, hence converging to something that preserve some similarity between different encoders.

		Policy							
		`green`		`red`		`blue`		`yellow`	
		Rel	*Abs*	*Rel*	*Abs*	*Rel*	*Abs*	*Rel*	*Abs*
Encoder	`green`	745 ± 235	846 ± 126	774 ± 287	35 ± 32	705 ± 292	9 ± 1	820 ± 146	708 ± 255
	`red`	762 ± 292	159 ± 41	839 ± 120	802 ± 230	739 ± 190	8 ± 2	805 ± 198	710 ± 111
	`blue`	821 ± 127	8 ± 1	835 ± 174	14 ± 8	827 ± 180	890 ± 48	741 ± 214	7 ± 1
	`yellow`	740 ± 262	525 ± 175	787 ± 285	377 ± 85	780 ± 271	9 ± 2	848 ± 133	808 ± 236

Table 1: Episode maximum return comparing stitching using absolute and relative representations. Encoder (rows) and policy (columns) colors represent the track background on which that module was trained on.

4.2 Latent space analysis

Latent space analysis shows the features produced by the absolute and relative encoders. We collected environment observations for two different background colors by using an independent model and letting it complete an episode. This way, the observations are independent from the models we are comparing. Specifically, when using the independent model we save observations $x_i^{(1)}$ and $x_j^{(2)}$, for $i, j = 0...T$ coming from the same environment track but with different background colors. We perform the same interactions, the car will be in the same positions, the only thing that changes is the color of the background, hence when $i == j$ we would have the same image but with different colors. Each figure shows the pairwise similarity between the representations of two encoders $E^{(1)}$ and $E^{(2)}$, comparing $E^{(1)}(x_i^{(1)})$ and $E^{(2)}(x_j^{(2)})$. A strong value along the diagonal is a clear indicator that the representations produced by the two encoders are similar. Figure 3 compares absolute (3.a) and relative (3.b)representations for encoders trained on green and red backgrounds using PPO and same seed. Figure **??** performs the same comparison but with model trained using different seeds, hence having different starting weights. It is easy to notice that the absolute representations are even less similar in this setting, while the relative ones maintain their similarity. If we were to rerun the tests in Table 1 using a models trained on a different seed for each color, we would probably see even worse performance when using absolute representations.

Finally, Figure 4 shows comparison on green (4.a) and red (4.b) backgrounds, between relative encoders trained using two different RL algorithms DDQN and PPO. Surprisingly, the representations produced by the two encoders are similar, indicating that encoders tend to learn the same features regardless of the underlying learning algorithm.

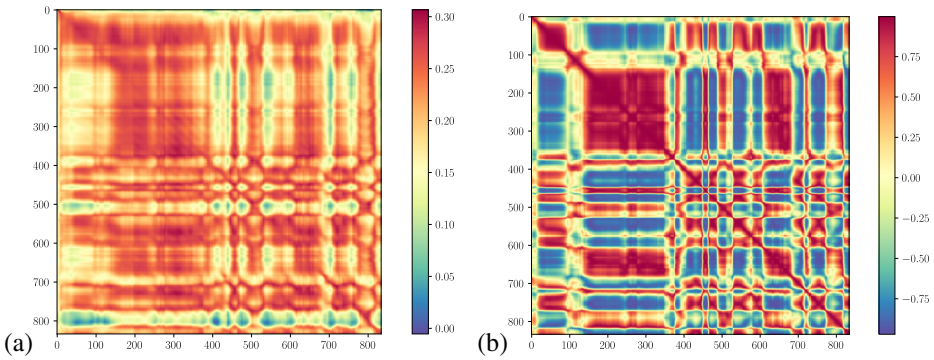

Figure 2: Similarity matrices comparing the similarity between the green background representations and the red background representations. (**a**) shows the absolute latent spaces, while (**b**) the relative latent spaces. The same seed is used for these two plots.

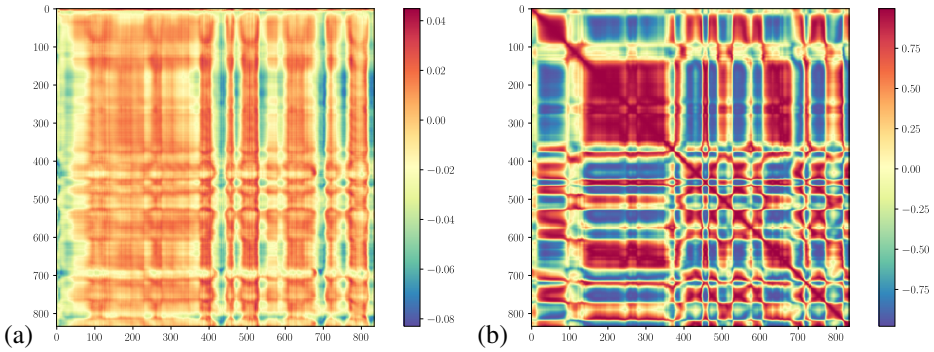

Figure 3: Similarity matrices comparing the similarity between the green background representations and the red background representations. (**a**) shows the absolute latent spaces, while (**b**) the relative latent spaces. Each matrix shows comparison between encoders trained on different seeds.

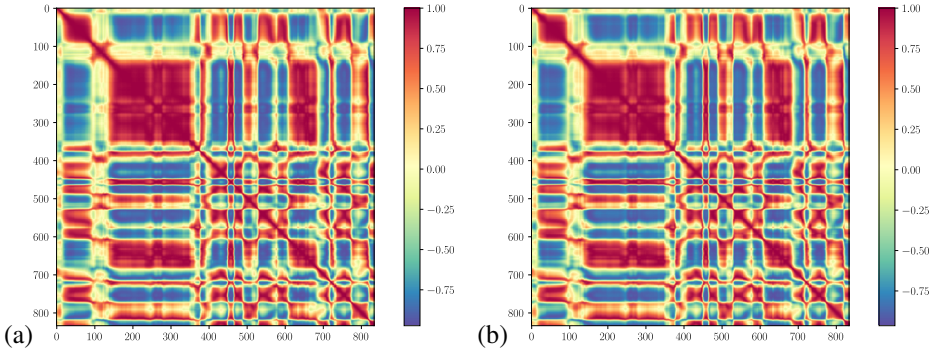

Figure 4: **a**) shows the similarity between PPO and DDQN representations on the green background, while (**b**)shows the similarity between PPO and DDQN representations on the red background.

5 Conclusion

The use of relative representation in Reinforcement Learning seems to be a promising approach for reusing trained models, both encoders and policies in a zero-shot fashion, showing compatibility between neural modules across slight environment variations. Further in-deep studies remain to be done, starting from the analysis of the latent space across different algorithms and environments. Then, it remains to be seen whether relative representations could be used to stitch modules across different tasks on environment where dynamics remain substantially the same. We would reuse a trained encoder and stitch different policies trained on various tasks.

References

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 225–236, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/01ded4259d101feb739b06c399e9cd9c-Abstract.html`.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Adrian Csiszarik, Peter Korosi-Szabo, Akos K. Matszangosz, Gergely Papp, and Daniel Varga. Similarity and matching of neural network representations. *ArXiv preprint*, abs/2110.14633, 2021. URL `https://arxiv.org/abs/2110.14633`.

Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. Towards reusable network components by learning compatible representations. *AAAI*, 35(9):7620–7629, 2021.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 991–999. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298701. URL https://doi.org/10.1109/CVPR.2015.7298701.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SrC-nwieGJ.

Antonio Norelli, Marco Fumero, Valentino Maiorca, Luca Moschella, Emanuele Rodolà, and Francesco Locatello. ASIF: Coupled Data Turns Unimodal Models to Multimodal Without Training. *ArXiv preprint*, abs/2210.01738, 2022. URL https://arxiv.org/abs/2210.01738.

Muammer Y. Yaman, Sergei V. Kalinin, Kathryn N. Guye, David Ginger, and Maxim Ziatdinov. Learning and predicting photonic responses of plasmonic nanoparticle assemblies via dual variational autoencoders. *ArXiv preprint*, abs/2208.03861, 2022. URL https://arxiv.org/abs/2208.03861.