# Zero-shot stitching in Reinforcement Learning using Relative Representations

**Antonio Pio Ricciardi**[*]
Department of Computer Science
Sapienza University of Rome
`ricciardi@di.uniroma1.it`

**Valentino Maiorca**
Department of Computer Science
Sapienza University of Rome
`maiorca@di.uniroma1.it`

**Luca Moschella**
Department of Computer Science
Sapienza University of Rome
`moschella@di.uniroma1.it`

**Emanuele Rodolà**
Department of Computer Science
Sapienza University of Rome
`rodola@di.uniroma1.it`

## 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 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, we would like to have model trained on a certain task, freeze its policy and swap its encoder with another, so that the policy could 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, by using this method it is possible to perform zero-shot stitching between encoders and policies trained on slight variations of the same environment. This enables the reuse of neural components and could be an initial step towards model stitching across different tasks. We performed our preliminary experiments using a modified version of the CarRacing environment (Figure 1). This modified environment features a discrete action space and allows for background color variations. We train our end-to-end agents, consisting of a standard CNN-based feature extractor and a policy. Empirical results then demonstrate that by employing 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.

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

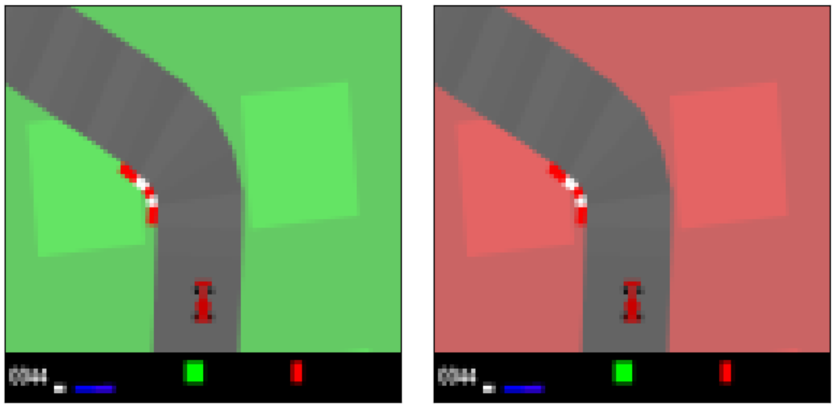

Figure 1: The modified version of the Car Racing environment, where we can change the color of the background.

## 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 rgb 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 $\mathcal{X}$ is the state space (a single rgb image), $\mathcal{A}$ 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 representations, as proposed by [Moschella et al., 2023], offer 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. Each sample is represented by the distances (or similarities) between its embedding and each of the anchor embeddings.

More formally, let us be given an encoder $E : \mathcal{X} \to \mathbb{R}^k$, a set of anchors $\mathcal{A} \subset \mathcal{X}$, and the row-wise matrix representation of the samples $\mathbf{X}$ and the anchors $\mathbf{A}$. By choosing cosine similarity as the similarity function, the relative representations $\mathbf{Z}_{rel}$ of the samples $\mathbf{X}$ are defined as:

$$\mathbf{Z}_{rel} = E(\mathbf{X})E(\mathbf{A})^T \, , \tag{2}$$

where the embeddings produced by $E$ are rescaled to unit norm.

Employing relative representations in a latent space, highlights its inherent structure. Under the assumption that good models learn similar representations it enables the latent communication between seemingly dissimilar spaces due to confounding factors.In this work, we apply relative representations to remove the following factors: i) the background color of the observations; ii) the training algorithm for the policy; iii) the random seed used for training.

## 3 Method

For our method, 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, training the policies directly on the relative representations.

Therefore, given two agents, denoted as $A$ and $B$, we define their associated encoders $E_A$ and $E_B$, respectively. An encoder takes an observation $x$ as input and produces a representation $E(x)$, that is converted into its relative form $z_{rel}$ by the transformation:

$$z_{rel} = E(x)E(\mathbf{A})^T \, . \tag{3}$$

Similarly, we define the two policies $\pi_A$ and $\pi_B$. A policy takes the representation $z_{rel}$ and generates an action $a$:

$$a \sim \pi(z_{rel}) \, . \tag{4}$$

In this work, we show how encoders and policies that were trained *independently* can be used jointly, and without any fine-tuning, to define a zero-shot stitched agent. For example, by using the encoder of agent $B$ and the policy of $A$, one gets the stitched agent:

$$\pi_A(E_B(x)E_B(\mathbf{A})^T) \, . \tag{5}$$

Alternatively, one gets a different agent by stitching the encoder of $A$ with the policy of $B$, namely $\pi_B(E_A(x)E_A(\mathbf{A})^T)$. We remark that the input distributions of $E_A$ and $E_B$ may be different, as long as they are semantically related (e.g., different background colors).

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

### 3.1 Anchors selection

Anchors are an essential component of the relative representations. In their paper [Moschella et al., 2023], the authors demonstrate that uniformly sampling anchors from the training dataset suffices. In reinforcement learning, where datasets are not normally used, we needed to come up with a different approach for anchors sampling, therefore we ran several episodes using a previously trained agent, gathering observations. Although using a random agent was an option, we aimed to ensure that a substantial portion of the observations gathered contained different variations of the track (straights, turns..). From the pool of collected data, we randomly sample $n_dim$ anchors, that is the flattened output dimension of the encoder.

## 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, the seed or the training algorithm used.

### 4.1 Training

We trained policies using the CleanRL implementation of PPO [Huang et al., 2022], using the default hypermarameters for both absolute and relative representations, with training curves being shown in Image 2. Training with relative representations demonstrates comparable performance with only a minor drop at the beginning, probably due to the continuously evolving anchors representations.

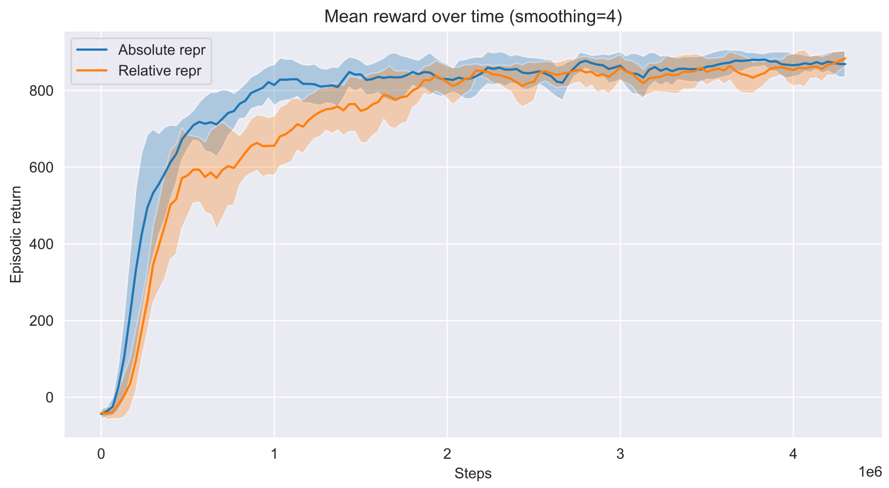

Figure 2: Training curves comparing absolute to relative representations. Relative training shows competitive performance with only a minor drop at the beginning.

### 4.2 Policy Stitching Evaluation

After training different agents on different background colors in the modified Car Racing environment, we froze the encoders and policies of each agent and evaluated their performance on different tracks, stitching for every combination of the background colors. Chosen colors are: *green*, *red*, *blue*, *yellow*. The encoder is always chosen according to the actual track background color. So, a green encoder is always tested on a track with a green background. Tests are performed for every color and over 5 track seeds, each seed corresponding to a different track. Table 1 compares the scores obtained when evaluating agents with relative and absolute stitching. In each cell of the table, we present the average return and standard deviation obtained over the 5 tracks. It's important to note that the recorded score represents the maximum return achieved during a run. In cases where the agent incurred penalties for going off-road and potentially missed some checkpoints, the score was not reduced. This approach accounts for situations in which an episode did not finish due to missed checkpoints although the agent drove well enough. Although suffering from higher variance, the scores obtained using relative representations remain consistent among almost 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.
Every agent was trained using different seeds, meaning that models' weights had a different initialization.

| | | Policy | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | `green` | | `red` | | `blue` | | `yellow` | |
| | | *Rel* | *Abs* | *Rel* | *Abs* | *Rel* | *Abs* | *Rel* | *Abs* |
| **Encoder** | `green` | 714 ± 288 | 863 ± 109 | 840 ± 94 | 7 ± 0.7 | 870 ± 84 | 26 ± 4 | 685 ± 237 | 12 ± 4 |
| | `red` | 774 ± 275 | 19 ± 3.8 | 692 ± 251 | 829 ± 116 | 288 ± 105 | 7 ± 0.7 | 638 ± 235 | 7 ± 0.7 |
| | `blue` | 256 ± 163 | 7 ± 1 | 307 ± 124 | 7 ± 0.6 | 690 ± 200 | 759 ± 288 | 556 ± 71 | 8 ± 3 |
| | `yellow` | 713 ± 204 | 7 ± 0.7 | 678 ± 81 | 33 ± 4 | 167 ± 66 | 26 ± 4 | 675 ± 233 | 874 ± 85 |

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. Same color pairs describe the original, non-stitched model's performance. Larger scores are better. High variance is due to different tracks difficulty.

## 4.3 Latent space analysis

We conducted a latent space analysis to compare the features produced by absolute and relative encoders. To collect environment observations, we used the same action sequence from an independent model to complete an episode, for two different background colors. This ensured that the collected observations were independent of the models being compared. Specifically, we saved observations $x_i^{(1)}$ and $x_j^{(2)}$, for two different background colors, and where $i$ and $j$ range from 0 to T, the total length of an episode, originating from the same environment track with the same seed but different background colors. Despite the color difference, the interactions performed in both cases were identical, resulting in images that differed only in background color when $i == j$.

Each figure in our analysis depicts the pairwise similarity between the representations of two encoders, denoted as $E^{(1)}$ and $E^{(2)}$, by comparing $E^{(1)}(x_i^{(1)})$ and $E^{(2)}(x_j^{(2)})$. A higher value along the diagonal of the figure indicates that the representations produced by the two encoders are highly 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 4 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 5 shows comparison on green (5.a) and red (5.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.

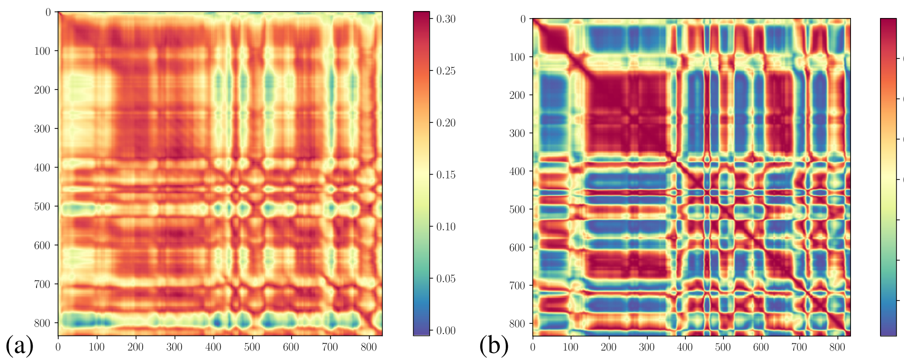

Figure 3: Similarity matrices comparing the similarity between the green background representations and the red background representations. (**a**) shows the absolute latent spaces, while (**b**) the relative latent spaces. The same seed is used to train these models, meaning that they had the same initialization.

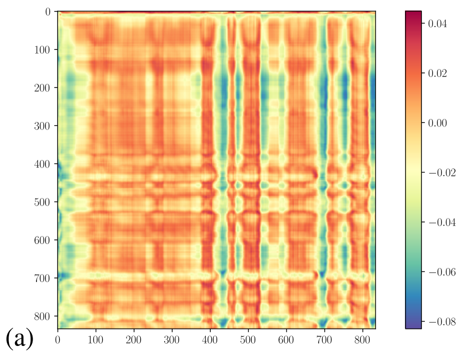
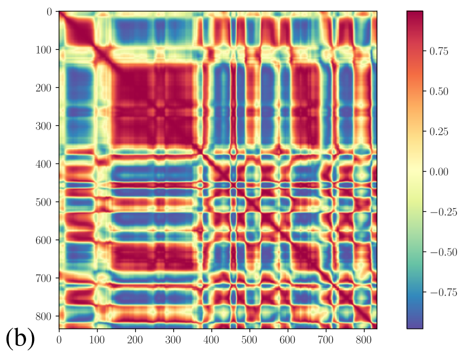

Figure 4: Similarity matrices comparing the similarity between the green background representations and the red background representations. (**a**) shows the absolute latent spaces, while (**b**) the relative latent spaces. Each matrix shows comparison between encoders trained on different seeds.

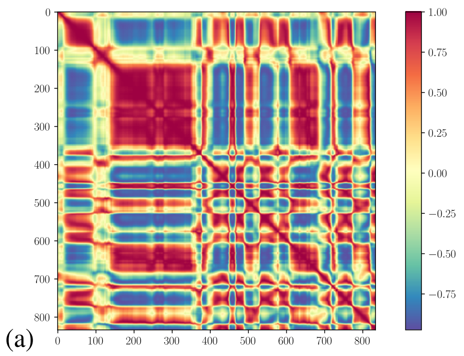
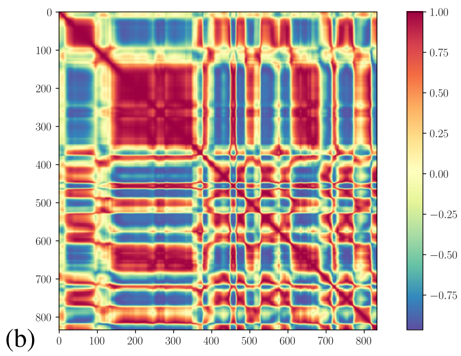

Figure 5: Plot comparing relative representations. **a**) shows the similarity between PPO and DDQN representations on the green background, while (**b**)shows the similarity between PPO and DDQN representations on the red background.

## 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, so that we could reuse a trained encoder by stitching different policies trained on various tasks.

**Acknowledgments** We gratefully acknowledge the financial support provided by the following organizations for this project:
European Research Council (ERC) Grant No. 802554 for the SPECGEO project.
PRIN 2020 Project No. 2020TA3K9N for LEGO.AI.
PNRR MUR Project PE0000013-FAIR.
This funding has been instrumental in the successful execution of our research endeavors.

## 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.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

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`.