

Mapping representations in Reinforcement Learning via Semantic Alignment for Zero-Shot Stitching

Antonio Pio Ricciardi¹, Valentino Maiorca¹, Luca Moschella¹, Riccardo Marin², Emanuele Rodolà¹

ricciardi@di.uniroma1.it

¹Sapienza, University of Rome

²University of Tübingen, Germany

Abstract

Deep Reinforcement Learning (RL) models often fail to generalize when even small changes occur in the environment’s observations or task requirements. Addressing these shifts typically requires costly retraining, limiting the reusability of learned policies. In this paper, we build on recent work in semantic alignment to propose a zero-shot method for mapping between latent spaces across different agents trained on different visual and task variations. Specifically, we learn a transformation that maps embeddings from one agent’s encoder to another agent’s encoder without further fine-tuning. Our approach relies on a small set of “anchor” observations that are semantically aligned, which we use to estimate an affine or orthogonal transform. Once the transformation is found, an existing controller trained for one domain can interpret embeddings from a different (existing) encoder in a zero-shot fashion, skipping additional trainings. We empirically demonstrate that our framework preserves high performance under visual and task domain shifts. We empirically demonstrate zero-shot stitching performance on the CarRacing environment with changing background and task. By allowing modular re-assembly of existing policies, it paves the way for more robust, compositional RL in dynamically changing environments.

1 Introduction

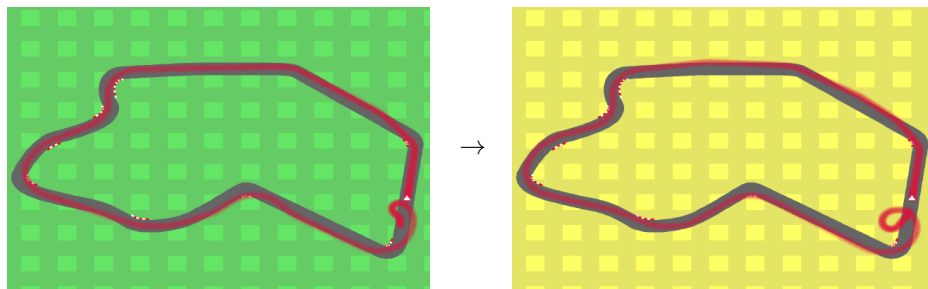


Figure 1: Using translation methods, a controller trained on an environment with a given visual variation (*left*) can be reused without any training or fine-tuning on a different environment (*right*) with comparable performance. In red we see the trajectory of a car driven by the same controller when connected to two different encoders, one for each visual variation.

Deep Reinforcement Learning (RL) has enabled agents to achieve remarkable performance in complex decision-making tasks, from robotic manipulation to high-dimensional games (Mnih et

al., 2015; Silver et al., 2017). Although recent RL techniques achieved strong improvements over sample efficiency (Yarats et al., 2021; Kostrikov et al., 2020), training new agents remains a costly process, both in computational and temporal terms. Despite these advances, most methods still require at least partial retraining when dealing with domain shifts such as visual appearance, reward functions, or action spaces (Cobbe et al., 2019; Zhang et al., 2020). These domain changes typically require expensive retraining, which can be prohibitive for real-world settings that require millions of interactions.

A variety of approaches have been proposed to address these shifting conditions. Domain randomization (Tobin et al., 2017; Sadeghi & Levine, 2016) trains agents across diverse visual styles or physics settings, promoting invariant features but demanding broader coverage of possible variations. Multi-task RL (Parisotto et al., 2015; Teh et al., 2017) attempts to learn shared representations across multiple tasks.

In the supervised setting, recent representation learning techniques (Moschella et al., 2023; Maiorca et al., 2023; Norelli et al., 2022; Cannistraci et al., 2023), show that it is possible to zero-shot recombine encoders and decoders to perform new tasks across different modalities (images, text..) and tasks (classification, reconstruction) and even architectures. In RL, methods adopting the relative representation framework (Moschella et al., 2023) have shown promising results in adapting encoders to different controllers with zero or few-shots adaptation, for robotic control from proprioceptive states (Jian et al., 2021) or for playing games in the Gymnasium suite (Towers et al., 2024) from pixels (Ricciardi et al., 2025). These methods, however, still require training models to use the new relative representations.

By contrast, Maiorca et al. (2023) suggest that modules from independently trained neural networks can be connected via a simple linear or affine transformation, with no training constraint or fine-tuning required, if such transformations can be reliably estimated from a small set of “anchor” samples, pairs of states or observations deemed semantically equivalent.

Our main contribution is the implementation of a RL method based on semantic alignment to map between latent spaces of different neural models, so that their encoders and controllers can be stitched with the goal of creating new agents that can act on visual-task combinations never seen together in training. This includes the use of the transformations to map modules from different networks, and the collection of anchor samples used to estimate these transformations. We call our method Semantic Alignment for Policy Stitching (SAPS). We perform analyses and empirical tests on the CarRacing and LunarLander environments to show the performance of new agents created via zero-shot stitching of encoders and controllers trained on different visual-task variations, demonstrating significant gains compared to existing zero-shot methods.

2 Related Work

Domain Adaptation and Generalization in RL One line of work tackling the mismatch between training and deployment environments focuses on domain adaptation. Early solutions often rely on domain randomization, which exposes agents to a broad range of visual or dynamical variations during training so that they learn features robust to such changes (Tobin et al., 2017; Sadeghi & Levine, 2016). Similarly, data augmentation techniques (Yarats et al., 2021; Laskin et al., 2020) modify raw input frames (e.g., random crops, color jitter) to improve out-of-distribution performance (Kostrikov et al., 2020; Yarats et al., 2021). However, these methods either demand extensive coverage of possible variations or can be extremely resource intensive.

Multi-task and Modular RL A common strategy to reuse knowledge across related tasks is multi-task or meta-RL, where a single agent is trained on multiple environments (Teh et al., 2017; Finn et al., 2017). While this can yield more robust representations, it often demands joint training on all tasks, which might be impractical when new tasks appear over time. Transfer RL (Taylor & Stone, 2009) instead aims to expedite learning in a target task by leveraging agents trained on a source task, reusing features or parameters (Barreto et al., 2017; Killian et al., 2017), value functions (Tirinzoni

et al. (2018); Liu et al. (2021), sub-policies (Fernández & Veloso, 2006; Devin et al., 2017). Other approaches exploit networks modularity, by designing agents as composition of planner and actuator modules (Karkus et al., 2020) or by combining submodules to solve harder tasks (Mendez et al., 2022; Russell & Zimdars, 2003; Simpkins & Isbell, 2019). These, however, require defining ad-hoc architectures to work. Our method instead focuses on zero-shot reuse of already existing models, while also eliminating the need for environment interaction or fine-tuning to adapt an already-trained module to a new domain.

Representation Learning Exploiting representation learning techniques is another approach to mitigate the complexity in RL. Some approaches propose to isolate visual or observational factors from task-dependent decision-making (Oord et al., 2018), or learning a robust policy given a source domain (Higgins et al., 2017). Other ignore task-irrelevant features via invariant encoders, using bisimulation metrics as training constraints (Zhang et al., 2020), using supervised learning to train inverse dynamics models (Hansen et al., 2020) or finetuning to match prior latent distributions (Yoneda et al., 2021). However, these approaches focus exclusively on visual variations, leaving shifts in the underlying task unaddressed. While such fine-tuning can be more efficient than learning from scratch, it still requires re-optimization whenever the environment changes. In contrast, our proposed approach does not need re-training or fine-tuning. Instead, it only needs to encode small transformation between existing latent spaces.

Model Stitching Model stitching refers to the process of combining separate neural modules (e.g. encoders, decoders, or intermediate layers from independently trained models) to create a new, fully functional model. Initially studied in supervised learning as means to measure latent space similarity across different models, (Lenc & Vedaldi, 2015; Bansal et al., 2021; Csiszarik et al., 2021), models stitching recently is being used for zero-shot model reuse (Moschella et al., 2023; Norelli et al., 2022; Maiorca et al., 2023; Cannistraci et al., 2023) by using sets of parallel data called *anchors*, which establish a semantic correspondence between models. These are used in the relative representation (Moschella et al., 2023) framework to project latent spaces to a common space, and the semantic alignment framework (Maiorca et al., 2023) to directly estimate a mapping between latent spaces of different models.

Recently, the relative representation framework has been used to perform stitching between encoders and controllers in the context of vision-based imitation learning (Jian et al., 2024), while in RL it has been used to perform few-shot and zero-shot stitching in robotic control from low dimensional proprioceptive states (Jian et al., 2023) or control from pixels in the Gymnasium suite (Ricciardi et al., 2025). Although powerful, these methods require training or fine-tuning models or the decoder, so that it learns to act given “relative” inputs, while also requiring additional training constraints to not degrade end-to-end performance. By contrast, our approach applies the idea of creating a mapping between layers by using anchor samples (Maiorca et al., 2023) and is directly applicable to already trained models, learning a transformation that maps between modules of different policies without further constraints or fine-tuning.

3 Preliminaries

We assume the underlying environment is a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, R, P, \gamma)$, with state space \mathcal{S} , action space \mathcal{A} , input observations $o \in \mathcal{O}$ and the transition function $P : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ that defines a probability distribution over the next state given the current state and action. The function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{R}$ assigns rewards, and γ is the discount factor that reduces the importance of delayed rewards. The agent’s behavior is dictated by a policy $\pi : \mathcal{O} \rightarrow \mathcal{A}$, which receives an observation and selects an action at each state, and is trained to maximize the discounted returns $\mathbb{E} \left[\sum_{i=0}^{\infty} \gamma^i \mathcal{R}(\mathbf{s}_i, \mathbf{a}_i) \right]$.

3.1 Background

We are interested in scenarios where both training setups and agent behaviors can vary. We find it convenient to use the same notation introduced in R3L (Relative Representations for Reinforcement Learning) (Ricciardi et al., 2025) to formalize these variations.

Environment variations We denote an environment by $\mathcal{M}^i = (\mathcal{O}_u, T_i)$. Here, \mathcal{O}_u is the distribution of observations o_u , and $T_i : \mathcal{S}^i \times \mathcal{A}_i \times \mathcal{R}_i \times \mathcal{P}^i \mapsto \mathcal{R}_i$ specifies the task. Two environments can differ in the distribution of observations (e.g., background color, camera perspective) or in the task itself (e.g., transition dynamics, action spaces, reward definitions). Since agents must discover the task solely through reward feedback, any shift—whether in observations or task—can significantly alter their learned representations.

Agents Following R3L, each policy π_u^i is typically obtained by end-to-end training on \mathcal{M}^i . However, we emphasize a modular view of this policy:

$$\pi_u^i(o_u) = \psi_u^i[\phi_u^i(o_u)] = \psi_u^i(\mathbf{x}_u^i) \quad (1)$$

where $\phi_u^i : \mathcal{O}_u \mapsto \mathcal{X}_u^i$ serves as the *encoder* that processes raw observations (e.g., images), and $\psi_u^i : \mathcal{X}_u^i \mapsto \mathcal{A}_i$ is the *controller* that outputs actions based on the latent embedding \mathbf{x}_u^i . This factorization disentangles observation-specific features (in ϕ_u^i) from task-specific decision rules (in ψ_u^i).

Latent Representation Now consider a second environment $\mathcal{M}^j = (\mathcal{O}_v, T_j)$, where \mathcal{O}_v differs from \mathcal{O}_u only in visual style (e.g., a shifted color scheme), and suppose we have a policy π_v^j trained on that environment. For two semantically corresponding observations $o_u \in \mathcal{O}_u$ and $o_v \in \mathcal{O}_v$, the respective latent embeddings differ:

$$\phi_u^i(o_u) \neq \phi_v^j(o_v) \implies \mathbf{x}_u^i \neq \mathbf{x}_v^j \quad (2)$$

In the next section, we describe how to map one latent space onto another to enable zero-shot stitching of encoders and controllers trained in different visual and task domains, without additional training.

4 SAPS: Semantic Alignment for Policy Stitching

Relative representations (Moschella et al., 2023), used as a base for zero-shot stitching in R3L, involve computing a distance function between a set of samples, called “anchors”, to project the output of each encoder to a shared latent space, enabling the subsequent training of a universal policy. Semantic alignment, instead, estimates a direct mapping between latent spaces.

Consider the environment \mathcal{M}_u^j for which no dedicated policy exists. However, we do have an encoder ϕ_u^i and a controller ψ_v^j , extracted from policies π_u^i and π_v^j , respectively. We estimate an affine transformation $\tau_u^v : \mathcal{X}_u^i \mapsto \mathcal{X}_v^j$, mapping embeddings produced by ϕ_u^i into the space of π_v^j . This yields a new latent space:

$$\tau_u^v(\phi_u^i(o_u)) \approx \phi_v^j(o_v) \quad (3)$$

$$\tau_u^v(\mathbf{x}_u^i) \approx \mathbf{x}_v^j \quad (4)$$

that is compatible with the existing ψ_v^j . This enables the stitching of encoders and controllers from π_u^i and π_v^j , respectively, to obtain a new policy $\tilde{\pi}_u^j$ that can act in \mathcal{M}_u^j , without additional training:

$$\tilde{\pi}_u^j(o_u) = \psi_v^j[\tau_u^v(\phi_u^i(o_u))] \quad (5)$$

Estimating τ As in Maiorca et al. (2023), assume to be given latent spaces \mathbf{X}_u and \mathbf{X}_v which here correspond to the embedding of two visual variations in the space of observations. We use SVD to obtain an affine transformation $\tau_u^v(\mathbf{x}_u) = \mathbf{R}\mathbf{X}_u + \mathbf{b}$.

Collecting the Dataset. The anchor embeddings \mathbf{X}_u and \mathbf{X}_v derive from sets of anchor points \mathbf{A}_u and \mathbf{A}_v . Following previous works (Maiorca et al., 2023; Moschella et al., 2023; Ricciardi et al., 2025) anchor pair $(\mathbf{a}_u, \mathbf{a}_v)$ must share a semantic correspondence, meaning both samples represent the same underlying concept (e.g., the same spatial position in a racing track, viewed under two different visual styles). In supervised learning contexts, anchor pairs can come from paired datasets (e.g., bilingual corpora). In the context of online RL, however, such datasets do not naturally exist. Hence, we collect datasets sharing a correspondence. This correspondence can be obtained by either rolling out a policy and replaying the same set of actions with different visual variations, as already done in Jian et al. (2023); Ricciardi et al. (2025), or by simply applying visual transformations to the image in pixel space. This yields corresponding observation sets \mathbf{A}_u and \mathbf{A}_v that can be embedded by each domain’s encoder to create \mathbf{X}_u and \mathbf{X}_v . Finally, we solve for τ_u^v using the SVD-based procedure above.

In our context, we assume that an agent trained end-to-end to solve a specific task in a specific environment will generate a comprehensive set of observations, providing a reasonable approximation of the entire latent space. Nevertheless, forcing the agent to explore more could be beneficial in this context. In our experiments, we gather parallel samples either by directly translating the observation in pixel space, when there is a well-defined known visual variation between the environments, or by replaying the same sequence of actions in both environments, that in this case must be deterministic and initialized with the same random seed. We leave to future research other possible approximation techniques for translating observations between different environments.

5 Experiments

We now evaluate SAPS using both qualitative and quantitative analyses. We first compare its zero-shot performance to R3L on benchmark tasks, then an analysis of how our alignment approach behaves under different conditions.

Environments Our agents act by receiving pixel images as input observation, consisting of four consecutive 84×84 RGB images, stacked along the channel dimension to capture dynamic information such as velocity and acceleration. We consider environments where we can freely change visual features (background color, camera perspective) or task (rewards, dynamics), therefore we use CarRacing (Klimov, 2016) and LunarLander as both implemented in R3L. CarRacing requires the agent to drive in a track using pixel observations, whose variations can be in the background color or the target speed, while LunarLander requires the agent to land on a platform, with variations comprising background color and different gravities. No context is provided, hence the agents do not receive any information about the task.

Baselines We mainly compare SAPS to (R3L), another zero-shot stitching method using relative representations whose approach is similar to ours. For an additional baseline we also compare to naive zero-shot stitching, where we stitch encoders and controllers with no additional processing, to showcase the progress reached by the methods performing latent alignment techniques.

5.1 Zero-shot stitching comparison

We follow the empirical analysis performed in R3L. We define the encoder as the network up to the first flatten layer after the convolutional blocks, with the remaining layers constituting the controller. The zero-shot stitching evaluation is conducted on visual-task variations that were not seen together during training. Encoders and controllers must match to the specific conditions they were trained on.

Table 1: Stitching performance in **CarRacing**, comparing **SAPS (ours)** to other methods, averaged over 5 seeds, with standard deviations.

		Controller						
		Visual Variations (task standard)			Task Variations (green)			
		green	red	blue	slow	scrambled	no idle	
Encoder	green	Naive	175 ± 304	167 ± 226	−4 ± 79	148 ± 328	106 ± 217	213 ± 201
		R3L	781 ± 108	787 ± 62	794 ± 61	268 ± 14	781 ± 126	824 ± 82
		SAPS (ours)	822 ± 62	786 ± 82	829 ± 49	764 ± 287	846 ± 66	781 ± 72
	red	Naive	157 ± 248	43 ± 205	22 ± 112	83 ± 191	138 ± 244	252 ± 228
		R3L	810 ± 52	776 ± 92	803 ± 58	476 ± 430	790 ± 72	817 ± 69
		SAPS (ours)	859 ± 41	807 ± 52	809 ± 60	824 ± 192	838 ± 52	853 ± 50
	blue	Naive	137 ± 225	130 ± 274	11 ± 122	95 ± 128	138 ± 224	144 ± 206
		R3L	791 ± 64	793 ± 40	792 ± 48	564 ± 440	804 ± 41	828 ± 50
		SAPS (ours)	839 ± 57	808 ± 70	814 ± 52	746 ± 319	832 ± 60	808 ± 62
	far	Naive	152 ± 204	65 ± 180	2 ± 152	−49 ± 9	351 ± 97	349 ± 66
		R3L	527 ± 142	605 ± 118	592 ± 86	303 ± 100	594 ± 39	673 ± 91
		SAPS (ours)	714 ± 45	712 ± 71	727 ± 52	762 ± 131	738 ± 44	626 ± 77

For example, an encoder trained with a green background should be used in such an environment, and a controller developed for low-speed driving should be applied to that task.

Stitching table Encoders and controllers from different policies trained under different conditions, can be independently assembled through zero-shot stitching. In Section 4 we defined how to estimate and use a transformation to then perform the stitching, defining the models as formed by encoders and controllers.

We present zero-shot stitching results in Table 1 and Table 2. The testing procedure to perform stitching is as follows: Given models trained with various seeds and different visual-task variations, we decompose encoders and controllers of a model, then connect each encoder with controllers that were trained either on different visual variations or different seeds, using the transformation to ensure compatibility between layers. Each cell reports the average score obtained across different stitched seeds. Therefore, if we have four visual variations, six task variations and five seeds, for each cell we perform $6 \times 5 = 30$ stitching tests, for a total of 120 across the entire table for a single stitching method. Each combination is tested over 10 different tracks.

In CarRacing, SAPS consistently achieves performance comparable to end-to-end scores across all tested visual and task variations, even in the challenging slow scenario where R3L experiences a noticeable drop. In LunarLander there is a considerable drop in the mean average performance, which is mainly due to some models highly underperforming, decreasing the mean value. This is, however, a strong improvement over R3L, for which we were not even able to train models, strongly highlighting the advantages of being able to perform zero-shot stitching from already trained, standard models. This table underscores SAPS’s robustness and adaptability in assembling agents for new environment variations, surpassing existing methods.

SAPS instead performs much worse in LunarLander with respect to end-to-end agents, although improving upon the naive stitching approach. We think that this might reflect how sensitive LunarLander is to even small latent-space mismatches: a slight misalignment in the latent space could cause the landing procedure to strongly deviate in its trajectory, causing crashes and large penalties. Here, R3L scores are missing because we were not able to train models using this method.

Table 2: Stitching performance comparing **SAPS (ours)** to other methods, in the **LunarLander** environment. Scores are averaged over 5 seeds, with standard deviations. For comparison, end-to-end models achieve 221 ± 86 for the white background, 192 ± 30 for the red background with gravity -10; for the white background, for gravity -3. R3L results are absent because we were not able to train models for it.

			Controller		
			Gravity: -10		Gravity: -3
			white	red	white
Enc	Gravity -10	red			
		white			
		<i>Naive</i>	-413 ± 72	-390 ± 176	-276 ± 8
		SAPS (ours)	19 ± 56	8 ± 60	-242 ± 51
Enc	Gravity -3	red			
		white			
		<i>Naive</i>	-444 ± 116	-403 ± 109	-271 ± 18
		SAPS (ours)	52 ± 44	33 ± 61	-204 ± 71

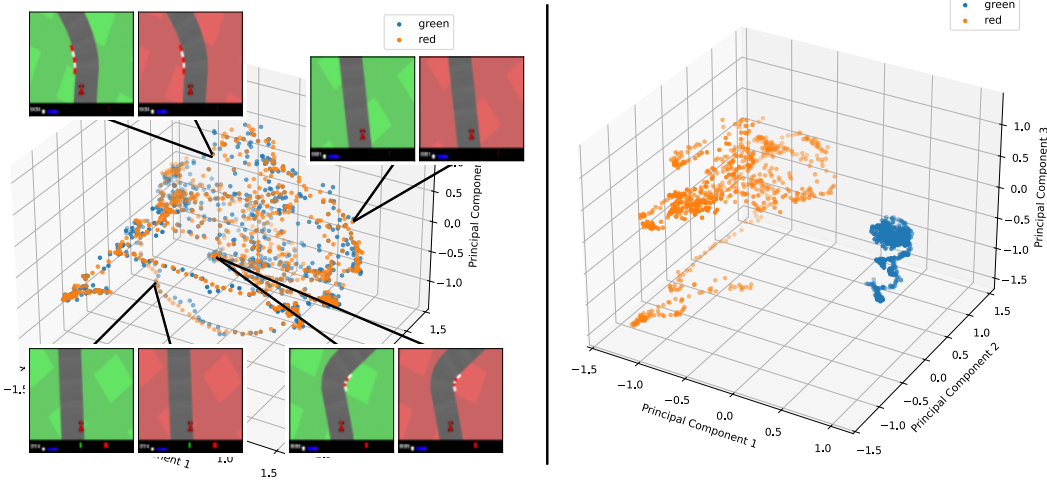


Figure 2: PCA visualization of encoder outputs. On the left, we illustrate how an affine alignment can effectively map one latent space to another: same frames with different backgrounds (green/red) cluster together, as indicated by the embedded screenshots. On the right, the source, unaligned embeddings remain separated, highlighting the benefit of our alignment approach in unifying observations from different environment variations.

5.2 Latent Space Analysis

To assess how effectively our proposed method (SAPS) aligns latent representations across different visual or task variations, we perform both qualitative and quantitative evaluations.

Qualitative Visualization. Figure 2 (left) presents a 3D PCA projection of latent embeddings for two CarRacing variations: green and red backgrounds. After applying an affine transformation learned by SAPS to the encoder outputs, the points corresponding to green and red observations become thoroughly intermixed in the shared latent space. This intermixing indicates that frames depicting the “same portion” of the track with different background colors now map to similar embeddings (see insets). Conversely, Figure 2 (right) shows the unaligned embeddings, where green and red remain clearly separated. These results suggest that SAPS successfully bridges the gap between encoders, producing a unified representation space under visual variation.

Quantitative Similarities. In Figure 3, we plot histograms of pairwise cosine similarity for matched frames from two different variations. Again, we compare (a) SAPS, (b) R3L and (c) Naive stitching. The top row shows CarRacing, while the bottom row is LunarLander. In both environments, SAPS

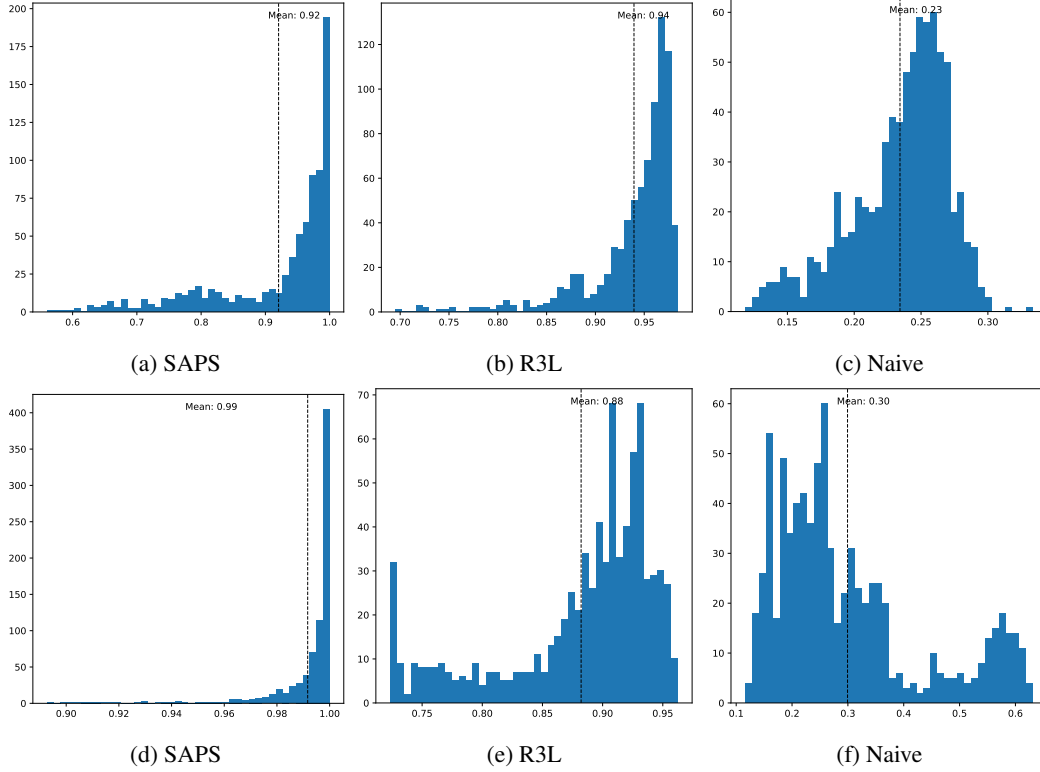


Figure 3: Histogram of pairwise cosine similarities between matched states from two different environment variations, for CarRacing (**top**) and LunarLander (**bottom**). Both SAPS and R3L show very high mean similarity along paired frames, indicating that corresponding observations in each variation map to nearly identical vectors. Mean similarity for encoders without any alignment or relative encoding is very low, emphasizing the utility of latent communication methods.

and R3L achieve a much higher mean cosine similarity (e.g., 0.92–0.99) than the Naive baseline (0.23–0.30). This confirms that independently trained models can exhibit near-identical encodings for semantically identical frames once aligned, while “naive” combinations of encoders and controllers remain incompatible.

Discussion. Overall, these findings indicate that: (i) SAPS’ affine transformation effectively repositions points in the latent space, causing corresponding frames to map to nearly the same vector (Figures 2–3); (ii) Compared to “Naive” reusability (no alignment) or purely relative approaches (R3L), SAPS achieves equivalent or better alignment without retraining models on a specialized representation format; (iii) The high average cosine similarity under SAPS confirms that visual variations (and, by extension, moderate task changes) can be handled by learning a lightweight transform from one latent space to another.

Hence, our latent space analysis demonstrates that SAPS successfully stitches together components from different RL models to produce a cohesive, unified representation, paving the way for zero-shot policy reuse in previously unseen environment variations.

6 Conclusion and Future Directions

Conclusion We presented Semantic Alignment for Policy Stitching (SAPS), a simple yet effective method for zero-shot reuse of RL agents trained in different environments. By estimating a lightweight transformation that maps one encoder’s latent space into another’s, we can seamlessly “stitch”

encoders and controllers, enabling new policies to handle unseen combinations of visual and task variations without retraining. Our experiments on CarRacing and LunarLander show SAPS achieves near end-to-end performance under diverse domain shifts, outperforming naive baselines and matching or exceeding more specialized zero-shot approaches (e.g., R3L) in many settings. This highlights the potential of direct latent-space alignment for compositional and robust RL.

Limitations and Future Works. Although SAPS effectively aligns latent representations across moderate environment variations, several open challenges remain. First, the method’s reliance on affine transformations can falter in domains exhibiting larger gaps (for instance, tasks that differ drastically in reward structure or observation type) where a simple linear map may be insufficient. Second, the approach depends on anchors, which must be collected from both source and target environments; in highly stochastic domains, obtaining robust correspondences can be nontrivial. Third, the method was tested on relatively constrained settings, leaving it unclear how well it scales to real-world robotics or continuous domains with high state space complexity. Possible extensions include (i) automating or relaxing the anchor-collection procedure to handle more diverse or partially observable environments, and (ii) validating SAPS on robotics tasks with real sensors and complex dynamics, where retraining from scratch is particularly time-consuming.

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 (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 225–236, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/01ded4259d101feb739b06c399e9cd9c-Abstract.html>.
- André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- Irene Cannistraci, Luca Moschella, Marco Fumero, Valentino Maiorca, and Emanuele Rodolà. From bricks to bridges: Product of invariances to enhance latent space communication, 2023.
- Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/cobbe19a.html>.
- Adrian Csizsarik, 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>.
- Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.
- Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727, 2006.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.
- Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 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 *International conference on machine learning*, pp. 1480–1490. PMLR, 2017.
- Pingcheng Jian, Chao Yang, Di Guo, Huaping Liu, and Fuchun Sun. Adversarial skill learning for robust manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2555–2561. IEEE, 2021.
- Pingcheng Jian, Easop Lee, Zachary Bell, Michael M Zavlanos, and Boyuan Chen. Policy stitching: Learning transferable robot policies. *arXiv preprint arXiv:2309.13753*, 2023.
- Pingcheng Jian, Easop Lee, Zachary Bell, Michael M Zavlanos, and Boyuan Chen. Perception stitching: Zero-shot perception encoder transfer for visuomotor robot policies. *arXiv preprint arXiv:2406.19971*, 2024.
- Peter Karkus, Mehdi Mirza, Arthur Guez, Andrew Jaegle, Timothy Lillicrap, Lars Buesing, Nicolas Heess, and Theophane Weber. Beyond tabula-rasa: a modular reinforcement learning approach for physically embedded 3d sokoban. *arXiv preprint arXiv:2010.01298*, 2020.
- Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. *Advances in neural information processing systems*, 30, 2017.
- Oleg Klimov. Carracing-v0. URL <https://gym.openai.com/envs/CarRacing-v0>, 2016.
- Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.
- Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.
- 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*, pp. 991–999. IEEE Computer Society, 2015. DOI: 10.1109/CVPR.2015.7298701. URL <https://doi.org/10.1109/CVPR.2015.7298701>.
- Chenyu Liu, Yan Zhang, Yi Shen, and Michael M Zavlanos. Learning without knowing: Unobserved context in continuous transfer reinforcement learning. In *Learning for Dynamics and Control*, pp. 791–802. PMLR, 2021.
- Valentino Maiorca, Luca Moschella, Antonio Norelli, Marco Fumero, Francesco Locatello, and Emanuele Rodolà. Latent space translation via semantic alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=pBa70rGHlr>.
- Jorge A Mendez, Harm van Seijen, and Eric Eaton. Modular lifelong reinforcement learning via neural composition. *arXiv preprint arXiv:2207.00429*, 2022.
- 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>.

-
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.
- Antonio Pio Ricciardi, Valentino Maiorca, Luca Moschella, Riccardo Marin, and Emanuele Rodolà. R3l: Relative representations for reinforcement learning, 2025. URL <https://arxiv.org/abs/2404.12917>.
- Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 656–663, 2003.
- Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.
- Christopher Simpkins and Charles Isbell. Composable modular reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4975–4982, 2019.
- Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.
- Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distal: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- Andrea Tirinzoni, Rafael Rodriguez Sanchez, and Marcello Restelli. Transfer of value functions via variational methods. *Advances in Neural Information Processing Systems*, 31, 2018.
- Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.
- Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.
- Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.
- Takuma Yoneda, Ge Yang, Matthew R Walter, and Bradley Stadie. Invariance through latent alignment. *arXiv preprint arXiv:2112.08526*, 2021.
- Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.