Learning Robust Representations for Transfer in Reinforcement Learning

Faisal Mohamed **Roger Creus Castanyer** **Hongyao Tang** **Zahra Sheikhbahaee**

Glen Berseth

Université de Montréal and Mila Quebec AI Institute
`{faisal.mohamed, roger.creus-castanyer, tang.hongyao, zahra.sheikhbahae, glen.berseth}@mila.quebec`

Abstract

Learning transferable representations for deep reinforcement learning (RL) is a challenging problem due to the inherent non-stationarity, distribution shift, and unstable training dynamics. To be useful, a transferable representation needs to be robust to such factors. In this work, we introduce a new architecture and training strategy for learning robust representations for transfer learning in RL. We propose leveraging multiple CNN encoders and training them not to specialize in areas of the state space but instead to match each other's representation. We find that learned representations transfer well across many Atari tasks, resulting in better transfer learning performance and data efficiency than training from scratch.

1 Introduction

Transfer learning of pre-trained models in the context of reinforcement learning (RL) has shown to be more challenging than in the supervised learning case. While the goal of many foundational models is to create a good parameter initialization for future tasks, RL, and in general, sequential decision-making problems experience more distribution shifts and variance in the learning process. These issues give rise to the question of how can we pre-train models that will be more robust to future distribution changes while still maintaining plasticity.

The lack of easy transfer hints that in addition to using a method that should learn a good representation that is helpful across tasks, it is also to train this representation to be robust to future task-specific updates. In this work, we propose introducing a modified mixture of expert architecture **?** . Mixtures of experts are typically used to help handle large state spaces or multi-modal behavior. However, in this case, we use this structure to induce a type of structured gradient noise that assists in learning a representation that achieves faster fine-tuning performance. Motivated by finding communication channels that multuple models agree on to ensure robustness, we explore constructing a multiple encoder single decoder model where we force the encoders to share the same communication channel. This type of robust encoding is helpful because (1) the encoder may contain more task-general information than prior methods that can overfit spurious correlations and (2) be less susceptible to noisy gradients when components of the model are transferred to downstream tasks.

Our contribution is a new pre-training method for RL tasks. In particular, our simple method can be used with image-based environments to learn better network initialization for similar and downstream tasks. This learning process results in more efficient fine-tuning, improved data efficiency, and better final performance than an agent train end-to-end from scratch.

2 Related Work

Scaling deep RL Recent work on scaling up deep RL algorithm has exhibited large improvement in data efficiency, network's parameter utilization, and stability of learning dynamics. Tang and Berseth [2024] proposed a churn reduction method that improve learning performance and scaling abilities of deep RL, Willi et al. [2024] shed the light on the role mixture of experts and how they improve training under non-stationarity, Nikishin et al. [2022] discovered the primacy bias; a tendency to learn from interactions and ignore new experience in later training stages and introduced to mitigate this bias, which resulted in better scaling and utilization of the network parameters. Additional works [Abbas et al., 2023, Nauman et al., 2024, D'Oro et al., 2023, Lee et al., 2024, Nikishin et al., 2024] show that improving network's plasticity results in better scaling and data utilization, however these methods have not been used to pre-train a network and transfer that network to a similar or different task.

Pre-training for deep RL Pre-training vision based models for control has been investigated in various works, Kim et al. [2024] investigated the generalization capabilities of a resnet-50 model [He et al., 2016] pre-trained with numerous objectives in Atari environments, while gou [2023] studied the performance of pre-trained vision transformers. Majumdar et al. [2023] introduced a large-scale study of pre-trained vision models. Hu et al. [2023] showed the performance of a pre-trained vision model depends on the downstream learning algorithm, and not all algorithms perform equally for the same vision model. Baker et al. [2022]. Baker et al. [2022], Seo et al. [2022] proposed pre-training vision models from videos, which showed improvement in data efficiency and final performance. While the RL community has used these pre-trained models to accelerate transfer, the pre-training often reduces final policy performance. Instead, our work investigates how to get the benefits of pre-training without the cost of final policy performance by improving sample efficiency in the fine-tuning stage.

3 Background

In this section, we provide a very brief review of the fundamental background used by our method. reinforcement learning (RL) is formulated within the framework of a Markov Decision Processes (MDP) where at every time step t, the world (including the agent) exists in a state $\mathbf{s}_t \in \mathcal{S}$, where the agent is able to perform actions $\mathbf{a}_t \in \mathcal{A}$. The action to take is determined according to a policy $\pi(\mathbf{a}_a|\mathbf{s}_t)$ which results in a new state $\mathbf{s}_{t+1} \in \mathcal{S}$ and reward $r_t = R(\mathbf{s}_t, \mathbf{a}_t)$ according to the transition probability function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{s}_t)$. The policy is optimized to maximize the future discounted reward $\mathbb{E}_{r_0,\ldots,r_T}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where T is the max time horizon, and γ is the discount factor.

In transfer learning, an agent is trained on a source task(s) $\mathcal{T}_s$, then we used the trained agent for learning on new target task(s) $\mathcal{T}_t$, by task we mean an MDP, for example we train an agent on different atari games, where each game is represented as a different MDP $\mathcal{T}_i$, then we use the pre-trained agent for faster learning in new games. For effective transfer, we assume a common structure between the source and the target task.

4 Method

In this section, we describe our model architecture and training algorithm. Our method consists of two modifications: (1) a network architecture with a shared decoder network that forces the encoders to agree on a shared encoder representation and (2) a data mixing strategy to increase robustness and further force learning a shared representation.

Shared Encoders. In a base PPO agent with multiple encoders, these encoders are shared between the actor and the critic models. In our work we instead create multiple encoders as shown in Figure 1. The use of multiple encoders is motivated by the idea that any representation that many agents (encoders) can agree on must be a more reusable representation and will also be robust to communicating with newcomers (future tasks).

Data Mixing. To make the encoders more robust and avoid specialization of the models, we purposefully mix data generated by different encoders. In data collection, an encoder f_{ψ_i} is chosen

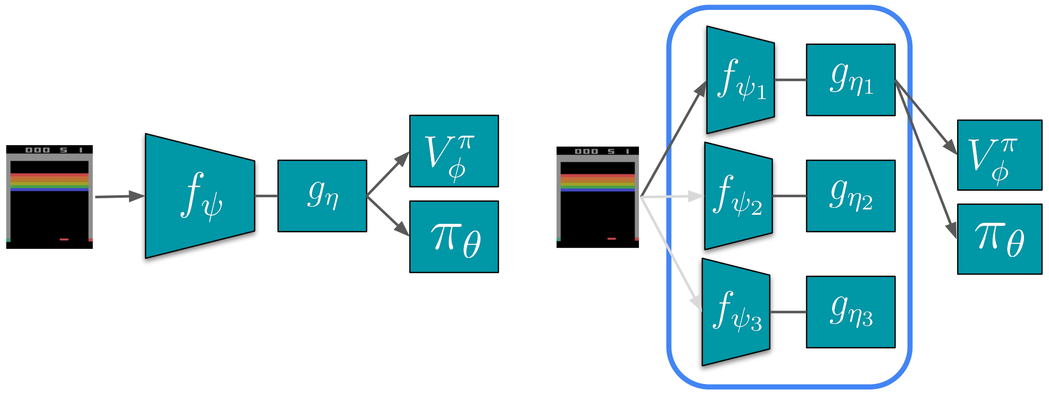


Figure 1: On the left is the original PPO architecture the image observation is passed through a shared CNN encoder that output latent representations to the policy and the value function. On the right we train the ppo agent with multiple encoders, an encoder is sampled randomly (the black arrow), while the rest of the encoders remains inactive (the gray arrows), the image observation is passed through the sampled encoder, and the output is passed to the policy and value networks.

randomly, and the transition data is stored in a shared rollout buffer. In the model update, we sample an encoder randomly at each update iteration, and we update the model based on the shared rollout buffer. An important factor is sharing the data across different encoders, which we show in Section 5 results in more efficient pre-training and fine-tuning compared to training each encoder only on its own data. Algorithm 1 describes the training procedure in detail.

Algorithm 1 Pre-training with multiple encoders

```
 1: Initialize CNN encoders parameters $\psi_i$, where $i \in \{1, \dots, N\}$, policy network parameters $\theta$,
    and value network parameters $\phi$ and rollout buffer $\beta$.
 2: **for** iteration $= 0, \dots, I$ **do**
 3:   Sample initial observation $s_o \sim p(s_0)$
 4:   Store $s_o$ in $\beta$
 5:   **for** $t = 0, \dots, T$ **do**                                   ▷ Data collection
 6:     Sample an encoder randomly $f_{\psi_i}, i \sim \mathcal{U}(N)$
 7:     Sample an action $a_t \sim \pi_\theta(a_t \mid f_{\psi_i}(s_t))$
 8:     $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
 9:     Store transition in $\beta$
10:   **end for**
11:   **for** epoch $= 0 \dots, E$ **do**                                ▷ Model update
12:     Sample an encoder randomly $f_{\psi_i}, i \sim \mathcal{U}(N)$
13:     Sample a minibatch $\mathcal{B} \sim \beta$
14:     Update model $f_{\psi_i}$, $\pi_\theta$, and $V^\pi_\phi$              ▷ See Schulman et al. [2017]
15:   **end for**
```

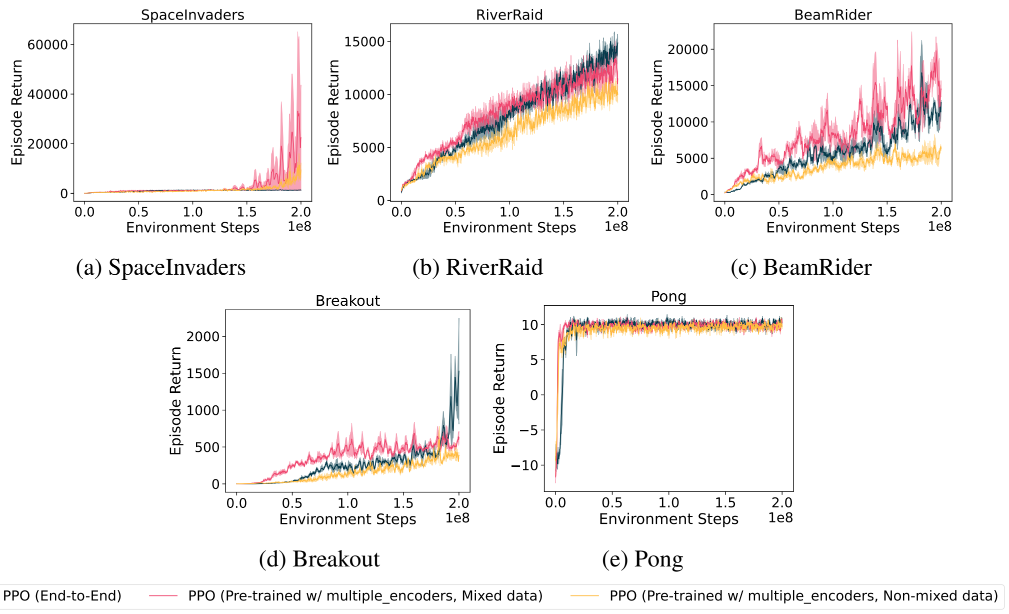
5 Experiments

To validate that our method learns useful representations and does not overfit to a particular training distribution, we experiment with same-task transfer $\mathcal{T}_i \rightarrow \mathcal{T}_i$ by pre-training the agent on an atari environment and transferring the encoders $f_{\psi_0}, \dots, f_{\psi_N}$ to the same environment (section 5.1). In section 5.2 we conduct transfer learning experiments, where the source environment is different than the target environment(s) $\mathcal{T}_i \rightarrow \mathcal{T}_j$. We use the episode return as the evaluation metric and as a baseline we compare to an agent trained from scratch end-to-end, also, we include the results where we don't mix the data coming from different encoders.

5.1 Same-task Transfer

We experiments with five atari environments; SpaceInvaders, RiverRaid, BeamRider, Breakout, and Pong we pre-train all agents for 200M time-steps, log the model weights, and reload them for re-training, we re-initialize the actor and critic networks, but fine-tune the CNN encoders.

Figure 2 shows the results for each environment, in five out of the six environments, our agent either matches or surpasses the end-to-end agent, which indicates that the learned representations are not overfitted to the training distribution in the pre-training process. In Breakout, the end-to-end agent outperformed both variants of our method, which might hint to some optimization challenges that need to addressed.



(a) SpaceInvaders (b) RiverRaid (c) BeamRider

(d) Breakout (e) Pong

Figure 2: In four out of the five environments (SpaceInvaders, RiverRaid, BeamRider, and Pong), our agent either matches or surpasses the end-to-end agent. In Breakout, the end-to-end agent outperformed both variants of our method.

5.2 Different-task Transfer

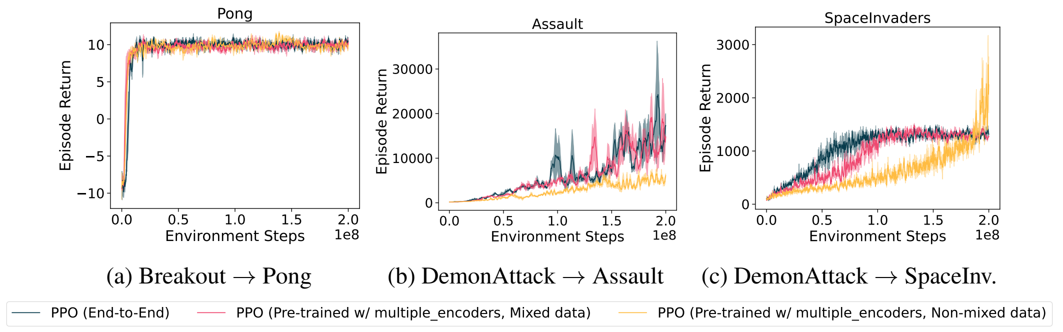
To show the effectiveness of our method, we conduct different-task transfer experiments, we pre-train two ppo agents, one on Breakout and the other on DemonAttack for 200M time-steps, the agent pre-trained on Breakout is transferred to Pong, while the agent pre-trained on DemonAttack is transferred to SpaceInvaders and Assault.

Figure 3 shows the results on each environment, in Pong both variants of our method solved the task efficiently and in fewer time-steps compared to the end-to-end agent, in Assault, the transferred agent (with mixed data) did not show any improvement compared to the end-to-end agent, while in SpaceInvaders the agent pre-trained with the non-mixed data variant outpeformed the two other agents.

In all environments, pre-training with mixed data collected from different encoders result in better transfer performance compared to the variant where each encoder is trained only on its own data(except for SpaceInvaders).

6 Conclusion and Discussion

This work introduced a new architecture and training strategy for transfer learning in RL. Interestingly the multiple encoders architecture and data mixing showed better transfer learning performance in Atari environments and improved data efficiency. This mixing is surprising because a large amount



(a) Breakout → Pong (b) DemonAttack → Assault (c) DemonAttack → SpaceInv.

Figure 3: Transfer learning performance, in Pong and Tennis, pre-training improved data efficiency and final performance. In Assault, pre-training did not introduce any additional benefits compared to the end-to-end agent, and in SpaceInvaders the agent pre-trained with non-mixed data had the best performance

of literature suggests that policy specialization is key to success and generalization [Peng et al., 2019, Chang et al., 2021, Eysenbach et al., 2018, Sutton et al., 1999, Bacon et al., 2017]. However, at least in the case of pre-training models for RL, our results indicate that models that are robust to other distributions are more reusable. In the future, we plan to evaluate our findings using more RL algorithms to strengthen the claims.

References

Pretraining the vision transformer using self-supervised methods for vision based deep reinforcement learning. 2023.

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 620–636. PMLR, 2023.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Michael Chang, Sid Kaushik, Sergey Levine, and Tom Griffiths. Modularity in reinforcement learning via algorithmic independence in credit assignment. In *International Conference on Machine Learning*, pages 1452–1462. PMLR, 2021.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron C. Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations*, 2023. URL `https://api.semanticscholar.org/CorpusID:259298604`.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Yingdong Hu, Renhao Wang, Li Erran Li, and Yang Gao. For pre-trained vision models in motor control, not all policy learning methods are created equal. In *International Conference on Machine Learning*, pages 13628–13651. PMLR, 2023.

Donghu Kim, Hojoon Lee, Kyungmin Lee, Dongyoon Hwang, and Jaegul Choo. Investigating pre-training objectives for generalization in vision-based reinforcement learning. *arXiv preprint arXiv:2406.06037*, 2024.

Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36: 655–677, 2023.

Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzcinski, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36, 2024.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in neural information processing systems*, 32, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Hongyao Tang and Glen Berseth. Improving deep reinforcement learning by reducing the chain effect of value and policy churn, 2024. URL https://arxiv.org/abs/2409.04792.

Timon Willi, Johan Obando-Ceron, Jakob Foerster, Karolina Dziugaite, and Pablo Samuel Castro. Mixture of experts in a mixture of rl settings. *arXiv preprint arXiv:2406.18420*, 2024.