2023 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-6891-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICME55011.2023.00474

P^{3}O: Transferring Visual Representations for Reinforcement Learning via Prompting

Guoliang You
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
glyou@mail.ustc.edu.cn

Xiaomeng Chu
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
cxmeng@mail.ustc.edu.cn

Yifan Duan
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
dyf0202@mail.ustc.edu.cn

Jie Peng
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
pengjieb@mail.ustc.edu.cn

Jianmin Ji
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
jianmin@ustc.edu.cn

Yu Zhang
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
yuzhang@ustc.edu.cn

Yanyong Zhang*
School of Computer Science and Technology
University of Science and Technology of China
Anhui, China
yanyongz@ustc.edu.cn

***Abstract*—It is important for deep reinforcement learning (DRL) algorithms to transfer their learned policies to new environments that have different visual inputs. In this paper, we introduce Prompt based Proximal Policy Optimization (P^{3}O), a three-stage DRL algorithm that transfers visual representations from a target to a source environment by applying prompting. The process of P^{3}O consists of three stages: pre-training, prompting, and predicting. In particular, we specify a prompt-transformer for representation conversion and propose a two-step training process to train the prompt-transformer for the target environment, while the rest of the DRL pipeline remains unchanged. We implement P^{3}O and evaluate it on the OpenAI CarRacing video game. The experimental results show that P^{3}O outperforms the state-of-the-art visual transferring schemes. In particular, P^{3}O allows the learned policies to perform well in environments with different visual inputs, which is much more effective than retraining the policies in these environments.**

***Index Terms*—Visual Transfer, Reinforcement Learning, Imitation Learning, Prompting Method**

I. Introduction

Deep Reinforcement Learning (DRL) has been applied to a large set of applications, including games, robotics, self-driving [1]–[4], etc. However, it is challenging for DRL algorithms to transfer these pre-trained models to new environments with different visual inputs [5]. For example, the performance of the pre-trained model is often reduced or even completely collapsed in a new environment, even if there are only minor differences from the source environments, such as adding irregular shapes or changing colors [6]. Retraining or fine-tuning the model from scratch for a new environment is usually expensive [7]. In many cases, the reward function and network structure designed for the source environment are unsuitable for the new one, and hence the policies obtained by retraining or fine-tuning also perform poorly.

To address this challenge, some studies focus on extracting cross-domain features for training [8], [9], either general features or task-specific features. Some focus on transforming between the source and target domains [6], [10], [11]. These methods require complex network structures and loss functions, as well as large amounts of data; they sometimes require re-completing the entire DRL training process, which is costly and time-consuming. Indeed, these methods do not use the pre-trained model knowledge from the source domain, but rather retrain or fine-tune the model.

In natural language processing (NLP), the prompting method has been incorporated for various tasks and led to

This work was supported by Guangdong Province R&D Program 2020B0909050001, Anhui Province Development and Reform Commission 2020 New Energy Vehicle Industry Innovation Development Project and 2021 New Energy and Intelligent Connected Vehicle Innovation Project, and Shenzhen Yijiahe Technology R&D Co., Ltd. * The corresponding author.

1945-788X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICME55011.2023.00474

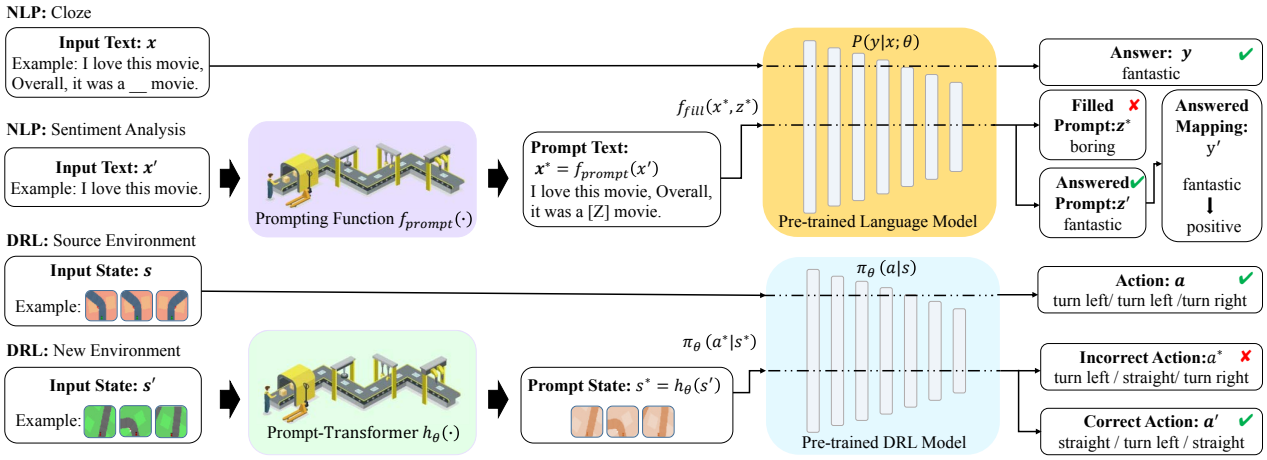

Fig. 1: The correspondence between P^{3}O (lower) and the prompting method in NLP (upper). The prompt-transformer, prompt state, incorrect action, and correct action in our pipeline are mapped to prompt-function, prompt text, filled prompt, and answered prompt in NLP, respectively.

promising results [12]. In general, the prompt function typically adds prompt tokens for adapting pre-trained language models to achieve satisfactory performance in downstream tasks. In this paper, we propose Prompt Based Proximal Policy Optimization (P^{3}O), a three-stage DRL algorithm that uses prompt to transfer visual representations from target to source tasks, leveraging pre-trained models to achieve similar performance in the target environment. The three stages in P^{3}O can be summarized as (1) pre-training, (2) prompting, and (3) predicting, where the correspondence between "prompting" in DRL and that in NLP is shown in Fig. 1.

Specifically, in the prompting process, we use a multi-layer convolutional neural network to build prompt-transformer to fit the prompting function. We expect to use DRL's continuous optimization to learn prompt-transformer's parameters, but many invalid explorations cause slow or failed convergence. Imitation learning with expert knowledge can reduce ineffective exploration and speed up learning. Therefore, we divide the training of prompt-transformer into two steps. In step 1, we use mini data in the target environment to initialize prompt-transformer with imitation learning, so that it guides prompt-transformer to form reasonable initialization weights and reduce invalid explorations in step 2. Then, in step 2, DRL continuously optimizes prompt-transformer by collecting observation data from target environments and trying to obtain output actions for higher rewards. It is worth noting that in both steps, we only update the weights of the prompt-transformer and freeze the other weights pre-trained in the source environment. We conduct the experiments on the CarRacing video game in OpenAI Gym. The results show that P^{3}O can apply pre-trained model knowledge to a target environment, achieving state-of-the-art performance in visual representation transfer.

In summary, our main contributions are as follows:

- We use the prompting method to solve the visual representation transfer problem in DRL, allowing the model pre-trained in the source environment to fully recover its performance in the target environment.
- We propose P^{3}O, a three-stage DRL algorithm that specifies a prompt-transformer for the representation transfer in a prompting method. It enables source knowledge to be applied to target environments with different visual representations.
- Experiments on the CarRacing video game show that P^{3}O outperforms the state-of-the-art method in most environments, with improved training efficiency and algorithm confidence interval.

II. Related Work

A. Deep Reinforcement Learning

A model trained in DRL may perform poorly in a new environment. Yurong et al. [10] use Generative Adversarial Networks (GANs) to transform the simulated domain into the real domain. Gamrian et al. [6] use Unaligned GANs to transform the target environment representation back to the source environment without paired data. Xing et al. [8] use Cycle-Consistent VAE to extract domain-general features in different environments for transfer. Roy et al. [11] minimize the Wasserstein-1 distance of the features between the source and the target to learn a common latent space.

B. Imitation Learning

Imitation learning [13] is used in fields such as joint motion [14], robot manipulation [15], and autonomous driving [16], *etc*. It learns quickly through expert demonstrations but cannot outperform human experts. Collecting expert data is complex, and the model's performance degrades in new environments.

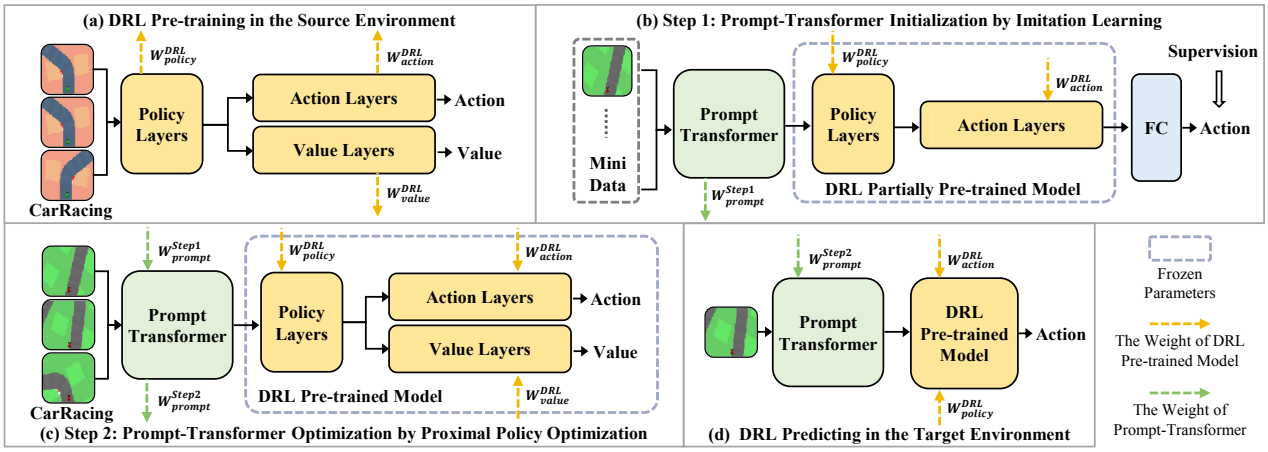

Fig. 2: The overview of P^{3}O. The DRL network is composed of policy, action, and value layers. (a) We train the model in the source environment as the pre-trained model. (b) The imitation learning network in step 1, which is used to initialize the prompt-transformer using mini expert data from target environments. (c) The weights of the prompt-transformer initialized in step 1 are optimized by DRL in target environments in step 2. (d) The optimal prompt-transformer optimized in step 2 is combined with the pre-trained model in (a) to obtain the highest-scoring action output in the target environment.

C. Prompting Method

The development of NLP can be grouped into four paradigms [12]: Fully Supervised Learning (Non-Neural Network), Fully Supervised Learning (Neural Network), "Pre-train, Fine-tune", and "Pre-train, Prompt, Prediction". "Pre-train, Fine-tune" has gained popularity and has produced many pre-trained language models (PLM) [17]–[19], which can be applied to new tasks after fine-tuning, but the increasing size of PLMs requires more hardware and data. The "Pre-train, Prompt, Prediction" paradigm allows tasks to adapt to the PLM without fine-tuning, speeding up learning and reducing training difficulty and parameters.

III. The P^{3}O Design

In this section, we explain the detailed design of P^{3}O. Fig. 2 shows the pipeline of our method, which consists of the following three stages:

(1) DRL pre-training. This stage is to use the PPO algorithm to obtain a high-performance pre-trained model in the source environment.

(2) Prompting. This stage is to optimize the prompt-transformer for representation transfer, which is composed of the following two steps:

(a) Step 1: prompt-transformer initialization by imitation learning. The first step is to use the mini data in the target environments to initialize prompt-transformer with imitation learning.

(b) Step 2: prompt-transformer optimization by proximal policy optimization. The second step is to optimize prompt-transformer in DRL by collecting data from target environments to restore the original performance in the source environment.

(3) DRL predicting. This stage is to use the optimal prompt-transformer to convert the representation from target to source and input it into the pre-trained model, which then predicts the correct actions.

A. DRL Pre-training in the Source Environment

We specify the CarRacing video game as a Markov Decision Process (MDP) problem. The agent is pre-trained in the source environment using the PPO algorithm [1], as shown in Fig. 2a). Specifically, PPO parameterizes a policy $\pi_\theta(a \mid s)$ and a value function $V_\theta(s)$, where a is action and s are the observation data in the source environments. The training objective of the policy network in PPO is to minimize the policy loss:

$$\mathcal{L}^{DRL}_{\text{policy}} = -\hat{\mathbb{E}}\Big[\min\Big(r_\theta \hat{A}(s,a), \text{clip}\left(r_\theta, 1-\epsilon, 1+\epsilon\right)\hat{A}(s,a)\Big]\Big] \tag{1}$$

where $\hat{\mathbb{E}}$ is the empirical expected value and $\hat{A}$ is the expected advantage. $r(\theta)$ is the ratio of the current policy to the previous policy, and ϵ is the hyper-parameter. The clip function is defined as $\text{clip}(\mu, \alpha, \beta) = \max(\min(\mu, \beta), \alpha)$. The training objective of the PPO value network is to minimize the mean squared error, where R is the return value:

$$\mathcal{L}^{DRL}_{\text{value}} = \frac{1}{n}\sum_{i=1}^{n}(V(s) - R)^2 \tag{2}$$

Finally, the policy trained in the source environments serves as the pre-trained model $\pi_\theta(a \mid s)$, with weights W^{DRL}_{policy}, W^{DRL}_{action}, and W^{DRL}_{value} reused without retraining or fine-tuning in step 1 and step 2.

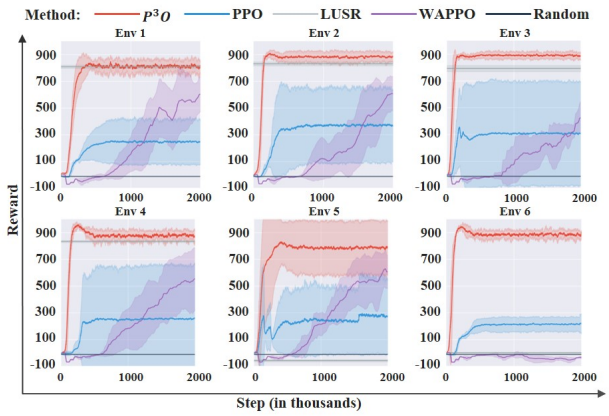

Fig. 3: The comparison of the reward curves for the training by P^{3}O (in red), PPO (in blue), LUSR (in gray), WAPPO (in purple), and random policy (in black) in the same new environments from $Env1 \sim Env6$.

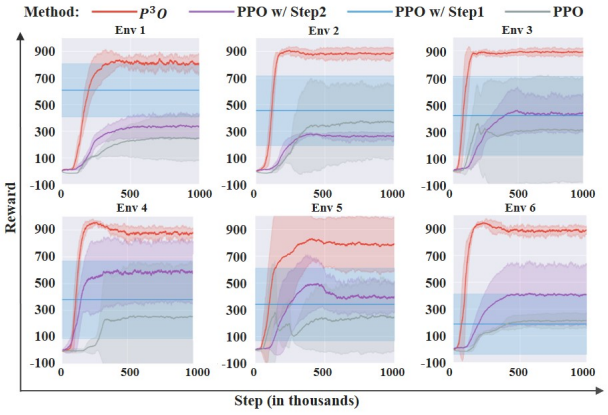

Fig. 4: The comparison of the reward curves for the training by PPO only (in gray), PPO with step 2 (in purple), PPO with step 1 (in blue), and P^{3}O (in red), *i.e.*, PPO with both step 1 and step 2.

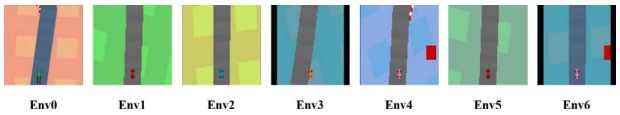

Fig. 5: The source environment $Env0$ and its variant environments $Env1 \sim Env6$ in the OpenAI CarRacing video game.

B. Prompt-Transformer for Representation Conversion in Target Environments

The prompt-transformer attempts to fit the representation transfer function $h_\theta(s')$ and transfer the representation from the target environment to the source one. It consists of four convolutional layers with kernel size of 4x4 and stride of 2 followed by a linear layer. We denote the learned optimal transfer function as $h^*_\theta(s')$. We use expert knowledge of imitation learning to initialize the weights of prompt-transformer in step 1, and optimize them by DRL in step 2 for better performance in the target environments.

C. Step 1: Prompt-Transformer Initialization by Imitation Learning

We use imitation learning to initialize prompt-transformer by relying on expert data in the target environment. This allows prompt-transformer to fit the initial representation transformation function $h_\theta(s')$. We use demonstration data denoted as $\{(s'_1, a'_1), (s'_2, a'_2), \ldots, (s'_n, a'_n)\}$ given by the expert to train the policy $\pi_\theta\left(a' \mid s'\right)$, where s' is the state and a' is the behavior in the target environment. Imitation learning discretizes the action space and uses a classifier (e.g., softmax output and cross-entropy loss). The training loss function is:

$$\mathcal{L}^{Step1}_{\text{imitation}} = -\sum_i a'_i \cdot \ln \pi_\theta\left(\cdot \mid h_\theta(s'_i)\right) \tag{3}$$

Specifically, Fig. 2 b) shows the pipeline of step 1, which includes the prompt-transformer, policy layers, action layers, and fully connected (FC) layers. The network structures of the policy and action layers are fixed and their weights, namely W^{DRL}_{policy} and W^{DRL}_{action}, are from pre-trained models in source environments. These weights are frozen during training. When training in target environments in step 1, the network uses a small amount of expert data to initialize the representation transformation function $h_\theta(s')$ with weights W^{Step1}_{prompt}. This function is used for step 2 to continue optimizing and try to reach the optimal value $h^*_\theta(s')$.

D. Step 2: Prompt-Transformer Optimization by Proximal Policy Optimization

In step 1, we obtain an efficient prompt-transformer initialization $h_\theta(s')$. However, even when the representation of target environments is transformed by $h_\theta(s')$ and input into the pre-trained model, the network still cannot obtain a score close to that in source environments. This indicates that $h_\theta(s')$ is not yet capable of transforming the representation enough for the pre-trained model to recognize them. To improve the performance of the pre-trained model in the target environment, we use the PPO algorithm to further optimize the representation conversion function $h_\theta(s')$. This function converts the representations of the target environment to the source environment, allowing the pre-trained model to achieve optimal performance. The loss function of the policy network is:

$$\begin{aligned}\mathcal{L}^{Step2}_{\text{policy}} = -\hat{\mathbb{E}}\Big[\min\Big(&r_\theta \hat{A}(h_\theta(s'), a'),\\ &\text{clip}\left(r_\theta, 1-\epsilon, 1+\epsilon\right)\hat{A}(h_\theta(s'), a')\Big]\end{aligned} \tag{4}$$

The loss function of the value network is:

$$\mathcal{L}^{Step2}_{\text{value}} = \frac{1}{n}\sum_{i=1}^{n}(V(h_\theta(s')) - R)^2 \tag{5}$$

Fig. 2 c) shows the pipeline of step 2, which includes the prompt-transformer, policy, action, and value layers. Their

TABLE I: The comparison of transfer performance between P^{3}O, PPO, PPO-FT, LUSR, and WAPPO. The ratio in parentheses, which shows the transfer ability, is obtained by dividing the score for environments of the target scenario and the score for the source scenario.

Envs		Average Score (Transfer Ratio) ↑				
		P^{3}O (Ours)	PPO	PPO-FT	LUSR	WAPPO
Sour.	Env0	816.2	432.5	816.2	825.5	684.9
Tar.	Env1	**887.4 (1.09)**	176.8 (0.41)	337.7 (0.41)	808.3 (0.98)	711.3 (1.04)
	Env2	**862.6 (1.07)**	304.3 (0.70)	84.3 (0.10)	830.1 (1.01)	659.8 (0.96)
	Env3	**854.4 (1.05)**	288.8 (0.67)	346.4 (0.42)	791.3 (0.96)	425.7 (0.66)
	Env4	**888.1 (1.09)**	207.1 (0.48)	327.7 (0.46)	833.0 (1.01)	615.1 (0.90)
	Env5	**891.4 (1.09)**	177.8 (0.41)	387.4 (0.47)	-58.5 (-0.07)	579.7 (0.85)
	Env6	**880.8 (1.08)**	128.9 (0.30)	240.6 (0.30)	-0.2 (0.00)	-19.1 (-0.03)

structures and weights are the same as those trained in the source environment. The prompt-transformer (W_{prompt}^{step1}) are pre-trained in step 1. The policy, action and value layers are frozen during training. The representation obtained from the transformation function in the target environment is $\boldsymbol{s}^* = h_\theta(s')$. When the reward reaches its maximum value, the learned weights W_{prompt}^{step2} of prompt-transformer are regarded as the optimal ones.

E. DRL Predicting in the Target Environment

In this stage, we use the optimal prompt-transformer to convert the representation of the target environment which is input into the pre-trained model to predict the action of the target environment. The prediction action of this stage can achieve similar or better scores in the target environment than the pre-trained model in the source environment, as shown in Fig. 2 d).

IV. Experiments

A. Implementation Details

Environments. Our experimental platform is the widely used OpenAI CarRacing video game [20]. Fig. 5 illustrates the environments for seven different scenarios, where $Env0$ is the source environment, and $Env1 \sim Env6$ are the target environments.

DRL Pre-training in the Source Environment. We use PPO to learn a high-performance policy model in source environments of $Env0$ and consider it as the pre-trained model. The input of the PPO network is the observation data of a single frame, *i.e.*, the game screen of the CarRacing environment, and the outputs are the actions, which are sampled from a Gaussian distribution. The learning rate in the pre-training stage is 1.0×10^{-3}. We train the network with batch size = 128, image size = 96×96.

Step 1: Prompt-Transformer Initialization by Imitation Learning. In each target environment, we collect 4 sets of expert data, each containing 1500 pairs of data that complete a full lap of the track. Next, we train the imitation learning network with a learning rate 1.0×10^{-3} using the small amount of expert data collected in each scenario. After training, we save the parameters of prompt-transformer trained in the six environments of step 1 for step 2.

TABLE II: The comparison of training efficiency between P^{3}O, PPO, PPO-FT, LUSR and WAPPO. 'F' denotes that the algorithm fails to converge in the scenario.

Envs	Convergence Step ↓				
	P^{3}O (Ours)	PPO	PPO-FT	LUSR	WAPPO
Env1	**470k**	880k	810k	-	2320k
Env2	**310k**	780k	1290k	-	2420k
Env3	**210k**	660k	1350k	-	2290k
Env4	**440k**	570k	1960k	-	2560k
Env5	**620k**	840k	990k	-	2110k
Env6	**410k**	670k	2000k	-	F

Step 2: Prompt-Transformer Optimization by Proximal Policy Optimization. The prompt-transformer in step 2 uses the weights obtained in step 1, and the weights of policy, action, and value layers are loaded from the pre-trained model. During step 2, the parameters of the policy, the action, and the value layers are also frozen, and only the prompt-transformer is learned. The learning rate used in this step is 2.0×10^{-4}.

B. Comparison of Transfer Performance

We compare the transfer performance of P^{3}O with four baselines: PPO [1], PPO with fine-tune (PPO-FT) [1], the prior state-of-the-art for the model-free policy gradient algorithm for DRL, LUSR [8], the prior state of the art for domain adaptation in DRL, and WAPPO [11], the prior state of the art for visual transfer in DRL. We report the reward for each task. In addition, we also compare PPO with Random Policy (Random). Note that the LUSR and Random methods use the pre-trained model and the random initialization model of the source environment, respectively, so we report the target performance in Fig. 3 as horizontal lines. The light-colored area of the curve in Fig. 3 and Fig. 4 is the confidence interval.

Algorithm Score. Fig. 3 shows the comparison results of the reward for the training by P^{3}O, PPO, LUSR, WAPPO, and Random methods in $Env1 \sim Env6$. The comparison results show that the reward of P^{3}O after convergence in the graph exceeds that of other methods in all environments. Table I summarizes the average score and transfer ratio of PPO, PPO-FT, LUSR, WAPPO and P^{3}O in the target environment. P^{3}O outperforms other algorithms in terms of average score and transfer ratio. The average transfer ratio of P^{3}O is 1.66 times and 1.48 times that of LUSR and WAPPO respectively. P^{3}O's transfer ratios in all target environments are over 1.0, indicating that P^{3}O can leverage the knowledge from the pre-trained model to achieve similar or even higher scores compared to the source environment. This also shows that prompt-transformer plays a consistent role in visual representation transfer across different environments, while other algorithms may lose their effect in some environments.

Algorithm Efficiency. Table II shows the convergence steps of several algorithms. LUSR only uses the pre-trained model, so it is labeled as '-'. Among them, P^{3}O converges in fewer steps and obtains the highest score than other algorithms. Specifically, PPO, PPO-FT, and WAPPO took 1.79, 3.42, and 5.71 times as many steps as P^{3}O to complete convergence, with lower scores. This shows that P^{3}O is more efficient and

TABLE III: Ablation analysis on P^{3}O in $Env1 \sim Env6$.

PPO	Step 1	Step 2	Average Score ↑					
			Env1	Env2	Env3	Env4	Env5	Env6
✓			176.8	304.3	288.8	207.1	177.8	128.9
✓		✓	246.5 ↑	173.2↓	393.8↑	570.0↑	466.5↑	329.3↑
✓	✓		607.5↑	453.9↑	414.1↑	379.5↑	338.9↑	186.7↑
✓	✓	✓	887.4↑	862.6↑	854.4↑	888.1↑	891.4↑	880.8↑

can complete the visual representation transfer task faster. Our P^{3}O (red curve) in Fig. 3 reaches the highest point faster than other methods in all experiments, further proving its efficiency. P^{3}O has higher algorithm efficiency, which enables it to complete the migration of policy at a lower cost and in a shorter time when encountering a new environment.

Algorithm Confidence Interval. P^{3}O has a tighter confidence interval than other algorithms, as shown in Fig. 3, indicating that its results are more consistent and stable. In our experiments, P^{3}O consistently outperforms other algorithms when the environment changed unexpectedly, showing its superior stability and adaptability.

C. Ablation Studies

In order to study the influence of each part of the P^{3}O algorithm on the experiment, we set up four groups of experiments: PPO, PPO w/ step 1, PPO w/ step 2, PPO w/ both steps, *i.e.*, P^{3}O. The experimental results are shown in Fig. 4 and Table III.

The Impact of Step 1. Step 1 uses imitation learning to guide the learning of prompt-transformer effectively. When comparing PPO and PPO with step 1, after introducing step 1, the rewards and average scores in Fig. 4 and Table III have a small range of improvement. It shows that step 1 can provide guidance for the learning of prompt-transformer, and prompt-transformer has the ability to transfer visual representation. When comparing PPO with step 2 and PPO with step 1 and step 2 (P^{3}O), the lack of step 1 guidance will lead to a large decrease in rewards and average scores in Fig. 4 and Table III, which also shows that step 1 provides effective initialization for prompt-transformer is an essential part of the experiment.

The Impact of Step 2. Step 2 is responsible for the continuous optimization of prompt-transformer. After initializing the prompt-transformer in step 1, we show the rewards and average scores in Fig. 4 and Table III to verify the effect of step 2. The comparison of experiments between PPO with step 1 only and PPO with both steps (*i.e.*, P^{3}O) shows that the reward and the average score have significantly improved after introducing step 2.

V. Conclusions

In this work, we propose P^{3}O, a three-stage DRL algorithm that transfers visual representations to quickly adapt to target environments by directly reusing the knowledge of pre-trained models from the source environment. Inspired by the prompting method in NLP, we design a representation transfer module prompt-transformer and use the three-stage DRL algorithm to optimize prompt-transformer so as to realize the representation transfer task from the target environment to the source environment. Experiments on the OpenAI CarRacing video game show that P^{3}O reaches the state-of-the-art in visual representation transfer. In future work, we will extend our algorithm to the cross-environment and cross-modal tasks for the high-performance DRL method driven by prompting.

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017, unpublished.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013, unpublished.

[3] G. Chen, S. Yao, J. Ma, L. Pan, Y. Chen, P. Xu, and et al., "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, p. 4836, 2020, in press.

[4] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, pp. 1143–1150, 2020, in press.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IROS*, 2017, pp. 23–30, in press.

[6] S. Gamrian and Y. Goldberg, "Transfer learning for related reinforcement learning tasks via image-to-image translation," in *International conference on machine learning*, vol. 97, 2019, pp. 2063–2072, in press.

[7] A. Fickinger, H. Hu, B. Amos, S. Russell, and N. Brown, "Scalable online planning via reinforcement learning fine-tuning," *NIPS*, vol. 34, pp. 16 951–16 963, 2021, in press.

[8] J. Xing, T. Nagata, K. Chen, X. Zou, E. Neftci, and J. L. Krichmar, "Domain adaptation in reinforcement learning via latent unified state representation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 452–10 459, in press.

[9] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *International Conference on Learning Representations, Austria*, 2021, in press.

[10] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *BMVC*, 2017, in press.

[11] J. Roy and G. D. Konidaris, "Visual transfer for reinforcement learning via wasserstein domain confusion," in *Proceedings of AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 9454–9462, in press.

[12] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, pp. 1–35, 2023, in press.

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," *NIPS*, vol. 29, pp. 4565–4573, 2016, in press.

[14] N. D. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *IEEE-RAS International Conference on Humanoid Robots, Pittsburgh*, 2007, pp. 392–397, in press.

[15] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," in *IEEE international conference on robotics and automation*, 2021, pp. 4613–4619, in press.

[16] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, and et al., "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016, unpublished.

[17] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of naacL-HLT*, 2019, pp. 4171–4186, in press.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, and et al., "Language models are few-shot learners," *NIPS*, vol. 33, pp. 1877–1901, 2020, in press.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, and et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, pp. 5485–5551, 2020, in press.

[20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016, unpublished.