Robust Domain Randomised Reinforcement Learning through Peer-to-Peer Distillation

Robust Domain Randomised Reinforcement Learning through Peer-to-Peer Distillation

Chenyang Zhao
School of Informatics
University of Edinburgh
c.zhao@ed.ac.uk

Timothy M. Hospedales
School of Informatics
University of Edinburgh
t.hospedales@ed.ac.uk

Abstract

In reinforcement learning, domain randomisation is an increasingly popular technique for learning more general policies that are robust to domain-shifts at deployment. However, naively aggregating information from randomised domains may lead to high variance in gradient estimation and unstable learning process. To address this issue, we present a peer-to-peer online distillation strategy for RL termed P2PDRL, where multiple workers are each assigned to a different environment, and exchange knowledge through mutual regularisation based on Kullback–Leibler divergence. Our experiments on continuous control tasks show that P2PDRL enables robust learning across a wider randomisation distribution than baselines, and more robust generalisation to new environments at testing.

1 Introduction

Deep reinforcement learning (RL) has been successfully applied to various tasks, including Go [22], Atari [13] and robot control tasks [21], etc. However, a growing body of research has shown that it remains challenging to learn deep RL agents that are able to generalise to new environments [5]. Agents can 'overfit' to the training environments visual appearance [25, 5] physical dynamics [14, 29] or even specific training seeds [27]. If there is a new environment, or domain-shift, between training and testing, RL tend to under-perform significantly. This problem has longer been appreciated, and is particularly salient, in the field of robotics, where there is an inevitable *reality gap* [25, 11] between the simulated environments used for training and deployment in the real-world. These issues have motivated an important line of research in improving zero-shot domain transfer [5, 1], i.e., to learn generalised policies from training domains that can be directly and successfully deployed in unseen testing domains without further learning or adaptation.

One common strategy to improve policy generalisation is to apply *domain randomisation* during training. With domain randomisation, we first generate a random set of, or distribution over, domains with different properties such as observation function [25], dynamics [16], or both [1]. Then a policy is trained with this distribution, for example by drawing a new random sample from the domain distribution during each learning episode. If this distribution of training domains is rich enough to include within its support properties of the real-world or target environment – and if a policy can be successfully trained to fit the entire training domain distribution – then that policy can be successfully deployed in the target environment without adaptation. A challenge with domain randomisation is that training a single model on a wide distribution of domains usually leads to high variance gradient estimates, making the RL policy optimisation problem challenging [12]. To ameliorate this issue, several studies have proposed distillation techniques where an ensemble of workers are each trained locally in a different domain – where gradients are lower variance, and hence RL is more stable. The local workers are then distilled into a single global policy that should perform across all domains

(NeurIPS 2020), Vancouver, Canada.Workshop of Deep Reinforcement Learning at 34th Conference on Neural Information Processing Systems

[7, 24, 18, 15, 6]. Distilling local policies into a global policy with a supervised objective provides a more stable way to learn from a broad distribution over domains and ultimately enable better generalisation to held-out testing domains – albeit at some additional computational cost [7].

This work continues this line of investigation. Our main contribution is to show that this centralised distillation is unnecessarily, and indeed sub-optimal. In particular, we propose an online distillation framework, where each worker both learns to optimise performance in a local domain and also mimics its peers from other domains with peer-to-peer distillation. More specifically, inspired by deep mutual learning [28, 2], we train each worker with two losses: a conventional RL loss and a distillation loss that measures the similarity in predictions between the local worker and the others. In this work, we use expected Kullback–Leibler (KL) divergence between predicted distributions as the objective to optimise for information exchange. This online distillation framework is named *Peer-to-Peer Distillation Reinforcement Learning* (P2PDRL). We empirically show that, compared to conventional baselines, and offline distillation alternatives, e.g. divide-and-conquer (DnC) reinforcement learning [7], P2PDRL achieves more competitive learning performance without the additional cost of a centralised distillation step. Overall P2PDRL learns more quickly, succeeds to learn on a wider distribution of domains, and transfers better to novel testing environments compared to competitors.

The key contributions of this work are summarised as:

- We propose P2PDRL, an online distillation framework for deep RL agents that uses an ensemble of local workers that learn with local experience only, while sharing knowledge via regularisation on the statistical distance of their prediction distribution;
- Our experiments show that P2PDRL can achieve more effective learning on a wider distribution of training environments compared to competitors, without distilling to a centralised policy. All workers show increased robustness to the domain shift of novel environments.

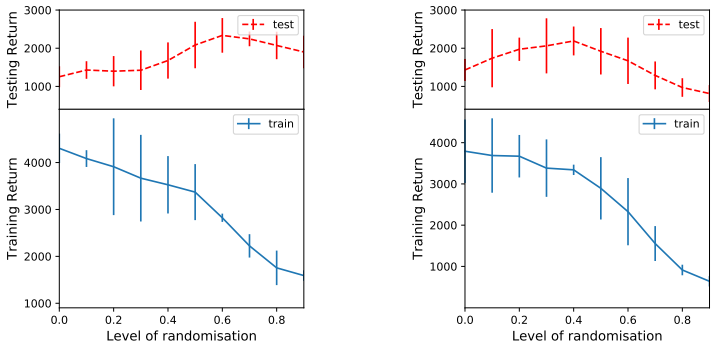
2 Related Work

Domain Generalisation in Deep RL A growing body of work has discussed generalisation problems in reinforcement learning. In the context of tasks with discrete and continuous observation/action spaces, [5] and [27] concluded that deep RL policies could easily overfit to random seeds used during training and underperform during testing. In the context of domain-shift between training and testing, [29] and [14] further highlighted the problem that deep RL agents often fail to generalise, e.g., across different friction coefficients or wind conditions.

To improve RL agents' performance in testing environments, several *adaptation* methods have been proposed [8, 19]. These use data from testing domains and aim to perform sample efficient adaptation. Recent advances [4, 17] perform meta-learning on training environments to learn how to adapt efficiently to testing environments. Where direct deployment to target domains without adaptation is desired or necessary, a common approach is *domain randomisation*. Here the training simulator is setup to provide a diverse array of environments for learning in terms of observations [20, 25, 1] or dynamic perturbations [23]. A key challenge here is that, in absence of a known and precisely-specified model of the testing domain, one must define a rather wide distribution of training domains to be confident that its span includes likely testing domains. However, learning from such a diverse training signal is extremely challenging in RL, and often leads to unstable learning and poor convergence [12, 26], as we also illustrate here in Fig. 1.

In this paper we address this dichotomy between overfitting of agents to individual domains, and inability of agents to fit to a wide distribution of domains. Our framework enables state of the art on-policy methods [21] to effectively and stably learn a wide distribution of domains by performing local RL and global knowledge exchange via distillation.

Policy Distillation The notion of model distillation [10] has increasingly been applied to policy distillation in reinforcement learning. It is often used for learning from multiple source tasks, where a student agent is trained to mimic the behaviour of one or multiple teacher policies [6]. Several early studies [18, 15] assumed pre-trained teacher policies and trained one student to match the state-dependent probability of the teacher's predicted actions. Later methods [24] learned a global policy in parallel with learning multiple local policies: the global policy is trained to distil from local individuals while local policies learning is regularised by the global policy. Finally, [7] proposed to



(a) Walker task (b) Ant task

Figure 1: Comparisons between asymptotic training performance and testing performance in a hidden testing domain, both as a function of the diversity of training distributions. Policies are trained with PPO. Results are averaged over 8 different random seeds.

alternate between training local agents regularised by a global policy, distilling a central global policy, and periodically resetting local agents to the distilled global policy.

Compared to these methods, we propose a online peer-to-peer distillation framework that avoids explicitly distilling to a global policy. Instead, we learn a cohort of workers collaboratively by performing local RL and global peer-to-peer knowledge distillation. Peers use their own local states for distillation, thus significantly simplifying data exchange between workers compared to alternatives such as [7]. This idea of online distillation has also been explored in supervised learning to achieve learning better performance in single-task learning [28] or to scale single-task supervised learning to large datasets through distributed computing [2]. To our knowledge, it has not been explored in RL, or as a tool to enable learning on a wider distribution of training domains, and hence improved robustness and cross-domain generalisation.

3 Methodology

3.1 Preliminaries

An episodic Markov decision process (MDP) is defined by $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, and γ is the discount factor. The objective of a learning agent is to learn a stochastic policy $\pi^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that maximises the expected cumulative return: $\pi^* = \arg\max_\pi \mathbb{E}_\pi \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$.

We denote the parameters that describe a domain as ξ. In domain randomisation, each training domain is randomly sampled with domain parameters ξ from a pre-defined set Ξ. Thus, the modified objective function becomes

$$\pi_* = \arg\max_\pi \mathbb{E}_{\xi \sim \Xi} \mathbb{E}_\pi \sum_{t=0}^{\infty} \gamma^t R(s_{t,\xi}, a_{t,\xi}). \tag{1}$$

Policies trained with a diverse set of domains should generalise better to unseen testing domains [1]. However, in practice, diverse training domains lead to high variance in gradients [12], thus leading to poor learning behaviour. In practice, this manifests as a significant drop in training performance as diversity of training domains increases. In such cases, the effect of the policy's inability to fit the training task at all dominates its greater exposure to diverse environments due to domain randomisation. Thus performance in both training and testing domains is poor.

This challenge is illustrated in Fig. 1, which shows the performance of an agent during training as well as its performance when tested on a held-out domain – all plotted as a function of training set diversity. As the diversity of training domains increases from zero, we see that training performance decreases continually as it becomes harder for a single policy to fit an increasingly diverse distribution of environments (Fig. 1, below). More interestingly, we also observe that testing performance initially increases, as the increased domain randomisation benefits robustness to the held out domain (Fig. 1,

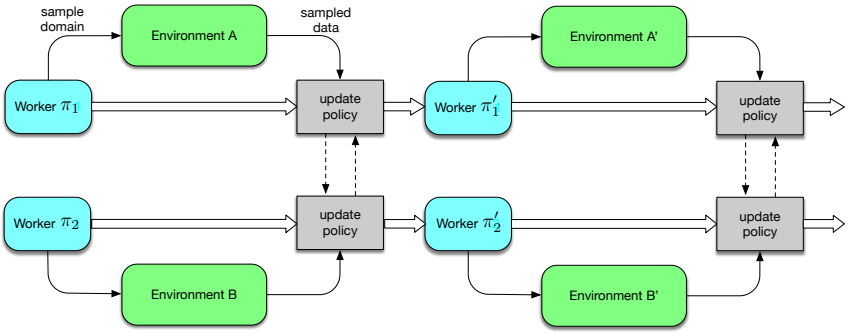


Figure 2: Schematic of P2PDRL with two workers. At each iteration, a worker is randomly assigned to a domain, and updates its policy by local RL, while being regularised by its peers learning in other domains. Dotted lines indicate matching prediction distribution by KL divergence based distillation.

above). However as training diversity continues to increase, testing performance starts to drop, as the stronger domain randomisation challenge makes it difficult for the policy to learn the task at all.

3.2 Online Distillation

To avoid training with data sampled from all domains, we propose to assign each worker to a domain, and train it with data from its local domain only. Alongside the conventional RL loss, workers are regularised by each other through an online distillation loss, which encourages all workers to act similarly across all domains given the same state input.

Local Optimisation We formulate the proposed peer-to-peer distillation method with a cohort of K workers, as illustrated in Fig. 2. At each iteration, each worker randomly samples a domain and generates trajectory data by following its local policy. The objective function includes a conventional RL loss and a KL divergence based distillation loss that matches predicted distributions across all workers. In particular, we use proximal policy optimisation (PPO) [21], a state of the art on-policy deep reinforcement learning algorithm, as the base optimiser. A surrogate loss is used in PPO:

$$\mathcal{L}_{\text{PPO}}(\theta;\xi) = \mathbb{E}_{s_t,a_t}\Big[\min\big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}A_{t,\xi}, \text{clip}(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)A_{t,\xi}\big)\Big], \tag{2}$$

where ϵ is the hyperparameter, and $A_{t,\xi}$ is the estimated advantage in domain ξ at timestep t. The critic networks, parameterised by ϕ_i, are trained locally with conventional MSE loss, where target values are computed through Bellman equations using local trajectory data.

Peer-to-Peer Distillation For online distillation across different workers, we minimise KL divergence between workers' predictions. More specifically, for the i-th worker,

$$\mathcal{L}^i_{\text{dis}}(\theta) = \frac{1}{K-1}\sum_{k}^{k\neq i}\mathbb{E}_{\pi_i,\xi_i}\big[D_{\text{KL}}(\pi_{\theta_i}(\cdot|s)||\pi_{\theta_k}(\cdot|s))\big], \tag{3}$$

where $\mathbb{E}_{\pi_i,\xi_i}(\cdot)$ is the expectation over trajectory data generated in domain ξ_i with policy π_i.

Discussion Our overall algorithm is summarised in Fig. 2 and Alg. 1. Note that information is only exchanged in distillation steps between workers. Different from DnC [7], we do not require local workers to have access to trajectory data sampled by peers; only model parameters need to be shared between workers. This simplifies data exchange and enables more efficient implementation. Furthermore, we do not distill local policies to a central global policy. Although each worker optimises locally, over time they *all* become domain invariant due to peer-to-peer knowledge exchange.

4 Experiments and Results

4.1 Experiment Setup

We focus our analysis on five continuous control tasks from OpenAI Gym, including *Walker, Ant, Humanoid, Hopper, HalfCheetah* [3]. All tasks are simulated in MuJoCo. Illustrative figures of tasks

Algorithm 1 Online Peer-to-Peer Distillation Reinforcement Learning with PPO

Input: Distribution over domains Ξ, number of workers K, hyperparameter α, max timesteps T.
Initialise: Initial actor θ_0, initial critic ϕ_0.
Initialise: every worker with θ_0, ϕ_0
while not converge **do**
  for $i = 1, 2, ..., K$ **do**
    Sample a domain $\xi_i \sim \Xi$ for worker i
    Collect trajectory data τ_i following policy π_i for T timesteps in domain ξ_i.
  end for
  for $1, 2, ...,$ max epoch number **do**
    for $1, 2, ...,$ num of minibatches per epoch **do**
      for $i = 1$ **to** K **do**
        Sample minibatch $\mathcal{B}_i$ from τ_i
        $\theta_i \leftarrow \theta_i - \nabla_{\theta_i}[\mathcal{L}_{\text{PPO}}(\mathcal{B}_i, \theta_i) + \alpha\mathcal{L}^i_{\text{dis}}(\mathcal{B}_i, \theta_i)]$
        Compute target value V^{targ} with Bellman equation.
        Update ϕ_i with MSE loss $(V_n - V^{\text{targ}})^2$.
      end for
    end for
  end for
end while

are included in Fig. 3. To generate different domains, we change wind condition, gravity constant, friction coefficient, robot mass and initial state distribution in simulation. We summarise the diversity of randomised distribution as a scalar $\epsilon \in [0, 1]$. Higher ϵ means more diverse distributions. Given a specific ϵ, dynamic parameters are randomly sampled from a uniform distribution defined by ϵ. For example, robot mass in *Walker* task m is sampled from $m \sim \mathcal{U}(m_0(1 - 0.5\epsilon), m_0(1 + 0.5\epsilon))$. Detailed task descriptions are listed in the Appendix.

Our experiments are designed to address the following questions:

Q1: How does P2PDRL compare to baselines in sample efficiency and asymptotic return?

Q2: Does P2PDRL learn policies that are more robust against domain shifts than competing methods?

We compare our P2PDRL, with the following prior methods as baselines:

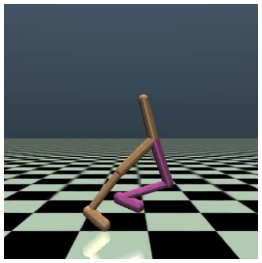
PPO [21]. PPO provides the state of the art on-policy Deep RL method, which is also used as the basic learning algorithms across all settings. At each iteration, agents are optimised with the union of datasets from all sampled domains.

Distributed PPO [9]. In contrast to PPO, where one agent is optimised by the rich data set, in this setting, workers gather gradient information with local data and a central learner updates a global policy with the average of gradients across all domains.

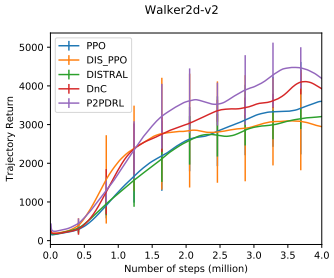
Distral [24]. Originally focused on knowledge transferring when learning multiple tasks, Distral trains an ensemble of task-specific policies, constrained against a single global policy at each gradient step. The global policy is trained with supervised learning, to distil the common behaviours from task-specific policies.

DnC [7]. Originally focused on providing robustness to choice of initial state in single domain learning, DnC partitions the initial state distribution into multiple sub-domains, and alternates between local policy optimisation steps and global distillation steps. In this work, we go beyond initial states, and train agents in the domains with various sets of dynamics parameters. Moreover, we do not require domain knowledge to partition the distribution. Instead, domains are sampled from the same distribution for all workers.

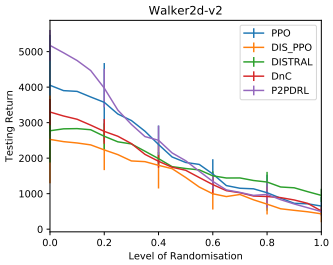
Implementation Details To demonstrate the performance of our proposed method, we take $K = 2$ workers in parallel, regularised by each other. Except where noted otherwise, we control the *total* number of timesteps to compare sample efficiency fairly: At each iteration, the vanilla PPO samples 4096 steps; and each worker in other algorithms samples 2048 steps. In DnC and P2PDRL, half the number of samples are used by each actor to train locally. For the implementation of actor and critic networks, we use two separate MLP networks with two hidden layers, 64 units in each layer. For



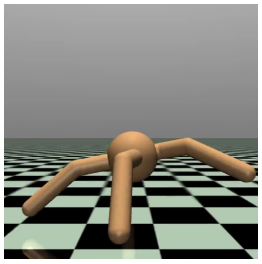
(a) Walker task



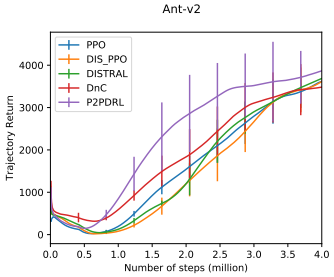
(b) Training return on Walker



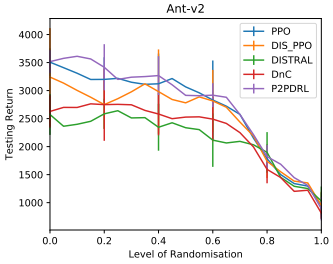
(c) Testing return on Walker



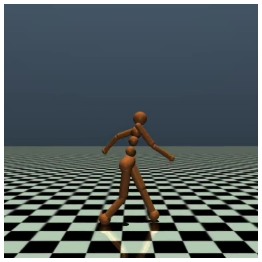
(d) Ant task



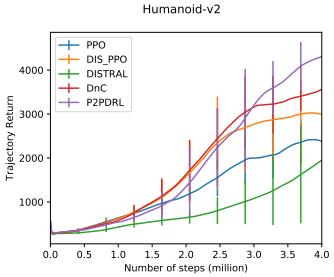
(e) Training return on Ant



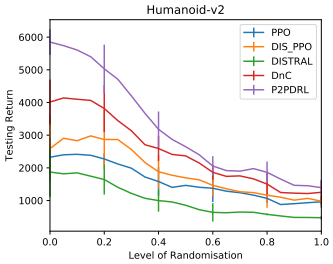
(f) Testing return on Ant



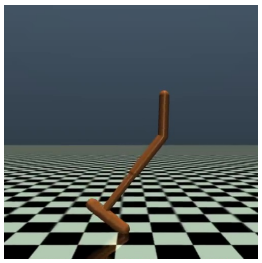
(g) Humanoid task



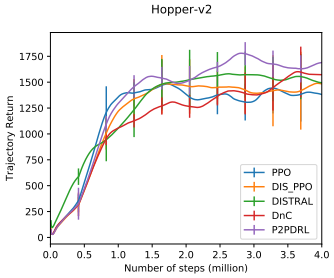
(h) Training return on Humanoid



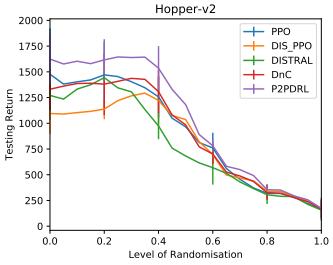
(i) Testing return on Humanoid



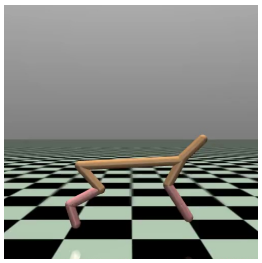
(j) Hopper task



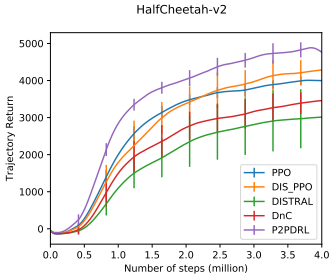
(k) Training return on Hopper



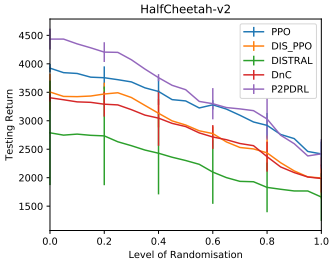
(l) Testing return on Hopper



(m) HalfCheetah task

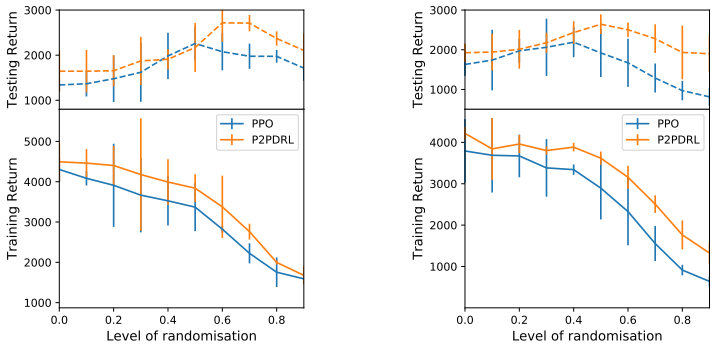


(n) Training return on HalfCheetah



(o) Testing return on HalfCheetah

Figure 3: Summary of training and generalisation performances across five different continuous control tasks. The leftmost column illustrates the task. The middle column compares the training return, represented by learning curves. The rightmost column shows the testing performance as a function of the testing randomised distribution diversity ϵ^{te}. All experiments are trained with a predefined training randomisation distribution $\epsilon^{tr} = 0.2$. Results are averaged over 8 random seeds.



(a) Walker task (b) Ant task

Figure 4: Comparisons between asymptotic training and testing performance, both as a function of the diversity of the training randomisation distribution. Policies are trained with different randomisation strengths ϵ^{tr} and tested with a pre-defined held-out testing domain.

each task, we run a hyperparameter sweep for each method, and report the best performing agents. All results are averaged over 8 random seeds. Full implementation details are listed in the Appendix.

4.2 Training and Generalisation Performance

Training Efficiency Under Domain Randomisation

As training distribution, we take $\epsilon^{tr} = 0.2$ for all five tasks. From the learning curves shown in Fig. 3(middle), we can see that P2PDRL generally outperforms the baseline methods in terms of asymptotic performance across all 5 tasks, and is able to learn more sample-efficiently in some tasks, such as *Ant* and *HalfCheetah*.

Generalisation to Novel Environments We next evaluate the generalisation performance of the policy learned above, by evaluating it on novel testing domain with different randomisation strengths ϵ^{te}. Fig. 3(right) shows that P2PDRL learned policies are able generally to generalise better against domain shifts to novel environments.

Dependence of Testing Performance on Randomisation Strength in Training We also demonstrate asymptotic training return of a policy and its generalisation performance as a function of range of training domains, as initially motivated in Fig. 1. We train our policy with different levels of randomisation ϵ^{tr} and test all learned policies in a pre-defined target domain where $\epsilon^{te} = 0.5$. As shown in Fig. 4, P2PDRL generally outperforms vanilla PPO, but their performance levels decrease in tandem as the training distribution becomes more diverse and harder to fit with a single model. Crucially, as discussed earlier, the generalisation performance of PPO in terms of testing return drops rapidly after around $\epsilon^{tr} = 0.5$. Meanwhile the testing return of P2PDRL is stable up to a much higher level of training domain randomisation.

5 Conclusion

Generalisation of trained agents to novel domains is an crucial but lacking capability in today's RL. We propose an online distillation based reinforcement learning algorithm P2PDRL, consisting of domain-local RL steps and global knowledge exchange through peer-to-peer distillation. We empirically show that P2PDRL provides improved generalisation to novel domains via stable training on a wide range of randomised domains.

References

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. In *ICLR*, 2018.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019.

[6] Wojciech Marian Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *AISTATS*, 2019.

[7] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *ICLR*, 2018.

[8] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *ICLR*, 2017.

[9] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS deep learning workshop*, 2014.

[11] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145, 2012.

[12] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *CoRL*, 2019.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. URL `http://dx.doi.org/10.1038/nature14236`.

[14] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[15] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

[16] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[17] Samuel Ritter, Jane X Wang, Zeb Kurth-Nelson, Siddhant M Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been there, done that: Meta-learning with episodic recall. In *ICML*, 2018.

[18] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

[19] Andrei A Rusu, Mel Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *CoRL*, 2017.

[20] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. In *Robotics: Science and Systems*, 2017.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[22] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[23] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *CoRR*, 2018.

[24] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *NeurIPS*, 2017.

[25] 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)*, 2017.

[26] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

[27] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[28] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

[29] Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M Hospedales. Investigating generalisation in continuous deep reinforcement learning. *arXiv preprint arXiv:1902.07015*, 2019.

Task Description

For our experiments, we modify five continuous control tasks from OpenAI gym, including *Walker2d*, *Ant*, *Humanoid*, *Hopper* and *HalfCheetah*. We use a scalar $\epsilon \in [0, 1]$ to control the diversity of randomised domains. The pre-defined distributions are listed in Table. 1.

Table 1: List of randomised distributions

Wind condition	$w \sim \mathcal{U}(-5.0\epsilon, 5.0\epsilon)$
Gravity constant	$g \sim \mathcal{U}\big(g_0(1-0.25\epsilon), g_0(1+0.25\epsilon)\big)$
Friction coefficient	$f \sim \mathcal{U}\big(f_0(1-0.3\epsilon), f_0(1+0.3\epsilon)\big)$
Robot mass	$m \sim \mathcal{U}\big(m_0(1-0.5\epsilon), m_0(1+0.5\epsilon)\big)$
Initial position	$x \sim \mathcal{U}(x_0-\epsilon, x_0+\epsilon)$

Hyperparameters

The detailed hyperparameters are listed below:

- Network: 2 separate MLP networks for actor and critic, each with 2 hidden layer and 64 hidden units each layer, tanh nonlinearities.
- Number of epochs each iteration: 10
- Minibatch size: 64
- Clipping parameter: 0.2
- Discount factor: 0.99
- GAE parameter: 0.95
- Optimiser: Adam optimiser

For PPO and Distributed PPO, we run a hyperparameter sweep on learning rate $\beta : [1\mathrm{e}-4,\ 3\mathrm{e}-4,\ 1\mathrm{e}-3,\ 3\mathrm{e}-3,\ 1\mathrm{e}-2]$. For Distral, DnC and P2PDRL, we run a hyperparameter sweep on learning rate $\beta : [1\mathrm{e}{-4}, 3\mathrm{e}{-4}, 1\mathrm{e}{-3}, 3\mathrm{e}{-3}, 1\mathrm{e}{-2}]$ and distillation loss coefficient $\alpha : [0.1, 0.3, 1.0, 3.0, 10.0]$. We report the best performing set of hyperparameters.