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

Chenyang Zhao CZHAO2@ED.AC.UK
University of Edinburgh, United Kingdom

Timothy Hospedales T.HOSPEDALES@ED.AC.UK
University of Edinburgh, United Kingdom

Abstract

In reinforcement learning, domain randomisation is a popular technique for learning general policies that are robust to new environments and domain-shifts at deployment. However, naively aggregating information from randomised domains may lead to high variance in gradient estimation and sub-optimal policies. To address this issue, we present a peer-to-peer online distillation strategy for reinforcement learning termed P2PDRL, where multiple learning agents are each assigned to a different environment, and then 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 performance to new environments at testing.

Keywords: domain randomisation, deep reinforcement learning, mutual learning

1. Introduction

Deep reinforcement learning (RL) has been successfully applied to various tasks, including Go (Silver et al., 2016), Atari (Mnih et al., 2015), and robot control tasks (Schulman et al., 2017), 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 (Cobbe et al., 2019). Agents can 'overfit' to the training environments visual appearance (Tobin et al., 2017; Cobbe et al., 2019), physical dynamics (Packer et al., 2018; Zhao et al., 2019), or even specific training seeds (Zhang et al., 2018a). If there is a new environment, or domain-shift, between training and testing, RL tends 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* (Tobin et al., 2017; Koos et al., 2012) 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 (Cobbe et al., 2019; Andrychowicz et al., 2020), 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 (Tobin et al., 2017), dynamics (Peng et al., 2018), or both (Andrychowicz et al., 2020). A policy is then 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, which will subsequently raise the RL policy optimisation problem (Mehta et al., 2020). To ameliorate this issue, several studies have proposed distillation techniques where an ensemble of teacher models are each trained locally in a different domain – where gradients are lower variance, and hence RL is more stable. The teacher models are then distilled into a global student policy that should perform across all domains (Ghosh et al., 2018; Teh et al., 2017; Rusu et al., 2016; Parisotto et al., 2016; Czarnecki et al., 2019). Distilling local policies into a global policy with a supervised objective provides a more robust 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 (Ghosh et al., 2018). Figure 1 summarises the learning flowcharts of different policy distillation methods.

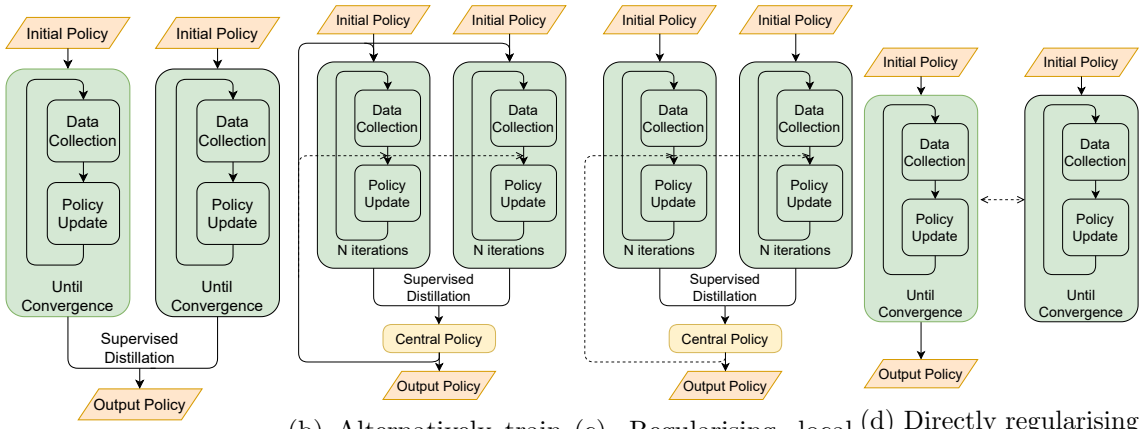
Our work continues this line of investigation. The main contribution is to show that the centralised distillation is unnecessarily, and indeed sub-optimal. In particular, we propose an online distillation framework, where each agent both learns to optimise performance in a local domain and mimics its peers from other domains with peer-to-peer distillation, as shown in Fig. 1d. More specifically, inspired by deep mutual learning (Zhang et al., 2018b; Anil et al., 2018), we train each model with two losses: a conventional RL loss, and a distillation loss that measures the similarity in predictions between the local model and the others. In this work, the expected Kullback–Leibler (KL) divergence between predicted distributions as the objective is used 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 (Ghosh et al., 2018), 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 that uses an ensemble of local agents that learn with local experience only, while sharing knowledge via regularisation on the statistical distance of their predicted 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 peers 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 RL. In the context of tasks with discrete and continuous observation/action



(a) Distilling from pre-trained teachers models

(b) Alternatively training local teachers and central models

(c) Regularising local training with the central policy

(d) Directly regularising local training with peers

Figure 1: Flowcharts of different policy distillation methods. The dashed line indicates training with regularised losses. (a) pre-trained teacher models are distilled into a student policy (Parisotto et al., 2016; Rusu et al., 2016); (b) after every N iterations of locally training, the teacher models are distilled into and then re-initialised as the central policy (Ghosh et al., 2018); (c) a central policy is trained to mimic teacher models and local model training is regularised by KL-divergence to the central policy (Teh et al., 2017); (d) P2PDRL directly regularise local training with KL-divergence to every other peers.

spaces, Cobbe et al. (2019) and Zhang et al. (2018a) 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, Zhao et al. (2019) and Packer et al. (2018) 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 (Gupta et al., 2017; Rusu et al., 2017). These use data from testing domains and aim to perform sample efficient adaptation. Recent advances (Clavera et al., 2019; Ritter et al., 2018) 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 (Sadeghi and Levine, 2017; Tobin et al., 2017; Andrychowicz et al., 2020) or dynamic perturbations (Tan et al., 2018). A key challenge 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 high variance in gradient estimation and convergence to sub-optimal policies(Mehta et al., 2020; Yu et al., 2020), as we illustrate in Fig. 2.

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 (Schulman et al., 2017) to effectively and robustly

learn a wide distribution of domains by performing local RL and global knowledge exchange via distillation.

Policy Distillation The notion of model distillation (Hinton et al., 2014) has increasingly been applied to policy distillation in RL. 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 (Czarnecki et al., 2019). Several early studies (Rusu et al., 2016; Parisotto et al., 2016) assumed pre-trained teacher policies and trained one student to match the state-dependent probability of the teacher's predicted actions (Fig. 1a). Later methods (Teh et al., 2017) 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 (Fig. 1c). Finally, Ghosh et al. (2018) proposed to 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 (Fig. 1b).

Compared to these methods, we propose an online peer-to-peer distillation framework that avoids explicitly distilling to a global policy (Fig. 1d). Instead, we learn a cohort of agents collaboratively by performing local RL and global peer-to-peer knowledge distillation. Peers use their own local states for distillation, thus simplifying data exchange compared to alternatives such as Ghosh et al. (2018).

Mutual Learning While distillation (Hinton et al., 2014) is widely used in teacher-student training, a minority of studies (Zhang et al., 2018b; Anil et al., 2018) have explored simultaneous mutual learning among a cohort of students. These have considered improving conventional generalisation in supervised learning. We provide the first investigation of peer-to-peer learning for RL, and for the purpose of improving robustness to domain-shift.

3. Methodology

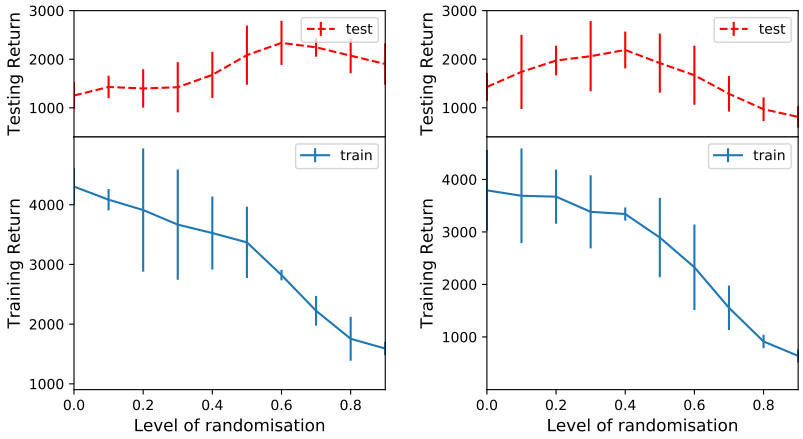
3.1. Preliminaries

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 (Andrychowicz et al., 2020). However, in practice, diverse training domains may cause high variance in gradients (Mehta et al., 2020), 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.



(a) Walker task (b) Ant task

Figure 2: Comparisons between asymptotic training performance and testing performance in a hidden testing domain, both as a function of training distribution diversity. Policies are trained with PPO. Results are averaged over 8 random seeds.

This challenge is illustrated in Fig. 2, 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. 2, below). More interestingly, we also observe that testing performance initially increases, as the increased domain randomisation benefits robustness to the held out domain (Fig. 2, 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 learning agent to a domain, and train it with data from its local domain only. Alongside the conventional RL loss, agents are regularised by each other through an online distillation loss, which encourages all agents to act similarly across all domains given the same state input.

We formulate the proposed peer-to-peer distillation method with a cohort of K local agents. Each agent consists of an actor network, parameterised by θ_i, and a critic network, parameterised by ϕ_i.

Fig. 1d illustrates the process of P2PDRL in the case of two agents. At each iteration, each agent randomly samples a domain ξ_i and generates trajectory data by following its local policy. The objective function includes a conventional RL loss, which learns from the local experience, and a KL divergence-based distillation loss that matches predicted distributions across all local agents.

Local Optimisation To optimise over the local experience, we use proximal policy optimisation (PPO) (Schulman et al., 2017), an industry standard on-policy deep RL

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

Input: Distribution over domains Ξ, num. of learning agents K, hyperparameter α, max timesteps T.
Initialise: Initial actor θ_0, initial critic ϕ_0.
Initialise: every agent with θ_0, ϕ_0
while not converge **do**
  for $i = 1, 2, ..., K$ **do**
    Sample a domain $\xi_i \sim \Xi$ for agent 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

algorithm, as the base optimiser. The following surrogate loss is optimised in PPO to update actor θ_i:

$$\mathcal{L}_{\text{PPO}}(\theta_i; \xi_i) = \mathbb{E}\Big[\min\big(r(t)A_{t,\xi_i}, \text{clip}(r(t), 1-\epsilon, 1+\epsilon)A_{t,\xi}\big)\Big], \tag{2}$$

where ϵ is the hyperparameter, $A_{t,\xi}$ is the estimated advantage in domain ξ at timestep t, and $r(t)$ is the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The critic networks 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 Similar to Zhang et al. (2018b), we minimise KL divergence between models' predictions for online distillation across different agents. More specifically, the KL mimicry distillation loss for the i-th agent is,

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

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

During training, the peers are optimised jointly and collaboratively, with the optimisation processes for the all peers being closely coupled. The overall loss function for each peer is

$$\mathcal{L}_{\theta_i} = \mathcal{L}_{\text{PPO}}(\theta_i, \xi_i) + \alpha\mathcal{L}_{\text{dis}}(\theta_i, \xi_i). \tag{4}$$

In this way, each network learns both to optimise the expected return given local experience as well as to match the probability estimate of its peer with the KL mimicry loss.

Discussion Our overall algorithm is summarised in Fig. 1d and Alg. 1. Note that information is only exchanged in distillation steps between learning agents. Different from DnC (Ghosh et al., 2018), we do not require local learners to have access to trajectory data sampled by peers; only model parameters need to be shared. This simplifies data exchange and enables more efficient implementation. Furthermore, we do not distill local policies to a central global policy. Although each agent optimises locally, over time, they *all* become domain invariant due to peer-to-peer knowledge exchange.

4. Experiments and Results

4.1. Summary

Our experiments investigate and answer several questions summarized here, with full details given in the following sections.
Q1: How does P2PDRL compare to other baselines in terms of training performance, including both sample efficiency and asymptotic return? **A1:** Training performance is similar or asymptotically slightly better than competitors.
Q2: Can P2PDRL learn more robust policies that perform better against domains shifts during testing, compared to competing methods? **A2:** Yes. P2PDRL policies are more robust, tending to outperform competitors when evaluated on novel domains.
Q3: Can P2PDRL's performance be explained or replicated by more conventional regularisation techniques? **A2:** No. Standard KL or entropy-based regularization cannot explain or replicate P2PDRL's robust performance.

4.2. Environment and Settings

We focus our analysis on five continuous control tasks from OpenAI Gym, including *Walker, Ant, Humanoid, Hopper, and HalfCheetah* (Brockman et al., 2016). All tasks are simulated in MuJoCo. 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 value means more diverse distributions. Given a specific ϵ, dynamic parameters are randomly sampled from an uniform distribution defined by ϵ. Details are listed in appendix A.1.

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

PPO (Schulman et al., 2017). PPO provides the state of the art on-policy Deep RL method, which is also used as the basic learning algorithms across all settings. We train the agent to maximise expected return for the full training distribution: At each iteration, the agent first randomly samples a domain and trajectory data within this domain, then learns to maximise the return in the sampled domain.

Distributed PPO (Heess et al., 2017). In this setting, we have multiple agents that compute gradients locally in the sampled domains and a central learner that collect gradient information and apply updates. At each iteration, each peer independently samples one local domain and gathers gradient information with local data by acting on-policy in their assigned domain. A central learner then updates a global policy with the average of gradients across all domains and reassigns the updated policy to all agents.

Distral (Teh et al., 2017). 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 common behaviour from task-specific policies.

DnC (Ghosh et al., 2018). Originally focused on providing robustness to choice of initial state in single domain learning, DnC partitioned the initial state distribution into multiple sub-domains, and alternates between local policy optimisation steps and global distillation steps. In this work, we modify DnC to train agents in the domains with different dynamics parameters. We do not assume domain knowledge to partition the distribution. Instead domains for DnC agents are sampled from the same distribution as all competitors.

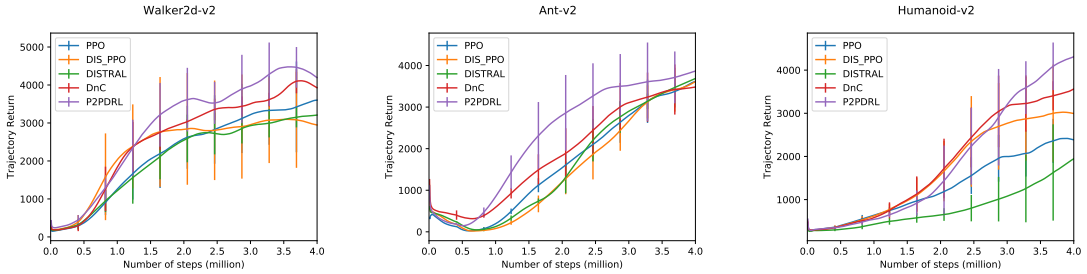
Implementation Details To demonstrate the performance of our proposed method, we by default take $K = 2$ learning agents in parallel, regularised by each other. For the implementation of actor and critic networks, we use two separate MLP networks with two hidden layers, 64 units in each layer. All results are averaged over 8 random seeds. More implementation details are listed in appendix A.2.

Hyperparameters for all PPO-derived methods include the learning rate and the batch size, i.e., number of environment steps sampled for each iteration. These are important due to their impact on convergence/fitting and may require algorithm-specific tuning. Additionally, when comparing sample efficiency, one needs to account for batch-size used by all methods including fairly accounting for samples used by multi-agent methods such as P2PDRL, Distral, DisPPO, and DnC. Since our focus is on generalisation performance, we prioritised optimizing for testing performance. To this end we performed an initial hyperparameter sweep in learning rates and batchsizes for each method independently. The outcome of this sweep in terms of batch size was that all methods obtained their individually best performance when using 4096 total steps (summing across all agents, where applicable) per iteration. Thus our results should be interpreted as all methods being individually tuned for testing performance. However, incidentally, they are also directly comparable in training sample efficiency as the use the same number of samples per iteration.

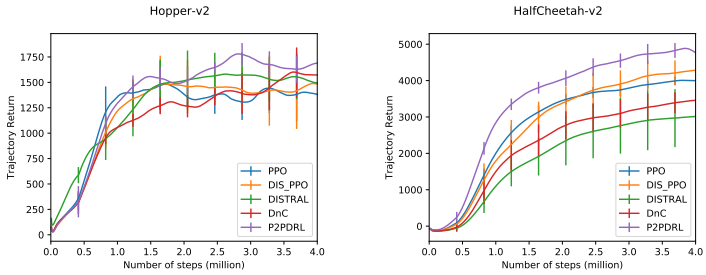
4.3. Training and Generalisation Performance

Generalisation to Novel Environments We evaluate the generalisation performance of the policy learned above, by evaluating it on a novel testing domain with different testing randomisation strengths ϵ^{te}. Fig. 3 compares the training return, represented by learning curves. We show that across all five tasks, P2PDRL can achieve competitive asymptotic performance comparing to baselines. More importantly, Fig. 4 compares the testing performances in a range of different distributions of domains. P2PDRL learned policies are able generally to generalise better against domain shifts to novel environments.

To further understand the training and testing performance, we also illustrate expected return surfaces in Fig. 5. Specifically, we use three reference points in the optimisation trajectory to define a 2-dimensional plane. As you can see from Fig. 5, the first point is the initial policy, the middle point is the policy trained after 8 million timesteps, and the last point is the outcome policy after training. Each point on this plane refers to an affine combination of reference policies. We plot both expected return in training scenario and

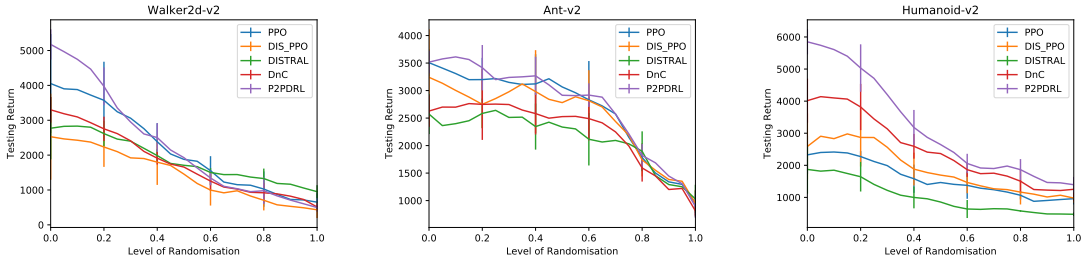


(a) Training return on Walker (b) Training return on Ant (c) Training return on Humanoid

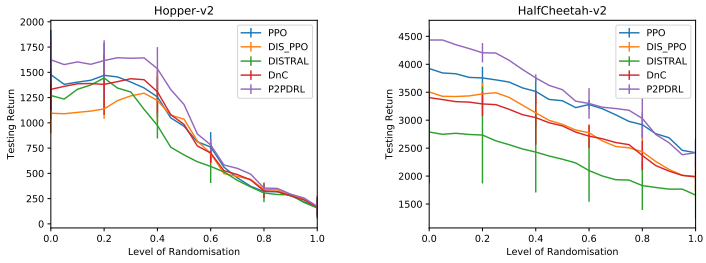


(d) Training return on Hopper (e) Training return on HalfCheetah

Figure 3: Summary of training performances across five different continuous control tasks, represented by learning curves. All experiments are trained with a predefined training randomisation distribution $\epsilon^{tr} = 0.2$. Results are averaged over 8 random seeds.

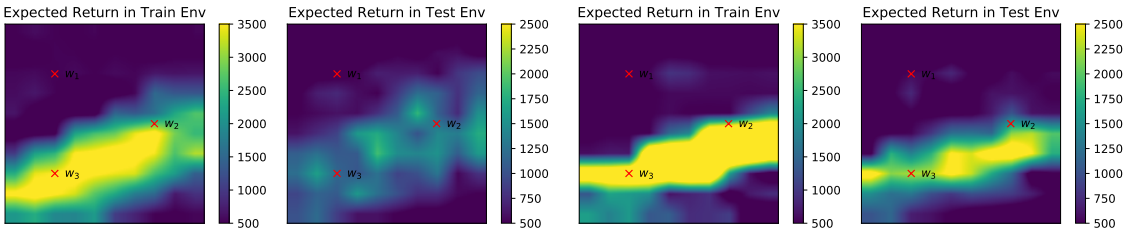


(a) Testing return on Walker (b) Testing return on Ant (c) Testing return on Humanoid



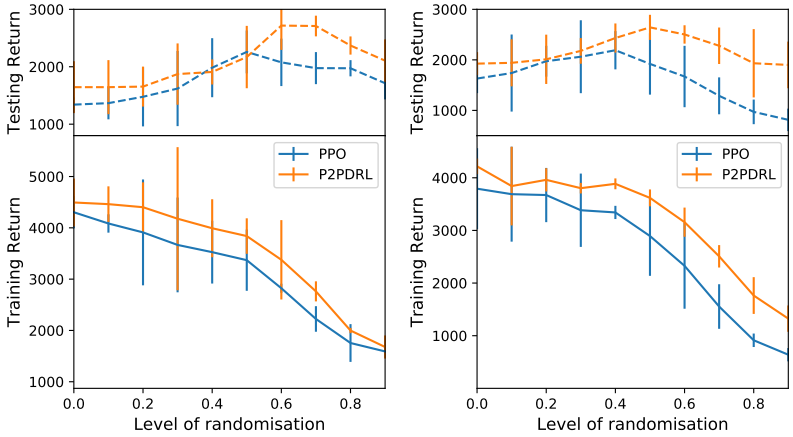
(d) Testing return on Hopper (e) Testing return on HalfCheetah

Figure 4: Summary of generalisation performances across five different continuous control tasks. The figures show 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) PPO Training (b) PPO Testing (c) P2PDRL Training (d) P2PDRL Testing

Figure 5: Illustrations of training and testing return surfaces, trained by PPO and P2PDRL in *Walker* task. Reference policies are: initial policy w_1, trained policy after 8 million steps w_2, and final policy w_3.



(a) Walker task (b) Ant task

Figure 6: 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.

a pre-defined testing scenario for points in these plane. Though both PPO and P2PDRL learn well in training domain, P2PDRL outperforms PPO in terms of testing performance.

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. 2. 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. 6, the training performance levels of PPO and P2PDRL 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.

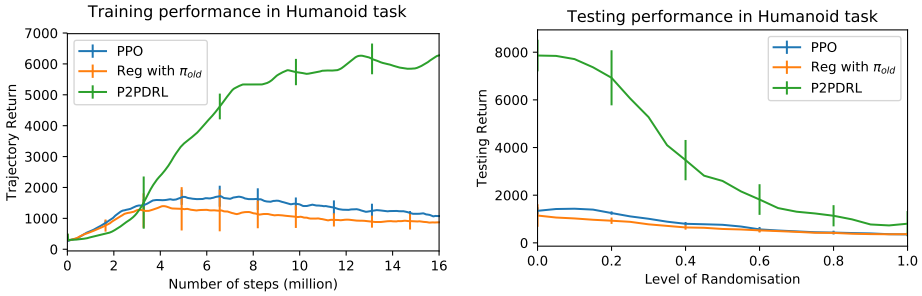
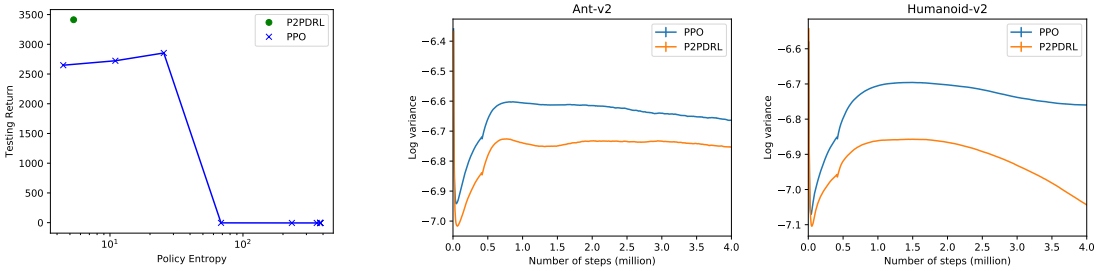


Figure 7: Comparison of P2PDRL with regularised single agent learning in *Humanoid* task. The left figure presents with learning curves and the right figure show the testing performance under variant domain distributions.



(a) Policy Entropy in Walker (b) Variance in Ant (c) Variance in Humanoid

Figure 8: (a): Testing performance of as a function of policy entropy in Walker task. Blue cross: PPO trained policies learned with different entropy regularisers. Green dot: P2PDRL trained policy. (b,c): Log variance of gradient estimation on *Ant* and *Humanoid* task. Policies are trained with PPO and P2PDRL.

4.4. Further Analysis

In this section, we design additional experiments to better understand P2PDRL. The first two experiments test the alternative hypothesis that P2PDRL's good performance is due to a simple regularization effect that can easily be duplicated with standard techniques.

P2PDRL v.s. Regularised Single Agent Learning To investigate the benefit of learning multiple peers, we compare P2PDRL with a single agent setting where the learning is regularised with KL-divergence to itself. More specifically, in the control experiments, we train with only one agent and replace the peer-to-peer distillation loss in equation 3 with KL-divergence to the policy itself π_{old}: $\mathcal{L}_{\text{reg}}(\theta) = \mathbb{E}_s\big[D_{\text{KL}}(\pi_\theta(\cdot|s)||\pi_{\theta_{\text{old}}}(\cdot|s)\big]$. The comparison shown in Fig. 7 shows clear benefit of learning a cohort of agents collaboratively over only applying KL regularised updates in the *Humanoid* task .

P2PDRL v.s. Entropy Regulariser Intuitively, P2PDRL helps prevent the agents converging to a deterministic policy through distilling from peers, and thus natually learns policies with higher entropy. We next compare P2PDRL with entropy regularised PPO in the *Walker* task. We run 11 different entropy regulariser coefficient, ranging from 10^{-3} to 10^2, and report the entropy and testing performances of learned policy. Fig. 8a shows that as

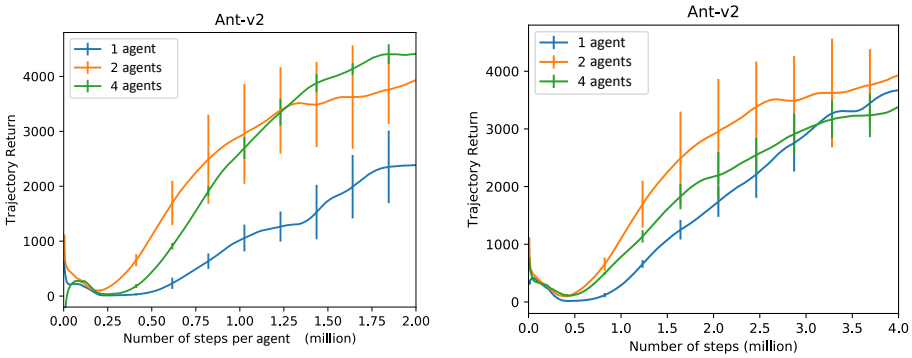


Figure 9: Analysis of number of learning agents in P2PDRL. Left: Controlling the number of timesteps experienced by each learning agent. Right: Controlling the total number of timesteps experienced by all learning agents.

the learned policy entropy increases, the agent initially benefits, but ultimately fails to learn the primary task later where the entropy regulariser increases further and dominates the objective function. More interestingly, we observe that P2PDRL achieves a higher testing performance with a relatively low entropy, suggesting that it learns more general policies, rather than simply regularizing learning through encouraging higher entropy.

Further Analysis on Gradient Variance As discussed in Mehta et al. (2020), conventional domain randomisation techniques suffer from the problem of high variance in gradient estimation. We here empirically analyse the variance of gradient estimator during the training process, comparing P2PDRL with PPO baseline. More specifically, at the start of each iteration, we sample a batch of data with 4096 timesteps and estimate gradients with minibatches of size 64. Gradient variances are computed over 64 sampled minibatches. We plot the quantity of $\log(\mathrm{Var}[\nabla_\theta \mathcal{L}_{\mathrm{PPO}}(\mathcal{B},\theta)])$ in PPO versus $\log(\mathrm{Var}[\nabla_\theta(\mathcal{L}_{\mathrm{PPO}}(\mathcal{B},\theta) + \alpha\mathcal{L}_{\mathrm{dis}}(\mathcal{B},\theta))])$ in P2PDRL, over number of timesteps during training. $\mathcal{B}$ is the sampled minibatch used for estimating gradients. As shown in Fig. 8b, 8c, in both tasks, P2PDRL leads to gradients with lower variances compare to the PPO baseline, suggesting that P2PDRL helps to stablise the training process.

Extension to Larger Cohort The previous experiments simply train small cohorts of $K = 2$ local learning agents, and already achieve strong results. As our primary aim is robustness and not finding opportunities for parallel computing, the impact of a larger cohort size is only of secondary interest. Nevertheless, in this experiment we study how P2PDRL scales with larger cohorts on an example task Ant. We compare learning with $K = 4$, $K = 2$, and a single independent agent (equivalent to PPO), controlling the number of timesteps experienced by *each* peer. In this case, we sample a batch of 2048 steps for each agent. Fig. 9 (left) shows that PPO fails to learn the task with a small batch size, while P2PDRL is able to learn with small batch size, with the guidance from peers. In such settings, we do also see gains from applying P2PDRL with more peers, with $K = 4$ outperforming $K = 2$ in terms of asymtopic performance.

We can also repeat this experiment controlling instead the total number of timesteps across all learning agents per iteration. Each agent thus gathers 1024, 2048 and 4096 timesteps respectively per iteration. As shown in Fig. 9 (right), again, in the case of learning *Ant* task, having larger cohorts is not necessarily better as the batch size becomes too small

for each agent to learn locally. In practice, the size of cohort needs to be designed to balance batch size per agent and knowledge exchange among agents.

5. Conclusion

Generalisation of trained agents to novel domains is a crucial but lacking capability in today's RL. We propose an online distillation based reinforcement learning algorithm P2PDRL. From our experiment results of 5 different continuous control tasks, P2PDRL improved the generalisation performance to novel domains compared to baselines. This is achieved through domain-local RL and global knowledge exchange through peer-to-peer distillation, which provides a more robust training on a wide range of randomised domains.

References

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.

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.

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

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.

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.

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

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

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

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.

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

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.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

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.

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.

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

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.

Samuel Ritter, Jane Wang, Zeb Kurth-Nelson, Siddhant Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been there, done that: Meta-learning with episodic recall. In *International Conference on Machine Learning*, pages 4354–4363. PMLR, 2018.

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.

Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pages 262–270. PMLR, 2017.

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

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

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.

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.

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.

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.

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 *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

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

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018b.

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

Appendix A. Experimental details

A.1. Environment Setup

We evaluate our method with five different benchmark tasks provided by OpenAI gym: *Walker2d, Ant, Humanoid, Hopper, and HalfCheetah*. We modify wind condition (horizontal force), gravity factor, friction coefficient, robot link masses, and initial positions for generating domains with dynamic variants. We introduce a scalar $\epsilon \in [0, 1]$ to control the variances of domain distributions. Given a specific ϵ value, the dynamic parameter distributions are defined in table 1.

Table 1: List of randomised distributions.

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

A.2. Hyperparameter Choices

For PPO and Distributed PPO, we run a hyperparameter sweep on learning rate β : $[1\mathrm{e}-4,\ 3\mathrm{e}-4,\ 1\mathrm{e}-3,\ 3\mathrm{e}-3,\ 1\mathrm{e}-2]$ and batchsizes: $[2048, 4096, 8192, 16384]$. For Distral, DnC and P2PDRL, we run a hyper-parameter sweep also on distillation loss coefficient $\alpha : [0.1, 0.3, 1.0, 3.0, 10.0]$, in addition to learning rate and batchsizes. We report the best performing set of hyperparameters: for both PPO and distributed PPO, we use learning rate of $3\mathrm{e}-4$ and batchsizes of 4096; and for Distral, DnC and P2PDRL, we use learning rate of $1\mathrm{e}-4$, batchsizes of 4096, and distillation loss coefficient of 1.0. Other hyperparameter choices are listed in table 2.

Table 2: List of hyperparameter choices.

Hyperparameter	Value
Actor hidden layers	$[64, 64]$
Actor activation	tanh
Critic hidden layers	$[64, 64]$
Critic activation	tanh
Num. of epochs per iteration	10
Num. of minibatches per epoch	32
Clipping parameter	0.2
Discount factor	0.99
GAE parameter	0.95
Optimiser	Adam