Policy Consolidation for Continual Reinforcement Learning

Christos Kaplanis
Imperial College London

Murray Shanahan
Imperial College London / DeepMind

Claudia Clopath
Imperial College London

Abstract

We propose a method for tackling catastrophic forgetting in deep reinforcement learning that is *agnostic* to the timescale of changes in the distribution of experiences and does not require knowledge of task boundaries. Our *policy consolidation* model improves continual learning when compared to baselines on a number of continuous control tasks.

1 Introduction

An agent that continuously builds on its experiences to develop increasingly complex skills should be able to adapt quickly while also preserving its previously acquired knowledge. While artificial neural networks excel at learning from stationary datasets, once the i.i.d. assumption is violated, they have been shown to suffer from *catastrophic forgetting* [8], whereby training on new data quickly erases knowledge acquired from older data.

Evaluation of techniques that tackle the catastrophic forgetting problem has largely focused on sequential learning of *distinct* tasks. As such, many successful approaches have relied on the awareness of *task boundaries* for the consolidation of previous knowledge [10, 6, 7]. This is problematic because sometimes the data distribution faced by a neural network during training may evolve in a gradual way that can not be discretised easily into separate tasks. In reinforcement learning (RL), for example, the data distribution can change at multiple and unpredictable timescales within the training of a *single* task. This is due to the fact that states are often correlated in time and due to the fact that the agent's policy, which can greatly affect the distribution of its experiences, evolves during training.

In this paper, we develop an approach to mitigate catastrophic forgetting in a deep RL setting by consolidating the agent's policy over multiple timescales during training, which we call *policy consolidation* (PC). Rather than relying on task boundaries, consolidation in our model occurs at all times with the agent's policy being continually distilled into a cascade of hidden networks that evolve over a range of timescales. The hidden networks, in turn, distill back through the cascade into the policy network in order to make sure the agent's policy does not deviate too much from where it was previously. When compared to baseline agents, we find that the PC agent demonstrates improved continual learning when trained on a number of continuous control tasks, both when training is alternated between two tasks and also sometimes during the training of a single task.

2 From Synaptic Consolidation to Policy Consolidation

Previously, we adapted a model of multi-timescale learning at the *synaptic* level to show that it can alleviate forgetting in RL agents [5, 1]. One of the theoretical limitations of this model was that, while it improved memory at the level of individual parameters, due to a highly nonlinear relationship

2nd Continual Learning Workshop, Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

between the parameters and the output of the network, this would not guarantee improved *behavioural* memory of the agent. The PC agent addresses this limitation by consolidating memory directly at the behavioural level. Our model can also be viewed as an extension of one of the Proximal Policy Optimization (PPO) algorithms for RL [11], which ensure that the policy does not change too much with every policy gradient step - in a sense, preventing catastrophic forgetting at a very short timescale. The PC agent operates on the same principle, except that its policy is constrained to be close to where it was at several stages in its history, rather than just at the previous step. It should be noted that the technique of distilling policies from one network to another has been used before for compression [9] and multitask learning [12].

Below we motivate and derive the policy consolidation framework: first, we reinterpret the synaptic consolidation model from [1] and [5] by incorporating it into the objective function of the RL agent and then we combine it with concepts from PPO in order to directly consolidate the agent's behavioural memory at a range of timescales.

2.1 Synaptic Consolidation

The synaptic model from [1] was originally described with an analogy to a chain of communicating beakers of liquid (Figure 1a). The level of liquid in the first beaker corresponds to the visible synaptic weight, i.e. the one that is used for neural computation. Liquid can be added or subtracted from this beaker in accordance with any synaptic learning algorithm. The remaining beakers in the chain correspond to 'hidden' synaptic variables, which have two simultaneous functions: (i) the flow of liquid from shallower to deeper beakers *record* the value of the synaptic weight at a wide range of timescales, and (ii) the flow from deeper beakers back through the shallower ones *regularise* the synaptic weight by its own history, constraining it to be close to where it was previously. The wide range of timescales is implemented by letting the tube widths between beakers decrease exponentially and the beaker widths to grow exponentially as one traverses deeper into the chain.

The synaptic consolidation process can be formally described with a set of first-order linear differential equations, which can be translated into discrete updates with the Euler method as follows:

$$u_1 \leftarrow u_1 + \frac{\eta}{C_1}\left(\Delta w + g_{1,2}(u_2 - u_1)\right) \qquad k = 1 \tag{1}$$

$$u_k \leftarrow u_k + \frac{\eta}{C_k}\left(g_{k-1,k}(u_{k-1} - u_k) + g_{k,k+1}(u_{k+1} - u_k)\right) \qquad k = 2, ..., N \tag{2}$$

where u_k coresonds to the value of the kth variable in the chain, the $g_{k,k+1}$s correspond to the tube widths, the C_ks correspond to the beaker widths, Δw corresponds to the learning update and η is the learning rate. Now consider, as in [5], that we apply this model to the parameters of a deep neural network that encodes the policy of an RL agent. Let $\mathcal{L}(\mathbf{U}_1)$ be the RL objective the network is being trained to minimise, where $\mathbf{U}_k$ denotes a vector of the kth beaker values for all the parameters. If we define a new loss function $\mathcal{L}^*(\mathbf{U})$ as follows:

$$\mathcal{L}^*(\mathbf{U}) = \mathcal{L}(\mathbf{U}_1) + \frac{1}{2}g_{1,2}||\mathbf{U}_1 - \mathbf{U}_2||_2^2 + \frac{1}{2}g_{2,3}||\mathbf{U}_2 - \mathbf{U}_3||_2^2 + ... + \frac{1}{2}g_{N-1,N}||\mathbf{U}_{N-1} - \mathbf{U}_N||_2^2 \tag{3}$$

then we notice that, by differentiating it with respect to $\mathbf{U}_k$, a negative gradient step with learning rate $\frac{\eta}{C_k}$ implements the consolidation updates in Equations 1 and 2 since

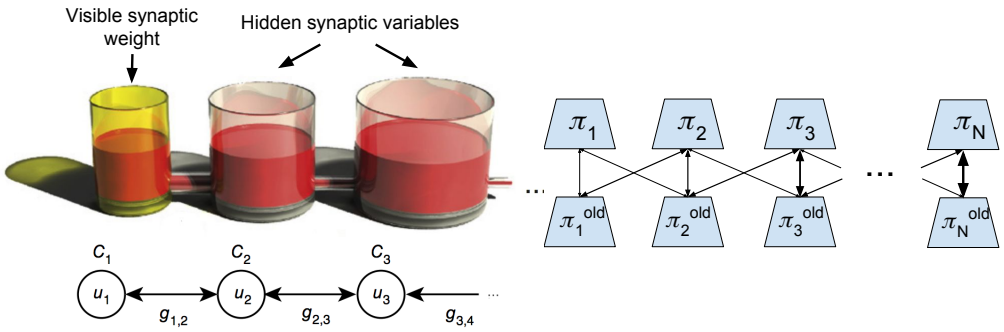
$$-\nabla_{\mathbf{U}_k}\mathcal{L}^*(\mathbf{U}) = g_{k-1,k}(\mathbf{U}_{k-1} - \mathbf{U}_k) + g_{k,k+1}(\mathbf{U}_{k+1} - \mathbf{U}_k) \tag{4}$$

and a step in the direction of $-\nabla_{\mathbf{U}_1}\mathcal{L}(\mathbf{U}_1)$ corresponds to Δw in Equation 2. Thus we can view the synaptic consolidation model as a process that minimises the *Euclidean distance* between the vector of parameters and its own history at different timescales. It is not obvious, however, that Euclidean distance in parameter space is a good measure of *behavioural* dissimilarity of the agent from its past.

2.2 Policy Consolidation

Since each parameter has its own chain of hidden variables, each vector $\mathbf{U}_k$ actually defines its own neural network and therefore its own policy, which we can denote π_k. With this view, we propose a new loss function that replaces the Euclidean distances in parameter space with a notion of distance in *policy space* between adjacent sets of beakers:

$$\mathcal{L}^*(\pi_1, ..., \pi_N) = \mathcal{L}(\pi_1) + \mathbb{E}_{s_t \sim \rho_1}\Big[g_{1,2}D_{\mathrm{KL}}\left(\pi_1||\pi_2\right) + g_{2,3}D_{\mathrm{KL}}\left(\pi_2||\pi_3\right) + ... + g_{N-1,N}D_{\mathrm{KL}}\left(\pi_{N-1}||\pi_N\right)\Big] \tag{5}$$



(a) Synaptic Consolidation (b) Policy Consolidation

Figure 1: (a) Depiction of synaptic consolidation model (adapted from [1]) (b) Depiction of policy consolidation model. The arrows linking the π_ks to the π_k^{old}s represent KL constraints between them, with thicker arrows implying larger constraints, enforcing the policies to be closer together.

where ρ_1 is the state distribution induced by following policy π_1 and the $D_{\mathrm{KL}}\left(\pi_k||\pi_{k+1}\right)$ terms refer to the Kullback-Leibler (KL) divergence between the action distributions (given state s_t) of adjacent policies in the chain.

In order to implement policy consolidation in a practical RL algorithm, we adapt one of the PPO algorithms [11]. The fixed KL version of PPO maximises the following objective at each policy update:

$$L^{KL}(\theta) = \mathbb{E}_t\Big[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta D_{\mathrm{KL}}\left(\pi_{\theta_{old}}(\cdot|s_t)||\pi_\theta(\cdot|s_t)\right)\Big] \tag{6}$$

where θ_{old} refers to the parameters of the network before the policy update and $\hat{A}_t$ is an estimator of the advantage function. The larger the KL coefficient β is, the smaller the policy updates are at each iteration. In our model, we use the same policy gradient as in PPO and introduce similar KL terms for each π_k with coefficients that increase exponentially for deeper beakers. This ensures that the deeper beakers evolve at longer timescales in *policy* space - i.e. the equivalent of the C_k beaker width terms in the synaptic consolidation model. We found in our experiments that the PC agent's performance was often more stable when we used the reverse KL constraint $\mathrm{KL}\left(\pi_k||\pi_{k_{old}}\right)$, and so we used following objective for the PC model (discussed in more detail in Appendix B):

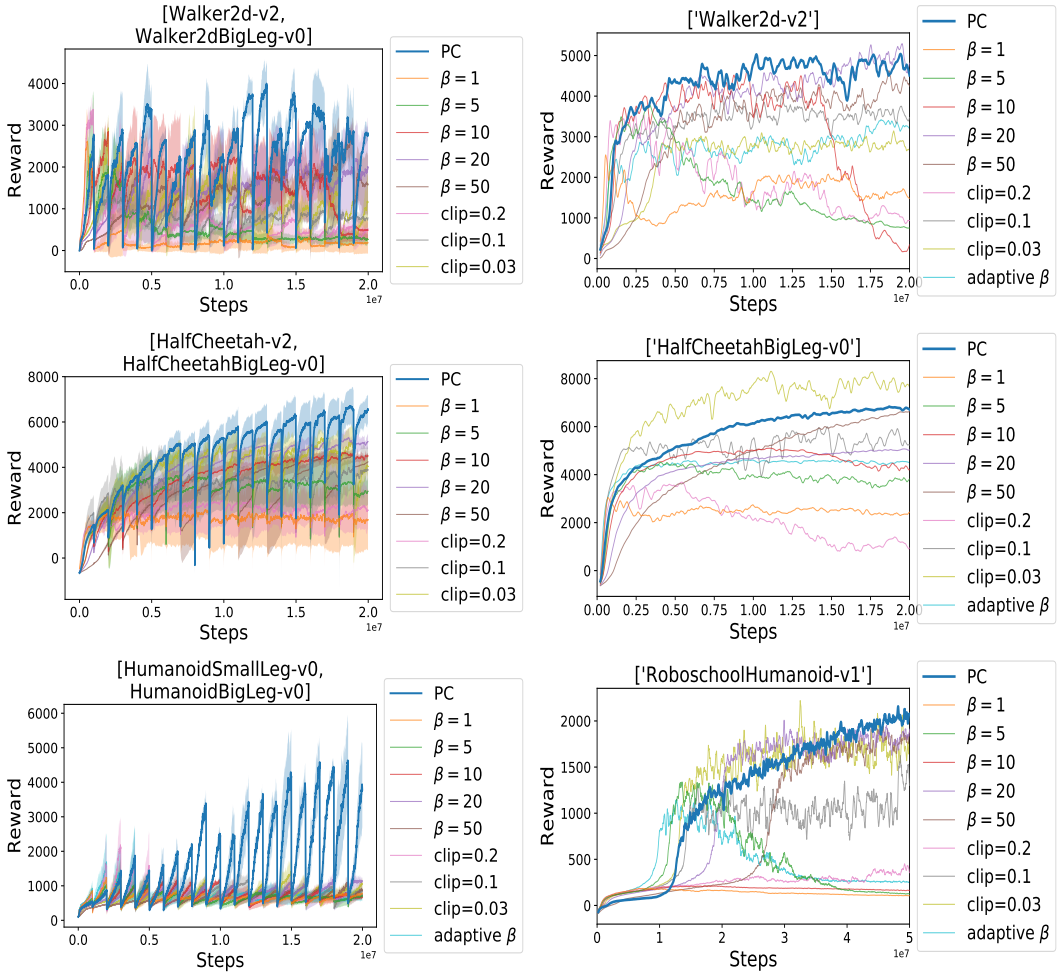
$$L^{PC}(\pi_1, ..., \pi_N) = \mathbb{E}_t\Bigg[\frac{\pi_1}{\pi_{1_{old}}}\hat{A}_t - \sum_{k=1}^{N}\Big[\beta\omega^{k-1}D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)\Big]$$
$$- \sum_{k=2}^{N-1}\Big[\omega D_{\mathrm{KL}}\left(\pi_k|\,|\pi_{k-1_{old}}\right) + D_{\mathrm{KL}}\left(\pi_k|\,|\pi_{k+1_{old}}\right)\Big] - \omega_{1,2}D_{\mathrm{KL}}\left(\pi_1|\,|\pi_{2_{old}}\right) - D_{\mathrm{KL}}\left(\pi_N|\,|\pi_{N-1_{old}}\right)\Bigg] \tag{7}$$

where $\omega_{1,2}$ controls how much the agent's policy is regularised by its history and ω determines the ratio of timescales of consecutive beakers. Each policy is constrained to be close to the *old* versions of neighbouring policies in order to ensure a stable optimisation at each iteration (Figure 1b).

3 Results

In order to test how effective the model is for alleviating catastrophic forgetting, we evaluated the performance of a PC agent on a number of continuous control tasks [2, 4] in two RL settings: (i) one where training was alternated periodically between two tasks and (ii) in the context of training just one task. Baseline agents were also trained using the fixed KL[1], adaptive KL[1] and clipped versions of PPO [3] with a range of βs and clip coefficients for comparison. The PC agent used the same hyperparameters for each run, except for the learning rate which was lower for all agents (including the baselines) for the Humanoid tasks. Full implementational details are included in Appendix A.

[1] The results presented in this section use the reverse KL constraint for the baselines since that was what was used for the PC agent, but we observed similar results with the original PPO objective (Appendix B)



(a) Alternating tasks

(b) Single task

Figure 2: Reward over time for (a) 2- and (b) 1-task runs; comparison of PC agent with fixed KL (with different βs), clipped (with different clip coeff) and adaptive KL agents (omitted for some runs since return went very negative). For 2-task runs, means and s.d. errorbars over 3 runs are shown.

In the two-task setting, the PC agent performs better on average than any of the controls, showing a particularly stark improvement in the Humanoid tasks where none of the controls were able to successfully learn both tasks (Figure 2a). In the one task setting, the PC agent does as well or better than the baselines in all tasks except for one in the HalfCheetah task. The PC agent also seems to exhibit lower variance of return during training than most of the baselines (Figure 2b).

4 Future Work

As a next step, we would like to evaluate the model on more tasks and in the context of different switching schedules between tasks, in order to test its ability to deal with multiple timescales of change in the data distribution. It would be especially interesting to test the model in an adversarial setting such as multi-agent RL, which can involve a high degree of nonstationarity in the environment. We would also like to test the effects of varying the hyperparameters of the model, i.e. the directionality of the KL constraints, the number of beakers, $\omega_{1,2}$ and ω, as well as comparing its performance to other consolidation models (e.g. [5, 6]). One limitation of the model is that it takes a long time for older policies to filter back into the agent's behaviour. It would be interesting to investigate whether the hidden beakers could be used directly for the agent's policy. Finally, it might be useful to augment the model with a way of *prioritising* consolidation of important aspects of the policy, e.g. the most valuable actions.

References

[1] M. K. Benna and S. Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19(12):1697–1706, 2016.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[4] P. Henderson, W.-D. Chang, F. Shkurti, J. Hansen, D. Meger, and G. Dudek. Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*, 2017.

[5] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2497–2506, StockholmsmÃssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/kaplanis18a.html`.

[6] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[7] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[8] M. McCloskey and J. N. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

[9] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[10] P. Ruvolo and E. Eaton. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pages 507–515, 2013.

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

[12] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

A Details of Implementation

For the baseline models, we mainly used the hyperparameters used for the training of Mujoco tasks in [11]. The value function network shared parameters with the policy network and no task-id input was given to the agents. In [11], different parameters were used for the Humanoid tasks as well as multiple actors - for simplicity we used the Mujoco parameters and a single actor. The hidden policies were all initialised with the same parameters as the visible policy for the PC agent, which means that the beginning of training can be slow as the agent is over-consolidated at the initial weights. This might be remedied in the future by introducing incremental flow from the deeper beakers as training progresses. Table 1 shows a list of hyperparameters used for the experiments. In future, we would like to do a broader parameter search for both the baselines and the policy consolidation model. For this work, many more baselines were run than policy consolidation agents in the interest of fairness.

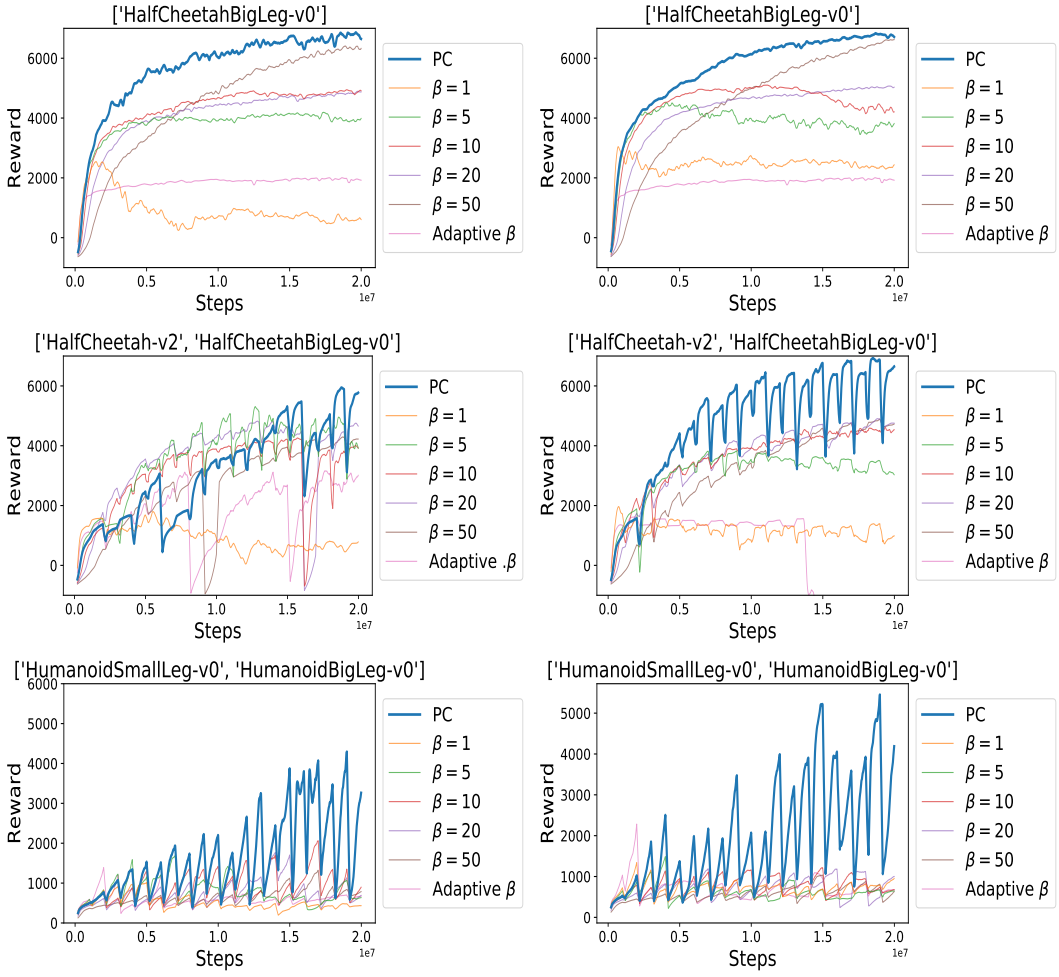
Table 1: Hyperparameters

PARAMETER	MULTI-TASK	SINGLE TASK
# TASK SWITCHES	19	0
# TIMESTEPS/TASK	1M	50M (HUMANOID) / 20M (OTHERS)
DISCOUNT γ	0.99	0.99
GAE PARAMETER (λ)	0.95	0.95
HORIZON	2048	2048
ADAM STEPSIZE (KTH POLICY)	$\omega^{1-k} \times 3 \times 10^{-4}$	$\omega^{1-k} \times 3 \times 10^{-5}$
VF COEFFICIENT	0.5	0.5
# EPOCHS PER UPDATE	10	10
MINIBATCH SIZE	64	64
NEURON TYPE	RELU	RELU
WIDTH HIDDEN LAYER 1	64	64
WIDTH HIDDEN LAYER 2	64	64
ADAM β_1	0.9	0.9
ADAM β_2	0.999	0.999
# HIDDEN POLICIES	7	7
$\omega_{1,2}$	1	1
ω	4	4
β (POL.CONS.)	0.5	0.5
ADAPTIVE KL d_{targ}	0.01	0.01

B Directionality of KL constraint

In our initial experiments we found that using a $D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)$ constraint for each policy in the PC model, rather than the $D_{\mathrm{KL}}\left(\pi_{k_{old}}||\pi_k\right)$ constraint used in the KL versions of PPO [11], resulted in better continual learning and so in the main results section we compared the PC model with KL baselines that also used the $D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)$ constraint. Here we show in a few experiments that we get the same qualitative improvements from the PC if we use the original KL constraint from PPO for both the PC model and the baselines (Figure 3). As can be seen particularly in the HalfCheetah and Humanoid alternating task settings, the $D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)$ version performs better.

The effect of the directionality of this KL constraint, as well as the directionality of the KL constraints between adjacent policies (of which there are four possible combinations) warrants further investigation and is an important avenue for future work.



(a) $D_{\mathrm{KL}}\left(\pi_{k_{old}}||\pi_k\right)$ (b) $D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)$

Figure 3: Reward over time using the (a) $D_{\mathrm{KL}}\left(\pi_{k_{old}}||\pi_k\right)$ and (b) $D_{\mathrm{KL}}\left(\pi_k||\pi_{k_{old}}\right)$ constraints.