Multi-Task Deep Reinforcement Learning with PopArt

Matteo Hessel
DeepMind

Hubert Soyer
DeepMind

Lasse Espeholt
DeepMind

Wojciech Czarnecki
DeepMind

Simon Schmitt
DeepMind

Hado van Hasselt
DeepMind

Abstract

The reinforcement learning (RL) community has made great strides in designing algorithms capable of exceeding human performance on specific tasks. These algorithms are mostly trained one task at the time, each new task requiring to train a brand new agent instance. This means the learning algorithm is general, but each solution is not; each agent can only solve the one task it was trained on. In this work, we study the problem of learning to master not one but multiple sequential-decision tasks at once. A general issue in multi-task learning is that a balance must be found between the needs of multiple tasks competing for the limited resources of a single learning system. Many learning algorithms can get distracted by certain tasks in the set of tasks to solve. Such tasks appear more salient to the learning process, for instance because of the density or magnitude of the in-task rewards. This causes the algorithm to focus on those salient tasks at the expense of generality. We propose to automatically adapt the contribution of each task to the agent's updates, so that all tasks have a similar impact on the learning dynamics. This resulted in state of the art performance on learning to play all games in a set of 57 diverse Atari games. Excitingly, our method learned a single trained policy - with a single set of weights - that exceeds median human performance. To our knowledge, this was the first time a single agent surpassed human-level performance on this multi-task domain. The same approach also demonstrated state of the art performance on a set of 30 tasks in the 3D reinforcement learning platform DeepMind Lab.

Introduction

In recent years, the field of deep reinforcement learning (RL) has enjoyed many successes. Deep RL agents have been applied to board games such as Go (Silver et al. 2016) and chess (Silver et al. 2017), continuous control tasks (Lillicrap et al. 2016; Duan et al. 2016), classic video-games such as Atari (Mnih et al. 2015; Hessel et al. 2018; Gruslys et al. 2018; Schulman et al. 2015; 2017; Bacon, Harb, and Precup 2017), and 3D first person environments (Mnih et al. 2016; Jaderberg et al. 2016). While the results are impressive, they were achieved on one task at the time, each new task requiring to train a brand new agent instance from scratch.

Multi-task and transfer learning remain important open problems in deep RL. There are at least four different strains

of multi-task reinforcement learning that have been explored in the literature: off-policy learning of many predictions about the same stream of experience (Schmidhuber 1990; Sutton et al. 2011; Jaderberg et al. 2016), continual learning in a sequence of tasks (Ring 1994; Thrun 1996; 2012; Rusu et al. 2016), distillation of task-specific experts into a single shared model (Parisotto, Ba, and Salakhutdinov 2015; Rusu et al. 2015; Schmitt et al. 2018; Teh et al. 2017), and parallel learning of multiple tasks at once (Sharma and Ravindran 2017; Caruana 1998). We will focus on the latter.

Parallel multi-task learning has recently achieved remarkable success in enabling a single system to learn a large number of diverse tasks. The Importance Weighted Actor-Learner Architecture, henceforth IMPALA (Espeholt et al. 2018), achieved a 59.7% median human normalised score across 57 Atari games, and a 49.4% mean human normalised score across 30 DeepMind Lab levels. These results are state of the art for multi-task RL, but they are far from the human-level performance demonstrated by deep RL agents on the same domains, when trained on each task individually.

Part of why multi-task learning is much harder than single task learning is that a balance must be found between the needs of multiple tasks, that compete for the limited resources of a single learning system (for instance, for its limited representation capacity). We observed that the naive transposition of common RL algorithms to the multi-task setting may not perform well in this respect. More specifically, the *saliency* of a task for the agent increases with the scale of the *returns* observed in that task, and these may differ arbitrarily across tasks. This affects value-based algorithms such as Q-learning (Watkins 1989), as well as policy-based algorithms such as REINFORCE (Williams 1992).

The problem of scaling individual rewards appropriately is not novel, and has often been addressed through reward clipping (Mnih et al. 2015). This heuristic changes the agent's objective, e.g., if all rewards are non-negative the algorithm optimises *frequency* of rewards rather than their cumulative sum. If the two objectives are sufficiently well aligned, clipping can be effective. However, the scale of returns also depends on the rewards' sparsity. This implies that, even with reward clipping, in a multi-task setting the magnitude of updates can still differ significantly between tasks, causing some tasks to have a larger impact on the learning dynamics than other equally important ones.

Note that both the sparsity and the magnitude of rewards collected in an environment are inherently *non-stationary*, because the agent is learning to actively maximise the total amount of rewards it can collect. These non-stationary learning dynamics make it impossible to normalise the learning updates a priori, even if we would be willing to pour significant domain knowledge into the design of the algorithm.

To summarise, in IMPALA the magnitude of updates resulting from experience gathered in each environment depends on: 1) the *scale* of rewards, 2) the *sparsity* of rewards, 3) the *competence* of the agent. In this paper we use PopArt normalisation (van Hasselt et al. 2016) to derive an actor-critic update invariant to these factors, enabling large performance improvements in parallel multi-task agents. We demonstrated this on the Atari-57 benchmark, where a single agent achieved a median normalised score of 110% and on DmLab-30, where it achieved a mean score of 72.8%.

Background

Reinforcement learning (RL) is a framework for learning and decision-making under uncertainty (Sutton and Barto 2018). A learning system - the *agent* - must learn to interact with the *environment* it is embedded in, so as to maximise a scalar *reward* signal. The RL problem is often formalised as a Markov decision process (Bellman 1957): a tuple $(\mathcal{S}, \mathcal{A}, p, \gamma)$, where $\mathcal{S}, \mathcal{A}$ are finite sets of *states* and *actions*, p denotes the dynamics, such that $p(r, s' \mid s, a)$ is the probability of observing reward r and state s' when executing action a in state s, and $\gamma \in [0, 1]$ discounts future rewards. The *policy* maps states $s \in \mathcal{S}$ to probability distributions over actions $\pi(A|S = s)$, thus specifying the behaviour of the agent. The *return* $G_t = R_{t+1} + \gamma R_{t+2} + \ldots$ is the γ-discounted sum of rewards collected by an agent from state S_t onward under policy π. We define *action values* and *state values* as $q^\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a]$ and $v^\pi(s) = E_\pi[G_t \mid S_t = s]$, respectively. The agent's objective is to find a policy to maximise such values.

In multi-task reinforcement learning, a single agent must learn to master N different environments $T = \{D_i = (\mathcal{S}_i, \mathcal{A}_i, p_i, \gamma)\}_{i=1}^N$, each with its own distinct dynamics (Brunskill and Li 2013). Particularly interesting is the case in which the action space and transition dynamics are at least partially shared. For instance, the environments might follow the same *physical* rules, while the set of interconnected states and obtainable rewards differ. We may formalise this as a single larger MDP, whose state space is $\mathcal{S} = \{\{(s_j, i)\}_{s_j \in \mathcal{S}_i}\}_{i=1}^N$. The task index i may be latent, or may be exposed to the agent's policy. In this paper, we use the task index at training time, for the value estimates used to compute the policy updates, but not at testing time: our algorithm will return a single general policy $\pi(A|S)$ which is only function of the individual environment's state S and not conditioned directly on task index i. This is more challenging than the standard multi-task learning setup, which typically allows conditioning the model on the task index even at evaluation (Romera-Paredes et al. 2013; Collobert and Weston 2008), because our agents will need to infer what task to solve purely from the stream of raw observations and/or early rewards in the episode.

Actor-critic

In our experiments, we use an actor-critic algorithm to learn a policy $\pi_\eta(A|S)$ and a value estimate $v_\theta(s)$, which are both outputs of a deep neural network. We update the agent's policy by using REINFORCE-style stochastic gradient $(G_t - v_\theta(S_t))\nabla_\eta \log \pi(A_t|S_t)$ (Williams 1992), where $v_\theta(S_t)$ is used as a baseline to reduce variance. In addition we use a multi-step return G_t^v that bootstraps on the value estimates after a limited number of transitions, both to reduce variance further and to allow us to update the policy before G_t fully resolves at the end of an episode. The value function $v_\theta(S)$ is instead updated to minimise the squared loss with respect to the (truncated and bootstrapped) return:

$$\Delta\theta \propto -\nabla_\theta(G_t^v - v_\theta(S_t))^2 = (G_t^v - v_\theta(S_t))\nabla_\theta v_\theta(S_t) \quad (1)$$

$$\Delta\eta \propto (G_t^\pi - v_\theta(S_t))\nabla_\eta \log(\pi_\eta(A_t|S_t)), \quad (2)$$

where G_t^v and G_t^π are stochastic estimates of $v^\pi(S_t)$ and $q^\pi(S_t, A_t)$, respectively. Note how both updates depend linearly on the scale of returns, which, as previously argued, depend on scale/sparsity of rewards, and agent's competence.

Efficient multi-task learning in simulation

We use the IMPALA agent architecture (Espeholt et al. 2018), proposed for reinforcement learning in simulated environments. In IMPALA the agent is distributed across multiple threads, processes or machines. Several actors run on CPU generating rollouts of experience, consisting of a fixed number of interactions (100 in our experiments) with their own copy of the environment, and then enqueue the rollouts in a shared queue. Actors receive the latest copy of the network's parameters from the learner before each rollout. A single GPU learner processes rollouts from all actors, in batches, and updates a deep network. The network is a deep convolutional ResNet (He et al. 2015), followed by a LSTM recurrent layer (Hochreiter and Schmidhuber 1997). Policy and values are all linear functions of the LSTM's output.

Despite the large network used for estimating policy π_η and values v_θ, the decoupled nature of the agent enables to process data very efficiently: in the order of hundreds of thousands frames per second (Espeholt et al. 2018). The setup easily supports the multi-task setting by simply assigning different environments to each of the actors and then running the single policy $\pi(S|A)$ on each of them. The data in the queue can also be easily labelled with the task *id*, if useful at training time. Note that an efficient implementation of IMPALA is available open-source [1], and that, while we use this agent for our experiments, our approach can be applied to other data parallel multi-task agents (e.g. A3C).

Off-policy corrections

Because we use a distributed queue-based learning setup, the data consumed by the learning algorithm might be slightly off-policy, as the policy parameters change between acting and learning. We can use importance sampling corrections $\rho_t = \pi(A_t|S_t)/\mu(A_t|S_t)$ to compensate for this (Precup,

---

[1] www.github.com/deepmind/scalable_agent

3797

Sutton, and Singh 2000). In particular, we can write the n-step return as $G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^n v(S_{t+n}) = v(S_t) + \sum_{k=t}^{t+n-1} \gamma^{k-t}\delta_k$, where $\delta_k = R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$, and then apply appropriate importance sampling corrections to each error term (Sutton et al. 2014) to get $G_t = v(S_t) + \sum_{k=t}^{t+n-1} \gamma^{k-t}(\prod_{i=t}^{k} \rho_i)\delta_k$. This is unbiased, but has high variance. To reduce variance, we can further clip most of the importance-sampling ratios, e.g., as $c_t = \min(1, \rho_t)$. This leads to the v-trace return (Espeholt et al. 2018)

$$G_t^{v-trace} = v(S_t) + \sum_{k=t}^{t+n-1} \gamma^{k-t}\left(\prod_{i=t}^{k} c_i\right)\delta_k \quad (3)$$

A very similar target was proposed for the ABQ(ζ) algorithm (Mahmood 2017), where the product $\rho_t\lambda_t$ was considered and then the trace parameter λ_t was chosen to be adaptive to lead to exactly the same behaviour that $c_t = \rho_t\lambda_t = \min(1, \rho_t)$. This shows that this form of clipping does not impair the validity of the off-policy corrections, in the same sense that bootstrapping in general does not change the semantics of a return. The returns used by the value and policy updates defined in Equation 1 and 2 are then

$$G_t^v = G_t^{v-trace} \quad \text{and} \quad G_t^\pi = R_{t+1} + \gamma G_{t+1}^{v-trace}. \quad (4)$$

This is the same algorithm as used by Espeholt et al. (2018) in the experiments on the IMPALA architecture.

Adaptive normalisation

In this section we use PopArt normalisation (van Hasselt et al. 2016), which was introduced for value-based RL, to derive a scale invariant algorithm for actor-critic agents. For simplicity, we first consider the single-task setting, then we extend it to the multi-task setting (the focus of this work).

Scale invariant updates

In order to normalise both baseline and policy gradient updates, we first parameterise the value estimate $v_{\mu,\sigma,\theta}(S)$ as the linear transformation of a suitably *normalised* value prediction $n_\theta(S)$. We further assume that the normalised value prediction is itself the output of a linear function, for instance the last fully connected layer of a deep neural net:

$$v_{\mu,\sigma,\theta}(s) = \sigma \cdot n_\theta(s) + \mu = \sigma \cdot (\underbrace{\boldsymbol{w}^\top f_{\theta\setminus\{w,b\}}(s) + b}_{= n_\theta(s)}) + \mu. \quad (5)$$

As proposed by van Hasselt et al., μ and σ can be updated so as to track mean and standard deviation of the values. First and second moments of can be estimated online as

$$\mu_t = (1-\beta)\mu_{t-1} + \beta G_t^v, \quad \nu_t = (1-\beta)\nu_{t-1} + \beta(G_t^v)^2, \quad (6)$$

and then used to derive the estimated standard deviation as $\sigma_t = \sqrt{\nu_t - \mu_t^2}$. Note that the fixed *decay* rate β determines the horizon used to compute the statistics. We can then use the normalised value estimate $n_\theta(S)$ and the statistics μ and σ to normalise the actor-critic loss, both in its value and policy component; this results in the scale-invariant updates:

$$\Delta\theta \propto \left(\frac{G_t^v - \mu}{\sigma} - n_\theta(S_t)\right) \nabla_\theta n_\theta(S_t), \quad (7)$$

$$\Delta\eta \propto \left(\frac{G_t^\pi - \mu}{\sigma} - n_\theta(S_t)\right) \nabla_\eta \log \pi_\eta(A_t|S_t). \quad (8)$$

If we optimise the new objective naively, we are at risk of making the problem harder: the normalised targets for values are non-stationary, since they depend on statistics μ and σ. The PopArt normalisation algorithm prevents this, by updating the last layer of the normalised value network to preserve unnormalised value estimates $v_{\mu,\sigma,\theta}$, under any change in the statistics $\mu \to \mu'$ and $\sigma \to \sigma'$:

$$\boldsymbol{w}' = \frac{\sigma}{\sigma'}\boldsymbol{w}, \qquad b' = \frac{\sigma b + \mu - \mu'}{\sigma'}. \quad (9)$$

This extends PopArt's scale-invariant updates to the actor-critic setting, and can help to make tuning hyperparameters easier, but it is not sufficient to tackle the challenging multi-task RL setting that we are interested in this paper. For this, a single pair of normalisation statistics is not sufficient.

Scale invariant updates for multi-task learning

Let D_i be an environment in some finite set $T = \{D_i\}_{i=1}^N$, and let $\pi(S|A)$ be a task-agnostic policy, that takes a state S from any of the environments D_i, and maps it to a probability distribution onto the shared action space $\mathcal{A}$. Consider now a multi-task value function $\mathbf{v}(S)$ with N outputs, one for each task. We can use for $\mathbf{v}$ the same parametrisation as in Equation 5, but with vectors of statistics $\boldsymbol{\mu}, \boldsymbol{\sigma} \in R^N$, and a vector-valued function $\mathbf{n}_\theta(s) = (n_\theta^1(s), \ldots, n_\theta^N(s))^\top$

$$\mathbf{v}_{\boldsymbol{\mu},\boldsymbol{\sigma},\theta}(S) = \boldsymbol{\sigma} \odot \mathbf{n}_\theta(S) + \boldsymbol{\mu} = \boldsymbol{\sigma} \odot (\mathbf{W}\mathbf{f}_{\theta\setminus\{\mathbf{W},\mathbf{b}\}}(S) + \mathbf{b}) + \boldsymbol{\mu} \quad (10)$$

where $\mathbf{W}$ and $\mathbf{b}$ denote the parameters of the last fully connected layer in $\mathbf{n}_\theta(s)$. Given a rollout $\{S_{i,k}, A_k, R_{i,k}\}_{k=t}^{t+n}$, generated under task-agnostic policy $\pi_\eta(A|S)$ in environment D_i, we can adapt the updates in Equation 7 and 8 to provide scale invariant updates also in the multi-task setting:

$$\Delta\theta \propto \left(\frac{G_t^{v,i} - \mu_i}{\sigma_i} - n_\theta^i(S_t)\right) \nabla_\theta n_\theta^i(S_t), \quad (11)$$

$$\Delta\eta \propto \left(\frac{G_t^{\pi,i} - \mu_i}{\sigma_i} - n_\theta^i(S_t)\right) \nabla_\eta \log \pi_\eta(A_t|S_t), \quad (12)$$

where the targets $G_t^{\cdot,i}$ use the value estimates for environment D_i for bootstrapping. For each rollout, only the ith head in the value net is updated, while the same policy network is updated irrespectively of the task, using the appropriate rescaling for updates to parameters η. As in the single-task case, when updating the statistics $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ we also need to update $\mathbf{W}$ and $\mathbf{b}$ to preserve unnormalised outputs,

$$\boldsymbol{w}_i' = \frac{\sigma_i}{\sigma_i'}\boldsymbol{w}_i, \quad b_i' = \frac{\sigma_i b_i + \mu_i - \mu_i'}{\sigma_i'}, \quad (13)$$

where $\boldsymbol{w}_i$ is the ith row of matrix $\mathbf{W}$, and μ_i, σ_i, b_i are the ith elements of the corresponding parameter vectors. Note that in all updates only the values, but not the policy, are conditioned on the task index, which ensures that the resulting agent can then be run in a fully task agnostic way, since values are only used to reduce the variance of the policy updates at training time but not needed for action selection.

Figure 1: Summary of results: aggregate scores for IMPALA and PopArt-IMPALA. We report median human normalised score for Atari-57, and mean capped human normalised score for DmLab-30. In Atari, Random and Human refer to whether the trained agent is evaluated with random or human starts. In DmLab-30, test score includes evaluation on the held-out levels.

Agent	Atari-57		Atari-57 (unclipped)		DmLab-30	
	Random	Human	Random	Human	Train	Test
IMPALA	59.7%	28.5%	0.3%	1.0%	60.6%	58.4%
PopArt-IMPALA	110.7%	101.5%	107.0%	93.7%	73.5%	72.8%

Experiments

We evaluated our approach in two challenging multi-task benchmarks, Atari-57 and DmLab-30, based on Atari and DeepMind Lab respectively, and introduced by Espeholt et al. We also consider a new benchmark, consisting of the same 57 Atari games as Atari-57, but with the original unclipped reward scheme. We demonstrate state of the art performance on all three benchmarks. In all cases, to meaningfully aggregate scores across many diverse tasks, we normalise scores based on the scores achieved by a human player and by a random agent on that same task, as common in literature (van Hasselt, Guez, and Silver 2016). All experiments use population-based training (PBT) to adapt hyperparameters as training progresses (Jaderberg et al. 2017).

Domains

Atari-57 is a collection of 57 classic Atari 2600 games. The ALE (Bellemare et al. 2013), exposes them as RL environments. Most prior work has focused on training agents for individual games (Mnih et al. 2015; Hessel et al. 2018; Gruslys et al. 2018; Schulman et al. 2015; 2017; Bacon, Harb, and Precup 2017). Multi-task learning on this platform has not been as successful due to large number of environments, inconsistent dynamics and very different reward structure. Prior work on multi-task RL in the ALE has therefore focused on smaller subsets of games (Rusu et al. 2015; Sharma and Ravindran 2017). Atari has a particularly diverse reward structure. Consequently, it is a perfect domain to fully assess how well can our agents deal with extreme differences in the scale of returns. Thus, we train all agents both with and without reward clipping, to compare performance degradation as returns get more diverse in the unclipped version of the environment. In both cases, at the end of training, we test agents both with *random-starts* (Mnih et al. 2015) and *human-starts* (Nair et al. 2015); aggregate results are reported in Table 1 accordingly.

DmLab-30 is a benchmark consisting of 30 different visually rich, partially observable RL environments (Beattie et al. 2016). This benchmark has strong internal consistency (all levels are played with a first person camera in a 3D environment with consistent dynamics). However, the tasks themselves are quite diverse, and were designed to test distinct skills in RL agents: among these navigation, memory, planning, laser-tagging, and language grounding. The levels can also differ visually in non-trivial ways, as they include both natural environments and maze-like levels. Two levels (`rooms_collect_good_objects` and

`rooms_exploit_deferred_effects`) have held out *test* versions, therefore Table 1 reports both train and test aggregate scores. We observed that the original IMPALA agent suffers from an artificial bottleneck in performance, due to the fact that some of the tasks cannot be solved with the action set available to the agent. As first step, we thus fix this issue by equipping it with a larger action set, resulting in a stronger IMPALA baseline than reported in the original paper. We also run multiple independent PBT experiments, to assess the variability of results across multiple replications.

Atari-57 results

Figures 1 and 2 show the median human normalised performance across the entire set of 57 Atari games in the ALE, when training agent with and without reward clipping, respectively. The curves are plotted as function of the total number of frames seen by each agent.

PopArt-IMPALA (orange line) achieved a median performance of 110% with reward clipping and a median performance of 101% in the unclipped version of Atari-57. Recall that here we are measuring the median performance of a single trained agent across all games, rather than the median over the performance of a set of individually trained agents as it has been more common in the Atari domain. To our knowledge, both agents are the first to surpass median human performance across the entire set of 57 Atari games.

The IMPALA agent (blue line) performed much worse. The baseline barely reached 60% with reward clipping, and the median performance is close to 0% in the unclipped setup. The large decrease in the performance of the baseline IMPALA agent once clipping was removed is in stark contrast with what we observed for PopArt-IMPALA, that achieved almost the same performance in the two training regimes.

Since the level-specific value predictions used by multitask PopArt effectively increase the capacity of the network, we also ran an additional experiment to disentangle the contribution of the increased network capacity from the contribution of the adaptive normalisation. For this purpose, we trained a second baseline, that used level specific value predictions, but did not use PopArt to adaptively normalise the learning updates. Experiments showed that such MultiHead-IMPALA agent (pink line) actually performed slightly worse than the original IMPALA, both with and without clipping, confirming that the performance boost of PopArt-IMPALA is indeed due to the adaptive rescaling.

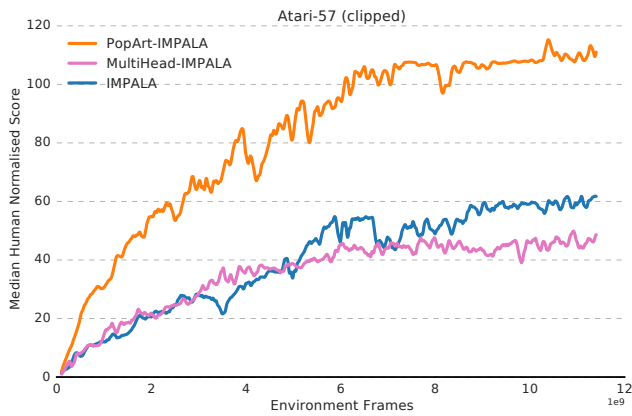We highlight that in our experiments a single instance of multi-task PopArt-IMPALA has processed the same amount

Figure 1: Atari-57 (reward clipping): Median human normalised score across all Atari levels, as function of the total number of frames seen by the agents across all levels. We compare PopArt-IMPALA to IMPALA and to an additional baseline, MultiHead-IMPALA, that uses task-specific value predictions but no adaptive normalisation. All three agent are trained with the clipped reward scheme.
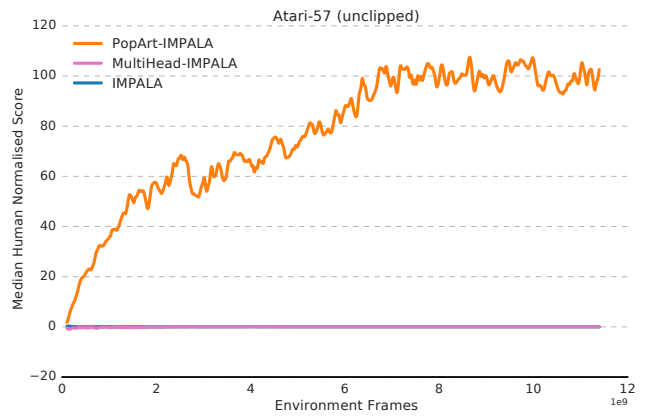

Figure 2: Atari-57 (unclipped): Median human normalised score across all Atari levels, as a function of the total number of frames seen by the agents across all levels. We here compare the same set of agents as in Figure 1, but now all agents are trained without using reward clipping. The approximately flat lines corresponding to the baselines mean no learning at all on at least 50% of the games.

of frames as a collection of 57 expert DQN agents ($57 \times 200 M = 1.14 \times 10^{10}$), while achieving better performance. Despite the large CPU requirements, on a cloud service, training multi-task PopArt-IMPALA can also be competitive in terms of costs, since it exceeds the performance of a vanilla-DQN in just 2.5 days, with a smaller GPU footprint.

Normalisation statistics

It is insightful to observe the different normalisation statistics across games, and how they adapt during the course of training. Figure 3 (top row) plots the *shift* μ for a selection of Atari games, in the unclipped training regime. The *scale* σ is visualised within the same figure by shading the area in the range $[\mu - \sigma, \mu + \sigma]$. We observed that the statistics differed by orders of magnitude across the different games: in crazy_climber the shift exceeded 2500, while in bowling it never went above 15. The adaptivity of the proposed normalisation emerged clearly in crazy_climber and qbert, where the statistics spanned multiple orders of magnitude during training. The bottom row in Figure 3 shows the corresponding agent's undiscounted episode return: it followed the same patterns as the statistics (with differences in magnitude due to discounting). Note how the statistics even tracked the instabilities in the agent's performance, as in qbert, ensuring that appropriate scaling was preserved throughout the experiment.

DmLab-30 results

Figure 4 shows, as a function of the total number of frames processed by each agent, the mean human normalised performance across all 30 DeepMind Lab levels, where each level's score is capped at 100% . For all agents, we ran three independent PBT experiments. In Figure 4 we plot the learning curves for each experiment and, for each agent, fill in the area between the best and worse of these three experiment.
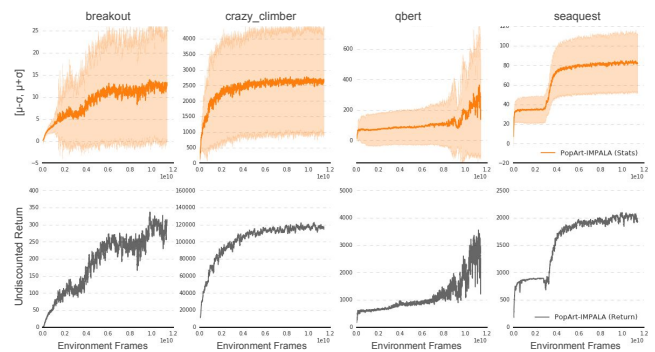

Figure 3: Normalisation statistics: Top: learned statistics, without reward clipping, for four distinct Atari games. The shaded region is $[\mu - \sigma, \mu + \sigma]$. Bottom: undiscounted returns.

Compared to the original paper, our IMPALA baseline used a richer action set, that included more possible horizontal rotations (± 10 and ± 60 in the corresponding dimension of the native DeepMind Lab space), and vertical rotations (± 10). Fine-grained horizontal control is useful on lasertag levels, while vertical rotations are *necessary* for a few psychlab levels. Note that this new baseline (solid blue in Figure 4) performed much better than the original IMPALA agent, which we also trained and reported for completeness (dashed blue). Including PopArt normalisation (in orange) on top of our baseline resulted in largely improved scores. Note how agents achieved clearly separated performance levels, with the new action set dominating the original paper's one, and with PopArt-IMPALA dominating IMPALA for all three replications of the experiment.
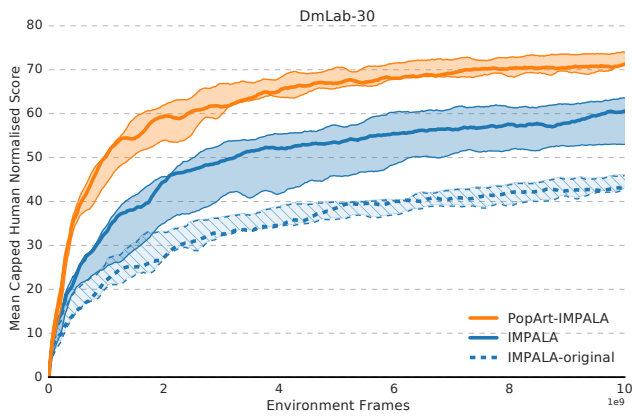
Figure 4: DmLab-30: Mean capped human normalised score of IMPALA (blue) and PopArt-IMPALA (orange), across the DmLab-30 benchmark as function of the number of frames (summed across all levels). Shaded region is bounded by best and worse run among 3 PBT experiments. For reference, we also plot the performance of IMPALA with the limited action set from the original paper (dashed).
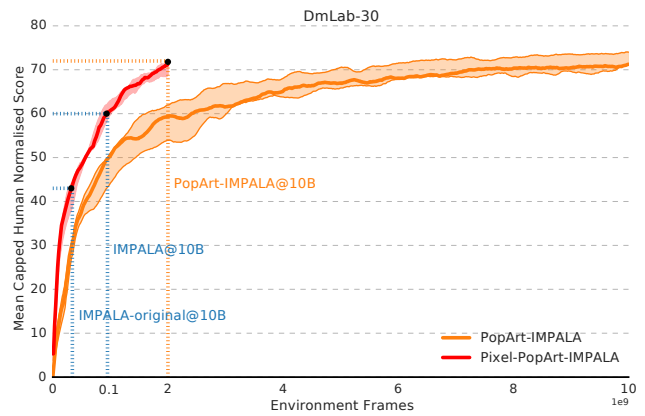

Figure 5: DmLab-30 (with pixel control): Mean capped human normalised score of PopArt-IMPALA with pixel control (red), across the DmLab-30 benchmark as function of the total number of frames across all tasks. Shaded region is bounded by best and worse run among 3 PBT experiments. Dotted lines mark the point where Pixel-PopArt-IMPALA matches PopArt-IMPALA and the two IMPALA baselines.

Extensions

In this section, we explore the combination of the proposed PopArt-IMPALA agent with pixel control (Jaderberg et al. 2016), to further improve data efficiency, and make training IMPALA-like agents on large multi-task benchmarks cheaper and more practical. Pixel control is an unsupervised auxiliary task introduced to help learning good state representations. As shown in Figure 5, the combination of PopArt-IMPALA with pixel control (red line) matched the final performance of the vanilla PopArt-IMPALA (orange line) within a fraction of the data ($\sim 2B$ frames). This was on top of the large improvement in data efficiency already provided by PopArt, meaning that the pixel control augmented PopArt-IMPALA required less than $1/10$-th of the data to match our own IMPALA baseline's performance (and $1/30$-th of the frames to match the original published IMPALA). Importantly, since both PopArt and Pixel Control only add a very small computational cost, this improvement in data efficiency directly translated in a large reduction of the cost of training IMPALA agents on large multi-task benchmarks. Note, finally, that other orthogonal advances in deep RL could also be combined to further improve performance, similarly to what was done by Rainbow (Hessel et al. 2018), in the context of value-based reinforcement learning.

Implementation notes

We implemented all agents in TensorFlow. For each batch of rollouts processed by the learner, we averaged the G_t^v targets within a rollout, and for each rollout in the batch we performed one online update of PopArt's normalisation statistics with *decay* $\beta = 3 \times 10^{-4}$. Note that β didn't require any tuning. To prevent numerical issues, we clipped the *scale* σ in the range $[0.0001, 1e6]$. We did not backpropagate gradients into μ and σ, exclusively updated as

in Equation 6. The weights $\mathbf{W}$ of the last layer of the value function were updated according to Equation 13 and 11. Note that we first applied the actor-critic updates (11), then updated the statistics (6), finally applied output preserving updates (13). For more *just-in-time* rescaling of updates we can invert this order, but this wasn't necessary. In all experiments we used population-based training (PBT) to adapt hyperparameters during the course of training (Jaderberg et al. 2017). As in the IMPALA paper, we used PBT to tune `learning_rate`, `entropy_cost`, the optimiser's `epsilon`, and—in the Atari experiments—the `max_gradient_norm`. In Atari-57 we used populations of 24 instances, in DmLab-30 just 8 instances. For other hyperparameters we used the values from (Espeholt et al. 2018).

Discussion

In this paper we propose a scale-invariant actor-critic algorithm that enables significantly improved performance in multi-task reinforcement learning settings. Being able to acquire knowledge about a wide range of facts and skills has been long considered an essential feature for an RL agent to demonstrate flexible intelligent behaviour (Sutton et al. 2011; Degris and Modayil 2012; Legg and Hutter 2007; Schmidhuber 2013). To ask our algorithms to be capable of mastering multiple tasks is therefore a natural step as we progress towards increasingly powerful agents.

The wide-spread adoption of deep learning in RL is quite timely in this regard, since sharing parts of a neural network across multiple tasks is also a powerful way of building robust representations. This is particularly important for RL, because rewards on individual tasks can be sparse, and therefore sharing representations across tasks can be vital to bootstrap learning. Several agents (Jaderberg et al. 2016; Lample and Chaplot 2016; Shelhamer et al. 2016; Mirowski

et al. 2016) demonstrated this by improving performance on a single external task by learning off-policy about auxiliary tasks defined on the same stream of experience (e.g. pixel control, immediate reward prediction or auto-encoding).

Multi-task learning, as considered in this paper, where we get to execute, in parallel, the policies learned for each task, has potential additional benefits, including deep exploration (Osband et al. 2016), and policy composition (Mankowitz et al. 2018; Todorov 2009; Fernández and Veloso 2006). By learning on-policy about tasks, it may also be easier to scale to much more diverse tasks: if we only learn about some task off-policy from experience generated pursuing a very different one, we might never observe any reward. A limitation of our approach is that it can be complicated, and expensive, to implement parallel learning outside of simulation. However, recent work on parallel training of real world robots (Levine et al. 2016) suggests that this is not necessarily an insurmountable obstacle if sufficient resources are available.

Adoption of parallel multi-task RL has up to now been fairly limited. That the scaling issues considered in this paper, may have been a factor in the limited adoption is indicated by the wider use of this kind of learning in supervised settings (Johnson et al. 2017; Lu et al. 2016; Misra et al. 2016; Hashimoto et al. 2016), where loss functions are naturally well scaled (e.g. cross entropy), or can be easily scaled thanks to the stationarity of the training distribution. We therefore hope and believe that the work presented here can enable more research on multi-task RL.

We also believe that PopArt's adaptive normalisation can be combined with other research in multi-task RL, that previously did not scale as effectively to large numbers of diverse tasks. Among the ideas that may be fruitfully combined we highlight policy distillation (Parisotto, Ba, and Salakhutdinov 2015; Rusu et al. 2015; Schmitt et al. 2018; Teh et al. 2017) and active sampling of the task distribution the agent trains on (Sharma and Ravindran 2017). The combination of PopArt-IMPALA with forms of active sampling might be particularly promising since it may allow to make more efficient use of the parallel data generation, focusing it on the task most amenable for learning. Elastic weight consolidation (Kirkpatrick et al. 2017) and other work from the continual learning literature (Ring 1994; Mcclelland, Mcnaughton, and O'Reilly 1995) might also be adapted to parallel learning setups to reduce interference (Mccloskey and Cohen 1989; French 1999) among tasks.

References

Bacon, P.; Harb, J.; and Precup, D. 2017. The option-critic architecture. *AAAI Conference on Artificial Intelligence*.

Beattie, C.; Leibo, J. Z.; Teplyashin, D.; Ward, T.; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; Schrittwieser, J.; Anderson, K.; York, S.; Cant, M.; Cain, A.; Bolton, A.; Gaffney, S.; King, H.; Hassabis, D.; Legg, S.; and Petersen, S. 2016. Deepmind lab. *CoRR* abs/1612.03801.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *JAIR*.

Bellman, R. 1957. A markovian decision process. *Journal of Mathematics and Mechanics*.

Brunskill, E., and Li, L. 2013. Sample complexity of multi-task reinforcement learning. *CoRR* abs/1309.6821.

Caruana, R. 1998. Multitask learning. In *Learning to learn*. Kluwer Academic Publishers.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Degris, T., and Modayil, J. 2012. Scaling-up knowledge for a cognizant robot. In *AAAI Spring Symposium: Designing Intelligent Robots*.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *ICML*.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; Legg, S.; and Kavukcuoglu, K. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*.

Fernández, F., and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*.

Gruslys, A.; Azar, M. G.; Bellemare, M. G.; and Munos, R. 2018. The reactor: A sample-efficient actor-critic architecture. *ICLR*.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR* abs/1611.01587.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. *AAAI Conference on Artificial Intelligence*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *CoRR* abs/1611.05397.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; and Kavukcuoglu, K. 2017. Population based training of neural networks. *CoRR* abs/1711.09846.

Johnson, M.; Schuster, M.; Le, Q. V.; Krikun, M.; Wu, Y.; Chen, Z.; Thorat, N.; Viégas, F. B.; Wattenberg, M.; Corrado, G.; Hughes, M.; and Dean, J. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* 5.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS*.

Lample, G., and Chaplot, D. S. 2016. Playing FPS games with deep reinforcement learning. *CoRR* abs/1609.05521.

Legg, S., and Hutter, M. 2007. Universal intelligence: A definition of machine intelligence. *Minds Mach.*

Levine, S.; Pastor, P.; Krizhevsky, A.; and Quillen, D. 2016. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*.

Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *ICLR*.

Lu, Y.; Kumar, A.; Zhai, S.; Cheng, Y.; Javidi, T.; and Feris, R. S. 2016. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *CoRR* abs/1611.05377.

Mahmood, A. 2017. Incremental off-policy reinforcement learning algorithms. *Ph.D. UAlberta*.

Mankowitz, D. J.; Zídek, A.; Barreto, A.; Horgan, D.; Hessel, M.; Quan, J.; Oh, J.; van Hasselt, H.; Silver, D.; and Schaul, T. 2018. Unicorn: Continual learning with a universal, off-policy agent. *CoRR* abs/1802.08294.

Mcclelland, J. L.; Mcnaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*.

Mccloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*.

Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A. J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; Kumaran, D.; and Hadsell, R. 2016. Learning to navigate in complex environments. *CoRR* abs/1611.03673.

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-stitch networks for multi-task learning. *CoRR* abs/1604.03539.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; Legg, S.; Mnih, V.; Kavukcuoglu, K.; and Silver, D. 2015. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped DQN. In *NIPS*.

Parisotto, E.; Ba, L. J.; and Salakhutdinov, R. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR* abs/1511.06342.

Precup, D.; Sutton, R. S.; and Singh, S. P. 2000. Eligibility traces for off-policy policy evaluation. In *ICML*.

Ring, M. 1994. Continual learning in reinforcement environments.

Romera-Paredes, B.; Aung, H.; Bianchi-Berthouze, N.; and Pontil, M. 2013. Multilinear multitask learning. In *ICML*.

Rusu, A. A.; Colmenarejo, S. G.; Gülçehre, Ç.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2015. Policy distillation. *CoRR* abs/1511.06295.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *CoRR* abs/1606.04671.

Schmidhuber, J. 1990. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *IJCNN*.

Schmidhuber, J. 2013. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. In *Frontiers in Psychology*.

Schmitt, S.; Hudson, J. J.; Zídek, A.; Osindero, S.; Doersch, C.; Czarnecki, W. M.; Leibo, J. Z.; Küttler, H.; Zisserman, A.; Simonyan, K.; and Eslami, S. M. A. 2018. Kickstarting deep reinforcement learning. *CoRR* abs/1803.03835.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2015. Trust region policy optimization. *CoRR* abs/1502.05477.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *CoRR* abs/1707.06347.

Sharma, S., and Ravindran, B. 2017. Online multi-task learning using active sampling. *CoRR* abs/1702.06053.

Shelhamer, E.; Mahmoudieh, P.; Argus, M.; and Darrell, T. 2016. Loss is its own reward: Self-supervision for reinforcement learning. *CoRR* abs/1612.07307.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR* abs/1712.01815.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Sutton, R. S.; Modayil, J.; Delp, M.; Degris, T.; Pilarski, P. M.; White, A.; and Precup, D. 2011. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS*.

Sutton, R. S.; Mahmood, A. R.; Precup, D.; and van Hasselt, H. 2014. A new q(λ) with interim forward view and Monte Carlo equivalence. In *ICML*.

Teh, Y. W.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distral: Robust multitask reinforcement learning. *CoRR* abs/1707.04175.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? In *NIPS*.

Thrun, S. 2012. *Explanation-based neural network learning: A lifelong learning approach*. Springer.

Todorov, E. 2009. Compositionality of optimal control laws. In *NIPS*. Curran Associates, Inc.

van Hasselt, H.; Guez, A.; Hessel, M.; Mnih, V.; and Silver, D. 2016. Learning values across many orders of magnitude. In *NIPS*.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence*.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, King's College, Cambridge, England.

Williams, R. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learning*.