PERCEPTION-PREDICTION-REACTION AGENTS FOR DEEP REINFORCEMENT LEARNING

Adam Stooke*
Department of Computer Science
University of Berkeley
adam.stooke@berkeley.edu

Valentin Dalibard
DeepMind

Siddhant M. Jayakumar
DeepMind

Wojciech M. Czarnecki
DeepMind

Max Jaderberg
DeepMind

ABSTRACT

Deep reinforcement learning agents with recurrent neural networks have been widely successful in solving partially observable environments. However, these agents can struggle with long-term memory tasks. In this paper, we introduce a novel recurrent agent architecture and associated auxiliary losses which improve learning in such tasks. We employ a temporal hierarchy, using a slow-ticking recurrent core to allow information to flow more easily over long time spans. A fast-ticking core incorporates new observations with the slow core's output to produce the agent's policy. Two other fast-ticking cores have access to only partial information (either long-term or short-term), and produce auxiliary policies which act as priors: an auxiliary loss regularizes all three policies against each other. We present the resulting *Perception-Prediction-Reaction* (PPR) Agent and demonstrate its improved performance over a strong LSTM-agent baseline in DMLab-30, particularly in tasks involving long-term memory. In a series of ablation experiments, we probe the importance of each component of the PPR Agent.

1 INTRODUCTION

In the reinforcement learning (RL) problem, an agent is trained to solve an environment cast as a Markov decision process (MDP), specified as a tuple of states, actions, transition probabilities, and rewards: $(\mathcal{S}, \mathcal{A}, P, r)$. By definition, time is discretized, and the agent must learn which states and actions lead to the best rewards without prior knowledge of P. In many interesting RL problems, however, the agent receives an observation, $x_t = o(s_t) \in \mathcal{X}$, which does not completely specify the state of the MDP at that time step, resulting in *partial observability*. Therefore, for partially observable Markov decision processes (POMDPs) (Astrom, 1965; Kaelbling et al., 1998), a focus of agent design is how to integrate the sequence of historical observations $(x_0, x_1, \ldots, x_t)$ to best approximate the state s_t and produce a policy π_t to maximise future rewards. In deep RL, recurrent neural networks (RNNs) allow integrating observations over time with constant computational complexity (Mnih et al., 2016). Agents based on traditional recurrent networks, *e.g.* LSTMs (Hochreiter & Schmidhuber, 1997), are widely effective, but they sometimes struggle to learn in more complex environments, especially those requiring long-term memory.

In this paper, we introduce a recurrent agent architecture, and associated auxiliary losses (Jaderberg et al., 2016), which aim to improve RL in partially observable environments, particularly those requiring long-term memory. Our method builds upon existing recurrent agents by injecting priors into both the structure of the agent and the optimisation objective of the agent. Specifically, we introduce a slowly ticking recurrent core to augment the standard fast ticking agent core – this allows a pathway for long-term memory storage and eases the backwards flow of gradients over long time spans. In addition, we construct two auxiliary policies, the first of which is required to use

*Work done while at DeepMind, London, UK.

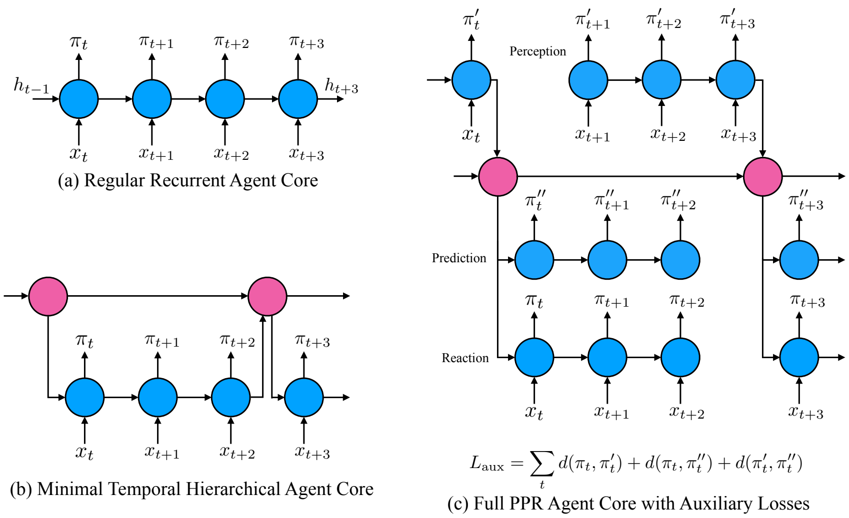


Figure 1: (a) A regular recurrent agent core (Mnih et al., 2016) which can integrate historical experience of observations x_t using an RNN to produce a policy π_t. (b) A minimal temporal hierarchical agent core, featuring fast- and slow-ticking recurrences (Jaderberg et al., 2018). The slow-ticking core skips large portions of time, facilitating BPTT. (c) The PPR agent recurrent structure introduced in this paper, featuring a slow-ticking core and three fast-ticking cores. The *perception* and *prediction* fast-ticking branches have different information hidden relative to the *reaction* fast-ticking core, which has full information and produces the behaviour policy. All fast cores can share the same NN weights. An auxiliary loss L_{aux} encourages the fast-ticking branches to predict the same policy with different information, where d is the symmetrized Kullback-Leibler Divergence.

only current observations without long-term memory (*perception*), and the second which must only use the long-term memory without current observations (*prediction*). These auxiliary policies are trained jointly with the full information policy (*reaction*), with all three policies regularizing each other and shaping the representation of the slow-ticking recurrent core.

We evaluate this agent, dubbed the *Perception-Prediction-Reaction* agent (PPR) on a suite of experiments on 3D, partially observable environments (Beattie et al., 2016), and show consistent improvement compared to strong baselines, in particular on tasks requiring long-term memory. Ablation studies highlight the efficacy of each of the structural priors introduced in this paper. Finally, we apply this agent to the challenging DMLab-30 domain (one agent which must learn across 30 different POMDPs simultaneously), and show that even in this highly varied RL domain, the PPR agent can improve performance.

2 The Perception-Prediction-Reaction Agent

This section introduces the structural and objective priors which constitute the PPR agent. We start with background on recurrent neural network-based agents for reinforcement learning.

Reinforcement Learning and Recurrent Agents. In an MDP, the goal of the RL agent is to find a policy over actions, $\pi(a_t|s_t)$, conditioned on the state, that maximizes the expected discounted sum of future rewards, $\mathbb{E}_{s_t,a_t\sim\pi,P}[\sum_{t=0}^{\infty}\gamma^t r_t]$. The objective remains the same under partial observability, only the agent must additionally estimate the state using incomplete information gleaned from observations, $x_t = o(s_t) \in \mathcal{X}$.

In POMDPs, recurrent agents can improve their internal understanding of the current state by carrying information from past observations, $(x_0, x_1, \ldots, x_{t-1})$, in an internal state, h_{t-1}, to complement the current observation, x_t. The agent updates its internal state by $h_t = f(x_t, h_{t-1})$, and the resulting policy receives conditioning as $\pi(a_t|h_t)$, see Figure 1 (a). Training by backpropagation through time (BPTT) (Webros, 1990; Rumelhart et al., 1988) allows rewards to influence the processing of observations and internal state over earlier time steps. Sophisticated recurrent functions, f, can extend the agent's ability to handle longer (and hence more difficult) sequences. LSTM-based agents have succeeded in a range of partially observable environments, including ones with rich visual observations (Mnih et al., 2016; Espeholt et al., 2018), but many such tasks remain difficult to master or are learned slowly with the traditional architecture.

Minimal Temporally Hierarchical Agent. Temporal hierarchy promises to further improve the processing of long sequences by dividing responsibilities for short- and long-term memory over different recurrent cores, simplifying the roles of each. One conspicuous approach is to employ an additional recurrent unit operating at a rate slower than the MDP – this reduces the number of intermediate computations between distant time steps and allows error gradients to skip backwards through large portions of time. An example incarnation of this concept is in Figure 1 (b): the slow core advances every τ time steps (depicted is $\tau = 3$); during the interim it provides a fixed output to modulate the fast core; the fast core provides summary information to the slow core. As depicted, the recurrence equations could take the following form:

$$h_t^S = \begin{cases} f_S(h_{t-1}^F, h_{t-1}^S) & \text{if } t \bmod \tau = 0 \\ h_{t-1}^S & \text{otherwise} \end{cases} \qquad h_t^F = \begin{cases} f_F(x_t, h_t^S, \emptyset) & \text{if } t \bmod \tau = 0 \\ f_F(x_t, h_t^S, h_{t-1}^F) & \text{otherwise} \end{cases} \tag{1}$$

where the superscripts S and F denote slow and fast cores, respectively, h_t^S, h_t^F are the recurrent states, x_t is the observation, and $\emptyset$ denotes a vector of zeros (*i.e.* the initial recurrent state). The policy could generically depend on the recurrent states, $\pi_t = g(h_t^F, h_t^S)$ (in our case g is an MLP). The internal state of the fast core is periodically reset to $\emptyset$ so as to divide memory responsibilities by time-scale; all information originating prior to $\tau\lfloor t/\tau \rfloor$ must have routed through the slow core. Even so, training this minimal hierarchical agent does not on its own guarantee efficient training long-term memory in a way that improves overall learning relative to the flat agent, see ablations in Figure 3 (b). Indeed, previous examples of temporally hierarchical agents (Vezhnevets et al., 2017; Jaderberg et al., 2018) introduce auxiliary objectives to best make use of this hierarchical structure.

Auxiliary Policy Priors. The PPR Agent is depicted in Figure 1 (c), and we build its description starting from the minimal hierarchical agent. First, we eliminate the possibility of a trivial feed-through connection from fast-slow-fast. Rather than attempt a partial information bottleneck, we prevent the fast core (now, *reaction*) which receives input from the slow core, from passing any output back to it. We introduce another fast ticking core (*perception*) which feeds its output into the slow core but does not take input from it. Resetting the fast internal states at the interval τ forms branches in the graph. The *reaction* branch produces the agent's policy by integrating new observations together with the slow core's output. The slow core assumes a central role in representing information originating prior to $\tau\lfloor t/\tau \rfloor$, as it receives periodic, short-term summaries from the *perception* branch, which also integrates observations. This forms a Perception-Reaction Agent *without auxiliary losses*, a baseline in our ablation experiments.

The final architectural element of the PPR Agent is an additional fast recurrent core (*prediction*). It branches simultaneously to *reaction* and receives the slow core's output and possibly partial observations p_t (*e.g.* previous actions). This creates an information asymmetry (as in Galashov et al. (2018)) against *perception*, which lacks long-term memory, and the fully-informed *reaction*. We leverage the asymmetry to simultaneously (a) shape the representation of the slow-ticking core, (b) maximize information extraction from observations, and (c) balance the importance of both in the policy. We do so by drawing auxiliary policies, π' and π'' from the *perception* and *prediction* branches, respectively, to form the auxiliary loss:

$$\mathcal{L}_{aux} = \sum_t d(\pi_t, \pi_t') + d(\pi_t, \pi_t'') + d(\pi_t', \pi_t'') \tag{2}$$

where d is a statistical distance – we use the symmetrized Kullback-Leibler Divergence. All three branches are regularized against each other; $\mathcal{L}_{aux}$ encourages their policies to agree to the extent possible despite differences in access to information. Rather than apply a loss directly on the recurrent state, which may assume somewhat arbitrary values, the policy distribution space offers grounding in the environment.

The recurrence update and policy equations of the PPR Agent are summarized in Table 1. Loosely speaking, *reaction* is a short-term sensory-motor loop, *perception* a sensory loop, *prediction* a motor loop, and the slow core a long-term memory loop, all of which are decoupled in forward operation. The auxiliary divergence losses can be seen as imposing two priors on the fully informed *reaction*

Table 1: Recurrence and policy equations of the PPR Agent.

Core		Recurrence Equation		Policy
		if $t \bmod \tau = 0$:	otherwise:	
Slow	$h_t^S =$	$\{f_S(h'_t, h^S_{t-1};\, \theta^S),$	$h^S_{t-1}\}$	
Reaction	$h_t =$	$\{f(x_t, h^S_t, \emptyset;\, \theta),$	$f(x_t, h^S_t, h_{t-1};\, \theta)\}$	$\pi_t = g(h_t;\, \phi)$
Prediction	$h''_t =$	$\{f(p_t, h^S_t, \emptyset;\, \theta),$	$f(p_t, h^S_t, h''_{t-1};\, \theta)\}$	$\pi''_t = g(h''_t;\, \phi'')$
		if $t \bmod \tau = 1$:	otherwise:	
Perception	$h'_t =$	$\{f(x_t, \emptyset, \emptyset;\, \theta),$	$f(x_t, \emptyset, h'_{t-1};\, \theta)\}$	$\pi'_t = g(h'_t;\, \phi')$

branch – that the policy should be expressible from only recent observations (*perception*) and from only long-term memory (*prediction*).

Implementation. The partial observations p_t may be chosen somewhat arbitrarily, but might require special care to enable useful regularization. For visual environments, the recurrent A3C Agent (Mnih et al., 2016) suggests a convenient delineation: the partial observation consists of the previous action and reward in the environment, $p_t = (a_{t-1}, r_{t-1})$, which supplement the screen image (processed through a CNN) to make the full observation.

In practice, the PPR architecture is implemented as a self-contained recurrent neural network core, and training only requires an additional loss term, allowing the agent to be easily incorporated in most existing deep RL frameworks. In our experiments we found it possible to use the same network weights for the recurrences all branches, as reflected in Table 1.

3 Related Work

Recurrent networks with multiple time-scales have appeared in numerous forms for supervised learning on sequences. Clockwork RNNs (Koutník et al., 2014) and Phased LSTMs (Neil et al., 2016), for example, mainly address the propagation of long-term dependencies by assigning different operating periods within one layer. Hierarchical Multiscale RNNs (El Hihi & Bengio, 1996; Schmidhuber, 1992; Chung et al., 2016) instead introduce operations allowing layers in a stacked RNN to influence temporal behavior of higher layers, for a learnable hierarchy.

In reinforcement learning, our work relates to the FTW agent of (Jaderberg et al., 2018). The FTW agent features a slow ticking and fast ticking core, similar to what is depicted in Figure 1 (b), and includes a prior to regularize the hidden state distribution between the slow and fast cores. Our work also builds on recent approaches to learning priors with information asymmetry for RL (Galashov et al., 2018; Teh et al., 2017). Other work utilises memory modules in agents for better learning through time (Miconi et al., 2018; Hung et al., 2018), and a wealth of previous work exists on more explicit hierarchical RL which often exploits temporal priors (Sutton et al., 1999; Heess et al., 2016; Vezhnevets et al., 2017).

4 Experiments

We conducted experiments seeking to answer the questions: (a) does the PPR hierarchy lead to improved (or worsened) learning relative to flat architectures, and if so, (b) which kind of tasks is it most effective at accelerating, and (c) what are the effects of different components of the architecture. We report here experiments on levels within the DMLab-30 suite (Beattie et al., 2016). It includes a collection of visually rich, 3D environments for a point-body agent with a discrete action space. The range of tasks vary in character from memory-, navigation-, and reactive agility-based ones. Language-based tasks are also included. Agent training details can be found in the Appendix.

DMLab Individual Levels. We tested PPR agents on 12 DMLab levels. For each level, we trained a PPR and a baseline agent for 2 billion environment frames. Figure 2 highlights some of the re-

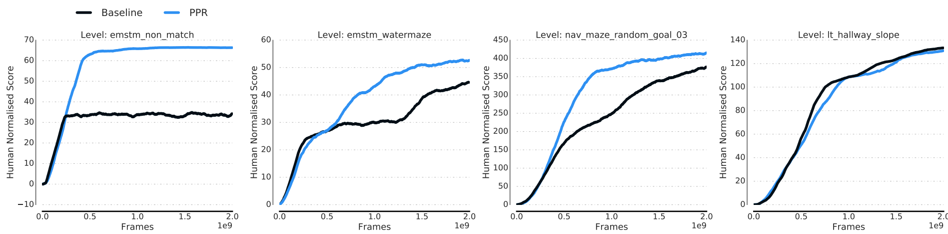


Figure 2: Learning curves of the PPR agent (blue) compared to the baseline recurrent agent (black) (Espeholt et al., 2018) on four representative DMLab tasks. The PPR agent can achieve higher scores and faster learning on long-term memory tasks (*e.g.* `emstm_non_match`, `emstm_watermaze`, `nav_maze_random_goal_03`), while not degrading in performance on more reactive tasks, such as lasertag (`lt_hallway_slope`). More levels can be found in the Appendix.

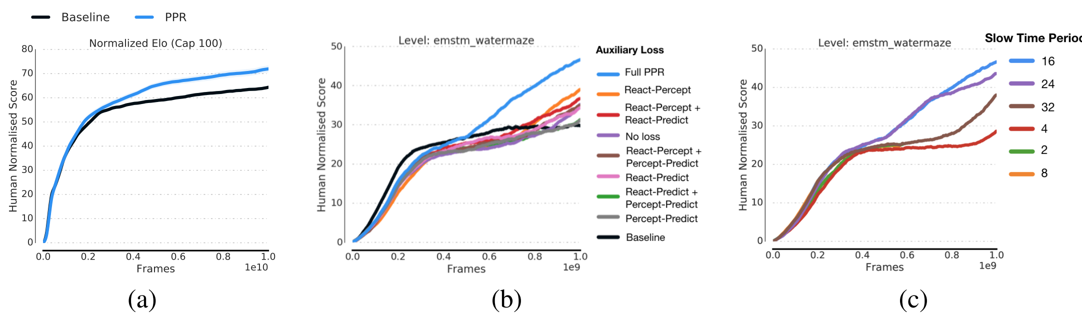


Figure 3: (a) Learning curves on the DMLab-30 task domain with the PPR agent (blue) and recurrent agent baseline (black) (Espeholt et al., 2018). The PPR agent consistently outperforms the Impala (Espeholt et al., 2018) on this challenging domain. (b) Ablation study on losses. (c) Ablation study on time periods.

sults. Compared to the baseline, the PPR agent showed significantly faster learning and significantly higher return on tasks by exhibiting long-term memory, and did not degrade performance on more reactive tasks. The full results are available in the Appendix.

DMLab-30. We next tested the PPR agent on simultaneously learning the entire DMLab-30 suite, to see whether benefits could extend across the range of tasks while under the same hyperparameters. Indeed, the PPR agent outperformed the flat LSTM baseline, achieving an average capped human-normalized ELO across levels of 72.0 mean (across 8 independent runs), compared to 64.3% with the baseline (Espeholt et al., 2018), Figure 3. Per-level scores from these learning runs can be found in the Appendix. This difference, while modest, is difficult to achieve compared to the highly tuned baseline agent and represents a significant improvement.

Ablations. To determine the effects of individual components of the PPR agent, including the auxiliary losses, we returned to experimenting on individual levels. Figure 3 (b) shows different combinations of the three PPR auxiliary loss terms activated. The predictive branch, which is only trained via the auxiliary loss, is revealed to be crucial to the learning gains. Although using two of the three losses can be effective, in general we measured more consistent results with all three active. Figure 3 (c) shows the results from different values for the slow ticking core interval, τ. In this task, a wide range of values worked well (shown: 4, 8, 16, 24, 32), with $\tau \geq 16$ working best.

Flat, Predictive Agent. Given the benefit the *prediction* branch brings to the PPR Agent, it is worth studying the flat baseline agent Figure 1 (a) with a *prediction* branch and auxiliary regularisation loss. We trained such an agent on individual DMLab levels. We found that in navigation levels, for example, it was possible to run the policy drawn from the *prediction* branch (*i.e.* short-horizon open-loop controls) and achieve scores similar to the baseline agent (see Appendix for details). Despite succeeding at policy prediction, we discovered no such agent to learn faster or better than the baseline. Evidently, the full PPR Agent is needed to accelerate learning.

5 Conclusion

In this paper we introduced a new agent to deal with partially observable environments, the PPR agent, which incorporates a temporally hierarchical recurrent structure, as well as imposing priors on the behaviour policy to be both predictable from long-term memory only, and from current observations only. This agent was evaluated on a challenging set of 3D RL problems, and showed improved performance, in particular on tasks involving long-term memory. We hope to build upon these ideas to further improve deep RL in partially observable environments in future work.

References

Karl J Astrom. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965.

Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. URL `http://arxiv.org/abs/1812.03079`.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016. URL `http://arxiv.org/abs/1612.03801`.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pp. 493–499, 1996.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR*, abs/1802.01561, 2018. URL `http://arxiv.org/abs/1802.01561`.

Alexandre Galashov, Siddhant M Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized rl. 2018.

Nicolas Heess, Gregory Wayne, Yuval Tassa, Timothy P. Lillicrap, Martin A. Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *CoRR*, abs/1610.05182, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *arXiv preprint arXiv:1810.06721*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017. URL `http://arxiv.org/abs/1711.09846`.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, May 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X. URL `http://dx.doi.org/10.1016/S0004-3702(98)00023-X`.

Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A clockwork rnn. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1863–II–1871. JMLR.org, 2014. URL `http://dl.acm.org/citation.cfm?id=3044805.3045100`.

Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic neural networks with backpropagation. *CoRR*, abs/1804.02464, 2018. URL `http://arxiv.org/abs/1804.02464`.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3889–3897, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL `http://dl.acm.org/citation.cfm?id=3157382.3157532`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1988.

Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

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*, pp. 4496–4506, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pp. 3540–3549, 2017.

P J Webros. Backpropagation through time: What it does and how to do it. 1990.

Appendix

A Agent and Training Details

In our experiments, we used LSTM recurrent cores with hidden size 256 and shared weights among the three fast branches. We trained our agents and baseline using the V-Trace algorithm (Espeholt et al., 2018) on trajectory segments 100 agent time steps long, using action repeat 4. We introduced extra hyperparemeters for the auxiliary loss weightings, one for each branch-pair, and included these in the set of hyperparameters tuned by Population-based training (PBT) (Jaderberg et al., 2017). Each experiment used a population size of 24. For visual levels, our convolution network was an 15-layer residual network. We typically fixed the slow core interval, τ, to 16. Our baselines all used an identical architecture with a flat LSTM core for memory as in Espeholt et al. (2018).

Earlier experiments in reactive tasks, such as laser tag, did sometimes result in degraded performance due to a learning mode in which the policy became less dynamic, to be easier to predict. One effective way to mitigate this phenomenon is to apply $\mathcal{L}_{aux}$ to only a (random) subset of training batches (found concurrently in (Bansal et al., 2018)), leaving the policy more free to pursue rewards. After experimenting with different ways to achieve this, we found rescaling $\mathcal{L}_{aux}$ by a factor randomly sampled from $U(0, 1)$ each batch worked best. This was used in all our experiments.

B Additional Learning Curves

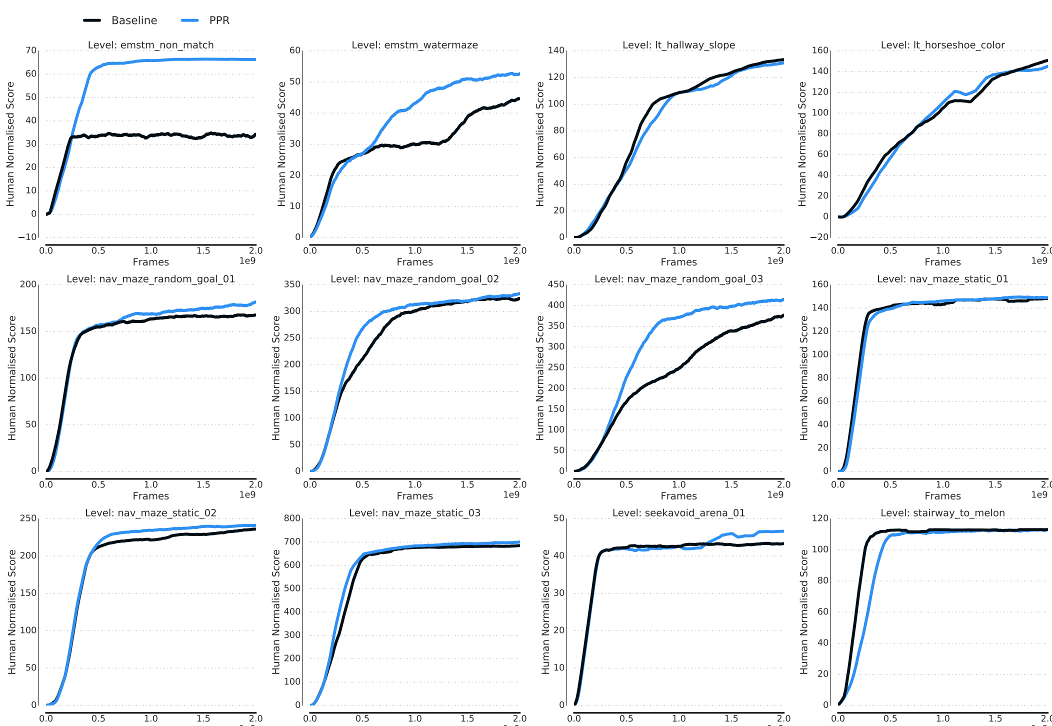


Figure 4: Learning curves of the PPR agent (blue) compared to the baseline recurrent agent (black) (Espeholt et al., 2018) on various individual DMLab tasks.

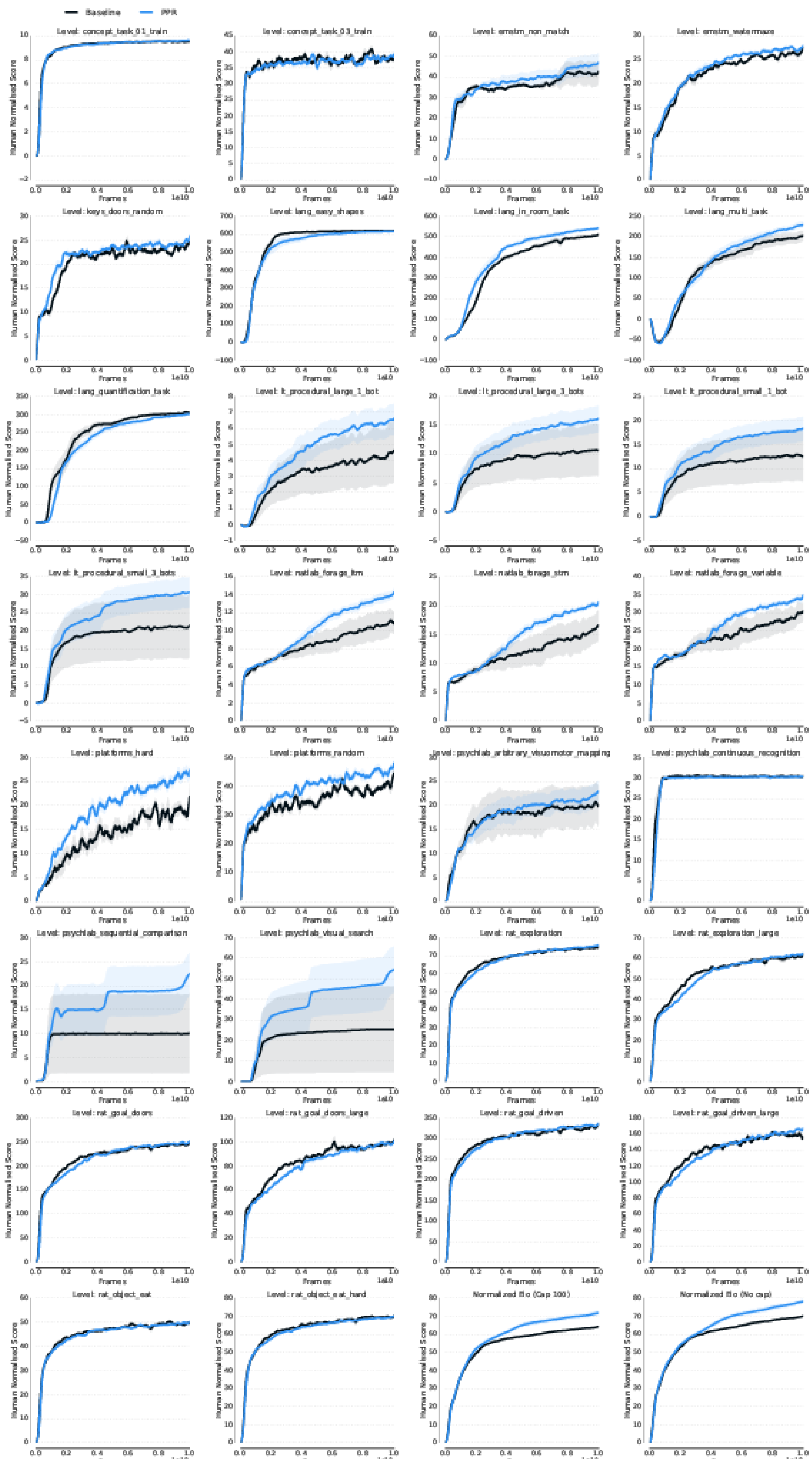
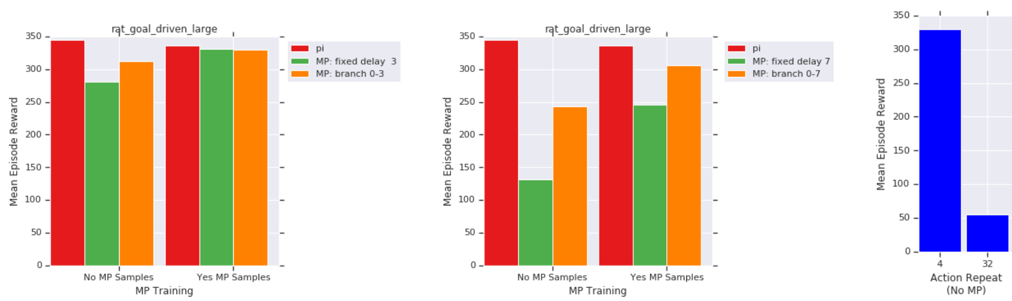


Figure 5: Learning curves on the DMLab-30 task domain with the PPR agent (blue) and recurrent agent baseline (black), separated by level. Shaded area shows the mean standard error. The PPR agent consistently outperforms the baseline Espeholt et al. (2018) on this challenging domain.

C FLAT, PREDICTIVE AGENT

We trained a flat, *prediction* agent on the DMLab navigation level `rat_goal_driven_large` using various different schemes drawing from the *prediction* policy during training ("Yes MP Samples") or not ("No MP Samples"). We evaluated final agent performance under different execution schemes, as well. Figure 6 (a) shows agents executing the *prediction* policy up to 3 time steps beyond the last injection of new observational information from the main LSTM; agent score is barely diminished. Figure 6 (b) shows the same *trained* agents evaluated at longer delays, up to 7 steps. The high performance of these agents proves successful learning of the *prediction* auxiliary policy over similar time scales used in our studies of the PPR Agent, yet in no case did we observe it to improve on the base agent's learning. This level most closely corresponds to `nav_maze_random_goal_03`, in which PPR showed improvements, see Figure 2. Figure 6 (c) shows a baseline agent trained with the standard frame-skip (4), and one with longer frame-skip (32), with correspondence to the delay in 6 (b), for comparison. The performance of the longer frame-skip agent suffered due to larger granularity in environment interaction frequency, despite having the same refresh rate/delay for incorporating new observations into the policy.



(a) (b) (c)

Figure 6: Final evaluation scores for trained flat, *prediction* agent in `rat_goal_driven_large`. Various schemes used for drawing the behavior policy from the *prediction* auxiliary policy, many of which perform similarly to the baseline (reactive) agent. (a) Agents executing *prediction* policy up to 3 time steps without new observations. (b) The same trained agents, but evaluated up to 7 time steps without new observations. (c) Long action-repeat trained agent.