Hypernetworks in Meta-Reinforcement Learning
Supplementary Materials

Jacob Beck
Department of Computer Science
University of Oxford, United Kingdom
jacob_beck@alumni.brown.edu

Matthew Jackson
Department of Engineering Science
University of Oxford, United Kingdom
jackson@robots.ox.ac.uk

Risto Vuorio
Department of Computer Science
University of Oxford, United Kingdom
risto.vuorio@keble.ox.ac.uk

Shimon Whiteson
Department of Computer Science
University of Oxford, United Kingdom
shimon.whiteson@cs.ox.ac.uk

1 Benchmark Details

We evaluate on grid-world [1], MuJoCo [2], and Meta-World [3]. For Meta-World[1], we use version two of ML10 and version one of Pick-Place (ML1). We use gridworld and MuJoCo environments from the reference VariBAD implementation[2]. The number of episodes per meta-episode are the same as in this implementation. Returns reported are summed across episodes in the meta-episode.

2 Model Implementation

2.1 Network Architecture

For all experiments, we use a three-layer MLP network to represent the base network, either learning the parameters directly or generating them with a hypernetwork. Both of these networks are conditioned on a task embedding of size 10 in all experiments. A range of architecture sizes were explored on Cheetah-Dir and Walker MuJoCo tasks, which are detailed in Table 1.

Table 1: Policy (base) network architectures.

Size	**Layer widths**
XS	(64, 64, 32)
S	(128, 64, 64)
M	(128, 128, 64)
L	(256, 128, 128)
XL (*Default*)	(256, 256, 128)
XXL	(1024, 512, 512)

2.2 Actor and Critic Networks

In the standard architecture, we use separate actor and critic networks. In the hypernetwork model, there are separate heads on the hypernetwork for the actor and critic parameters. Since all reported results use linear hypernetworks, this is equivalent to using separate hypernetworks for the actor and critic, conditioned on the same task embedding.

[1] https://github.com/rlworkgroup/metaworld
[2] https://github.com/lmzintgraf/varibad

2.3 Choice of Base Network Initialization

Our proposed methods, Bias-HyperInit and Weight-HyperInit, require a given initialization method for the base network, which we denote f. We chose normc intialization [4] for f, as it is the default initialization in the reference VariBAD implementation. Note that, as in VariBAD, the state and task encoders use Kaiming initialization with a uniform distribution [5] (or orthogonal initialization in the RNN), and the head of the critic (produced by the hypernetwork) also uses Kaiming. Additionally, the head of the actor has a gain of 0.01 for categorical distributions (i.e. in gridworld), and 1 for continuous distributions (as is standard for linear layers), as is default in VariBAD. However, as this default was noticed after experiments, the actor and critic heads of both HyperInit methods have a gain of $\sqrt{2}$, which is the default for ReLU activations.

3 Hyperparameter Tuning

Hyperparameters are identical to those in the reference VariBAD implementation, other than the learning rate and network architecture.

For Cheetah-Dir and Walker, the models were tuned over the following set of learning rates on three seeds: $\{3\text{e}{-3}, 1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}\}$. Learning rates of $1\text{e}{-3}$ and $1\text{e}{-4}$ yielded the best performance for standard and hypernetwork models respectively, over both domains. For the remainder of the MuJoCo tasks and grid-world, the learning rate was tuned over a narrower range of learning rates. These ranges were $\{3\text{e}{-3}, 1\text{e}{-3}, 3\text{e}{-4}\}$ for the standard model and $\{3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}\}$ for the hypernetwork. Again, we found that $1\text{e}{-3}$ and $1\text{e}{-4}$ yielded the best performance across domains for the standard and hypernetwork models respectively. Given this, we used these two tuned learning rates for further evaluation on Pick-Place. On ML10, we again evaluate over the full range: $\{3\text{e}{-3}, 1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}\}$.

The network sizes were tuned on Cheetah-Dir and Walker as well. The standard and hypernetwork models were turned over all base network sizes in Table 1, with the exception of the XXL base model, which was included only to demonstrate the performance of the standard model with an equal number of model parameters as the XL hypernetwork architecture. For both hypernetworks and the standard network, the XL size achieved the highest return (or not significantly different from the best) on both tasks. We then used XL for the rest of the MuJoCo tasks and for Pick-Place and ML10. Gridworld was tuned separately. For gridworld, we evaluated over M, L, and XL for the standard architecture and XS, S, and M for the hypernetwork architecture. All models solved the domain and performed similarly. The results reported on gridworld are therefore from the smallest evaluated architecture sizes, M and XS respectively.

Additionally, on Cheetah-Dir and Walker we evaluated two-layer (rather than linear) hypernetworks, with a hidden layer of width 16 or 32. (Note that the input size, i.e. the output from the task encoder, was kept constant at 10.) In both cases, for L and XL base network sizes, adding an additional layer decreased or did not affect return.

4 Learning Curves

Figure 1 presents learning curves for all remaining domains and models. These results are summarized in the main paper, but are reported here for completeness. Note that reported metrics are computed over the last 1% of data for all curves, other than ML10, which uses the last 10% of data.

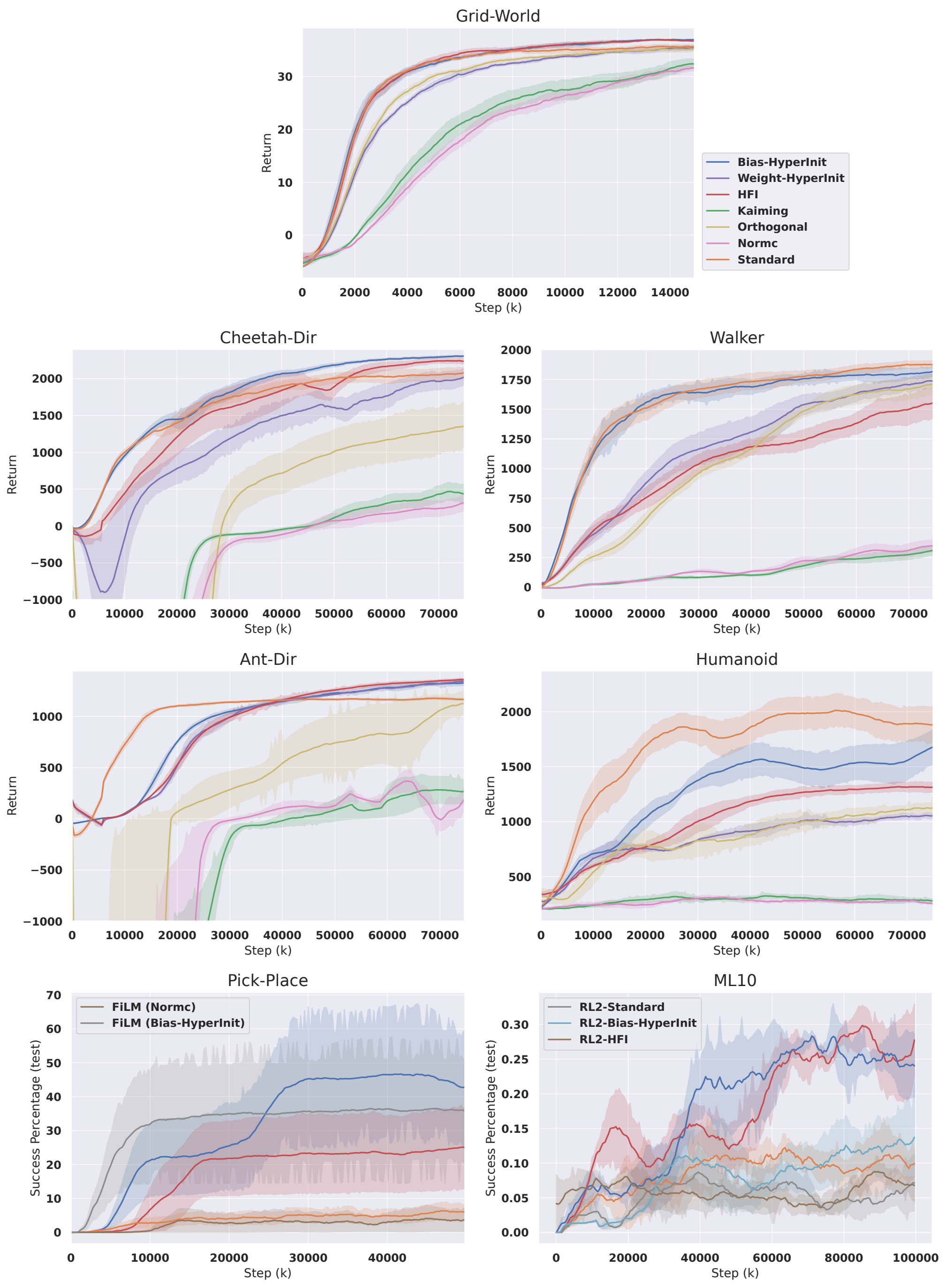

Figure 1: Learning curves for all remaining domains. (68% confidence interval for all plots.)

5 Meta-World Training Performance

The Pick-Place and ML10 benchmarks have distinct train and test tasks. While we present and discuss test performance in the main report, Table 2 details the train performance of the standard architecture and hypernetwork on these benchmarks. For Pick-Place, we observe similar train and test performance. However, we found that the ML10 train performance differed from test performance.

At meta-train time, the standard architecture and hypernetwork with Bias-HyperInit achieve nearly identical success rates on ML10. However, hypernetworks with Bias-HyperInit and HFI achieve

Table 2: Meta-train success percentage on Meta-World benchmarks.

Method	**Pick-Place**	**ML10**
Standard	4.4 ± 2.1	33.8 ± 3.0
HFI	$\mathbf{24.3 \pm 14.9}$	36.0 ± 2.3
Bias-HyperInit	$\mathbf{45.3 \pm 17.3}$	32.0 ± 0.4

significantly greater success rates on test tasks (see main report), suggesting that hyernetworks have improved generalization compared to standard models, which overfit to the train tasks.

6 Application to FiLM

When applied to FiLM [6], the Bias-HyperInit method becomes:

1. Sample parameters $\phi \sim f$, set $W = 0$, and set b to produce the bias parameters in ϕ (as in Bias-HyperInit),
2. Set the weights of base network to those in ϕ (as required by FiLM),
3. Set the remaining elements of b to 1, thereby leaving the weight distribution unchanged.

Note that Weight-HyperInit can also be adapted such that W produces distinct bias parameters for each task and scalars of 1 for all tasks, but we do not test this since Bias-HyperInit outperforms Weight-HyperInit.

References

[1] L. Zintgraf, S. Schulze, C. Lu, L. Feng, M. Igl, K. Shiarlis, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021. (cited on p. 1)

[2] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. (cited on p. 1)

[3] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2020. (cited on p. 1)

[4] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines, 2017. (cited on p. 2)

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. (cited on p. 2)

[6] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. *AAAI*, 2018. (cited on p. 4)