Hypernetworks in Meta-Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Training a reinforcement learning (RL) agent on a real-world robotics task remains generally impractical due to sample inefficiency. Multi-task RL and meta-RL aim to improve sample efficiency by generalizing over a distribution of related tasks. However, doing so is difficult in practice: In multi-task RL, state of the art methods often fail to outperform a degenerate solution that simply learns each task separately. Hypernetworks are a promising path forward since they replicate the separate policies of the degenerate solution while also allowing for generalization across tasks, and are applicable to meta-RL. However, evidence from supervised learning suggests hypernetwork performance is highly sensitive to the initialization. In this paper, we 1) show that hypernetwork initialization is also a critical factor in meta-RL, and that naive initializations yield poor performance; 2) propose a novel hypernetwork initialization scheme that matches or exceeds the performance of a state-of-the-art approach proposed for supervised settings, as well as being simpler and more general; and 3) use this method to show that hypernetworks can improve performance in meta-RL by evaluating on multiple simulated robotics benchmarks.

Keywords: Meta-Learning, Reinforcement, Hypernetwork

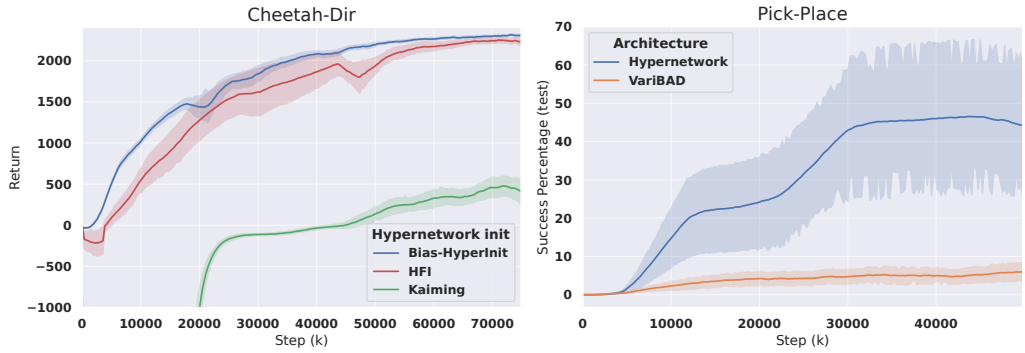
1 Introduction

Deep reinforcement learning (RL) has helped solve previously intractable problems but still remains highly sample inefficient. This sample inefficiency makes it impractical, particularly in settings where data collection happens in the real world. For example, a robot's actions have the potential to inflict damage on both itself and its surroundings. Multi-task RL and meta-RL aim to improve sample efficiency on novel tasks by generalizing over a distribution of related tasks. However, such generalization has proven difficult in practice. In fact, multi-task RL methods often fail to outperform a degenerate solution that simply trains a separate policy for each task [1].

One promising way to improve generalization is with a *hypernetwork*, a neural network that produces the parameters for another network, called the *base network* [2]. In multi-task RL, using a hypernetwork that conditions on the task ID to generate task-specific parameters can replicate the separate policies of the degenerate solution, while also allowing generalization across tasks. Furthermore, unlike the degenerate solution, hypernetworks can also be applied to meta-RL, where task IDs are not provided and test tasks may be novel, by conditioning them on the output of a task encoder.

However, hypernetworks come with their own challenges. Since hypernetworks generate base network parameters, the initialization of parameters in the hypernetwork determines the initialization of the base network it produces. Evidence suggests hypernetwork performance is highly sensitive to the initialization scheme in supervised learning [3]. However, to our knowledge this question has not been considered in meta-RL. In this paper, we show that hypernetwork initialization is also a critical factor in meta-RL, and that naive initializations yield poor performance.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.



(a) Hypernetwork initialization methods

(b) Architectures

Figure 1: Naive initializations such as Kaiming [4] fail for hypernetworks, whereas our proposed Bias-HyperInit does not and matches the state of the art, HFI [3] (claims 1, 2). Adding hypernetworks with the proposed Bias-HyperInit significantly improves the state-of-the-art meta-RL method, VariBAD [5] (claim 3).

Furthermore, we propose two novel initialization schemes: *Bias-HyperInit* and *Weight-HyperInit*. Both produce strong results, with the former matching or exceeding the performance of the state-of-the-art hypernetwork initialization method designed for supervised learning [3]. Moreover, both proposed methods are simpler and more general than this existing method, in that they may be applied to arbitrary base network architectures and target base network initializations without additional derivation. Using Bias-HyperInit, we present results that substantially improve the a state-of-the-art method on a range of meta-RL benchmarks.

Applying hypernetworks to meta-RL, we make the following contributions (examples in Figure 1):

1. We empirically demonstrate that initialization is a critical factor in the performance of hypernetworks in meta-RL, and that naive initializations fail to learn reliably;
2. We propose a novel hypernetwork initialization scheme that matches or exceeds the performance of a state-of-the-art approach proposed for supervised settings, as well as being simpler and more general; and
3. We use this method to show that hypernetworks can improve a state-of-the-art method on a range of meta-RL benchmarks (grid-world [5], MuJoCo [6], and Meta-World [1]).

2 Related Work

Meta-RL. Despite the advantages of hypernetworks [2], they remain relatively unexplored in meta-RL. We use hypernetworks to arbitrarily update a policy's parameters at every time-step, whereas all prior work we are aware of restrict this procedure in some way. Many procedures in few-shot meta-RL build off of MAML [7] to adapt the parameters of a policy network using a policy gradient [7, 8, 9]. Such methods require the estimation of a policy gradient, which reduces sample-efficiency when faster adaptation is possible, as in our benchmarks [5]. Most meta-learning procedures capable of zero-shot adaption using an RNN (or convolutions) that can represent an arbitrary update function [5, 10, 11]. These methods generally update a set of activations on which a fixed policy is then conditioned, whereas hypernetworks update all policy parameters. We include a state-of-the-art method from this class in our evaluations [5]. There are also unsupervised methods in zero-shot meta-RL for weight updates [12, 13] but none can produce a fully general learning procedure since they make use of local and unsupervised heuristics. Sarafian et al. [14] use hypernetworks in the context of meta-RL, but the policy network, not the hypernetwork, is conditioned on the RNN used for adaptation, preventing the hypernetwork from representing a general learning procedure. Finally, FLAP [15] learns to infer a set of weights trained in the multi-task setting; however since the adaptation procedure is not trained on a meta-RL objective, it is constrained. For example,

FLAP cannot learn to explore to reduce uncertainty. Finally, Xian et al. [16] use hypernetworks to predict model dynamics then use model predictive control. However, this model still requires planning to make use of an uncertain model, whereas model-free RL learns a policy that explores optimally in order to attain data for adaptation. Using a general procedure trained to arbitrarily modify the weights of a model-free policy has never been tried in RL, to the best of our knowledge.

Hypernetworks. Hypernetworks, or similar architectures, have been used in supervised learning (SL), multi-task RL, and meta-SL. Hypernetworks have been used in the supervised learning literature for sequence modelling [2], as well as in continual learning and image classification [3], where it was shown that the hypernetwork initialization scheme was crucial for performance. Similar models have also been used in multi-task RL and meta-SL, but not meta-RL. For instance, in multi-task RL, Yu et al. [17] use a network conditioned on a task encoding to produce the weights and biases for every other layer in another network conditioned on state. In meta-SL, there have also been attempts to use one network to adapt weights of another, both as a general function of the dataset [18, 19, 20], conditioned on an embedding adapted by gradient descent [21], and by adding deltas in a way framed as learning to optimize [22, 23]. The abundance of representations in meta-SL suggest there is a similarly large space of representation-based methods to explore in meta-RL. Our work – getting hypernetworks to work in practice for meta-RL – can be seen as a first step towards applying all of these methods in meta-RL.

3 Background

3.1 Problem Setting

An RL task is formalized as a Markov Decision Processes (MDP). We define an MDP as a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. At time-step t, the agent inhabits a state, $s_t \in \mathcal{S}$, observable by the agent. The agent takes an action $a_t \in \mathcal{A}$. The MDP then transitions to state $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t) \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$, and the agent receives reward $r_t = \mathcal{R}(s_t, a_t) \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ upon entering s_{t+1}. Given a discount factor, $\gamma \in [0, 1)$, the agent acts to maximize the expected future discounted reward, $R(\tau) = \sum_{r_t \in \tau} \gamma^t r_t$, where τ is the agent's trajectory over an episode in the MDP. To maximize this return, the action takes actions sampled from a learned policy, $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$.

Meta-RL algorithms learn an RL algorithm, i.e., a mapping from the data sampled from a single MDP, $\mathcal{M} \sim p(\mathcal{M})$, to a policy. Since an RL algorithm generally needs multiple episodes of interaction to produce a reasonable policy, the algorithm conditions on τ, which is the entire sequence of states, actions, and rewards within $\mathcal{M}$. As in the RL setting, this sequence up to time-step t forms a trajectory $\tau_t \in (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^t$. Here, however, τ may span multiple episodes, and so we use the same symbol, but refer to it as a *meta-episode*. The policy is then a meta-episode dependent policy, $\pi_\theta(a|s, \tau)$, parameterized by the *meta-parameters*, θ.

We define the objective in meta-RL as finding meta-parameters θ that maximize the sum of the returns in the meta-episode across a distribution of tasks (MDPs):

$$\arg\max_\theta \mathbb{E}_{\mathcal{M} \sim p(\mathcal{M})} \left[\mathbb{E}_\tau \left[R(\tau) \middle| \pi_\theta(\cdot), \mathcal{M} \right] \right] \tag{1}$$

3.2 Policy Architecture

We consider meta-RL agents capable of adaptation at every time-step, and adaptation within one episode is required to solve some of our benchmarks. In such methods [5, 10, 11], the history is generally summarized by a function, g, into an embedding that represents relevant task information. We write this embedding as $e = g(\tau)$, and call g the task encoder. The policy, represented as a multi-layer perceptron, then conditions on this task embedding as an input, instead of on the history directly, which we write as $\pi_\theta(a|s, e)$. We call this the *standard architecture*, shown in Figure 2.

In this paper, we build off of VariBAD [5], which can be seen as an instance of the standard architecture where the task encoder is the mean and variance from a recurrent variational auto-encoder

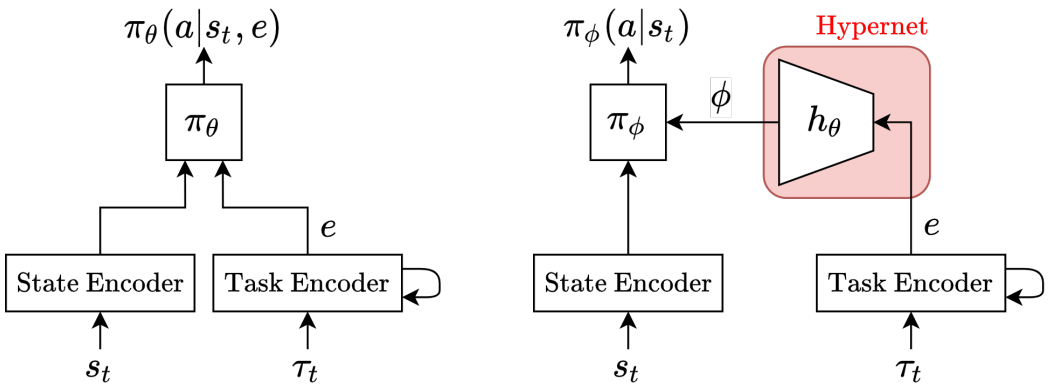


Figure 2: A standard architecture (left) and hypernetwork model (right).

(VAE) [24] trained using a self-supervised loss. The distribution over latent parameters is a useful representation of the agent's belief, i.e., posterior distribution, over what MDP it occupies. See Zintgraf et al. [5] for details.

3.3 Hypernetwork Initialization

Chang et al. [3] show that applying existing initialization methods for neural networks to hypernetworks produces unstable base network initializations with exploding or vanishing activations. Furthermore, they empirically demonstrate a reduction in training stability for Kaiming initialization [4]. We corroborate this failure for Kaiming initialization on meta-RL, as well as for Orthogonal initialization [25] and Normc initialization [26], which we collectively refer to as *default initializations.*

As a solution, Chang et al. [3] propose the first initialization designed for hypernetworks and show it to be effective in supervised learning. Their approach is based on Kaiming initialization, which samples network parameters such that the activations of the network at each layer maintain the same variance as in the previous layer. Chang et al. [3] extend this variance analysis to hypernetworks [2]. They propose two methods: Hyperfan-in (HFI) sets the variance of the initial parameter distribution of the hypernetwork to maintain constant variance of activations in the base network in the forward pass, and hyperfan-out (HFO) does the same for the backward pass. Since these are equally competitive, and produce state-of-the-art results, we include HFI as a baseline for comparison. However, this variance analysis is involved and requires modification for specific use cases depending on the activation function and whether or not the network produces weights or biases in the base network. This motivates the need for a simpler and more general initialization method, which we propose in this paper.

4 Methods

In this section we introduce our proposed architecture using hypernetworks and our proposed hypernetwork initialization. At a high level, the hypernetwork conditions on a task representation to generate all of the parameters for a base policy. The initialization provides a simple and general way to sample parameters for this hypernetwork at the start of training.

4.1 Policy Architecture

We propose the use of hypernetworks in meta-RL, instead of the standard architecture described earlier. In this setting, we use a hypernetwork to arbitrarily adapt the parameters of the base policy. We still use a task encoder, $e = g(\tau)$, but instead of conditioning a policy on these activations, we use a hypernetwork, h_θ, to generate policy parameters, $\phi = h_\theta(e)$. These parameters, i.e., weights

and biases, are then used directly for the base network, which we write: $\pi_\phi(a|s)$. This is also depicted in Figure 2.

In this case, the hypernetwork can arbitrarily adapt all the the parameters of π based on history. In comparison, in the standard architecture, the shared fixed parameters of the base network (θ) are still required to generalize between all of the tasks. Since training separate policies for each task often performs better than state-of-the-art methods for generalizing across all tasks [1], this motivates the ability to produce base policies with no or few shared parameters. Hypernetworks allow for shared parameters when possible, but also provide the ability to have no shared parameters in the base policy when diverse policies are necessary and little generalization between tasks is possible.

In fact, hypernetworks are capable of replicating the training of separate policies for each task under certain conditions. To see this, consider the case where the hypernetwork is linear and has no bias. Then, the hypernetwork consists only of a weight matrix, W. If this hypernetwork conditions on a one-hot task ID for task i: $e = \mathbb{1}_i$, then the parameters selected by this hypernetwork, ϕ are a separate set of parameters for each task: $\phi^i = h(e) = W\mathbb{1}_i$. In other words, training individual networks for each task is equivalent to training a hypernetwork when that hypernetwork is: 1) linear, 2) has no a bias, and 3) is conditioned on a one-hot task ID.

However, we can relax these assumptions and still retain the ability to produce distinct parameters, while also enabling generalization. If we add a bias, we reintroduce shared parameters in the hypernetwork, but they can still produce separate base networks for each task when little transfer between the policies is required. If we relax the one-hot assumption, the network is still capable of producing a one-hot encoding when the tasks are discrete and the task embedding is sufficiently large. Relaxing these restrictions allows for both generalization and the application of hypernetworks to meta-RL.

4.2 Hypernetwork Initialization

Default initialization methods fail for hypernetworks. However, given that hypernetworks generalize training separate networks for each task, it must be possible to initialize them as if each of the corresponding base networks were initialized independently, from some given initialization function, f, known to train reliably. Using this insight, we propose and evaluate two initialization schemes for the hypernetwork. We propose one where (under some assumptions) each base network is independently initialized from f. We also propose one where (under no assumptions) all base networks share an initialization sampled from f.

Our first method is *Weight-HyperInit*. Let W and b be the weights and bias in the last layer of our hypernetwork, $h(e)$, respectively. We define this weight-only initialization as follows:

$$W_{:,i} := \phi^i \sim f(\phi) \;\; \forall i, \qquad b := \mathbf{0},$$

where f is an arbitrary initialization scheme for the base network specifying a distribution over a vector of parameters, ϕ, and $W_{:,i}$ is the i-th column of W.

Weight-HyperInit reproduces any given initialization for each base network (π_ϕ), under the assumptions that $e = \mathbb{1}_i$ and the hypernetwork is linear. In this case, each column of W is simply a sample from the base scheme, one of which is selected for each task via the one-hot encoding. For example, given the task embedding $e = \mathbb{1}_3$, the following base network initialization ϕ_{init} is produced:

$$\phi_{\text{init}} = We + b = \begin{pmatrix} \phi_1^1 & \phi_1^2 & \phi_1^3 & \cdots \\ \phi_2^1 & \phi_2^2 & \phi_2^3 & \cdots \\ \phi_3^1 & \phi_3^2 & \phi_3^3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} + \mathbf{0} = \phi^3. \tag{2}$$

Moreover, in the case that there is also no bias, it is also equivalent to training separate networks for each task. Although these assumptions are not met for meta-RL, we find that this is still an improvement over default neural network initialization schemes.

Additionally, we propose *Bias-HyperInit*. This bias-only initialization is defined as follows:

$$W_{i,j} := 0 \ \forall i, j, \qquad b := \phi_{\text{shared}} \sim f(\phi).$$

Bias-HyperInit achieves an arbitrary initialization for the base network without any assumptions, by setting the parameters for any task to be the same at initialization. This encourages parameter sharing between base networks at the beginning of training, where possible, but also allows for separate base network parameters to be learned, if necessary. Under any set of assumptions, the base network is initialised to the following:

$$\phi_{\text{init}} = Wx + b = \begin{pmatrix} 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{pmatrix} + \phi_{\text{shared}} = \phi_{\text{shared}}, \tag{3}$$

where x is the final hidden layer of the hypernetwork. ($X = e$ in the case of a linear hypernetwork.)

Both methods initialize only W and b, which define the head of the network. All other layers in h may be initialized by any default initialization scheme. All such choices are detailed in supplementary materials.

4.3 Baselines

VariBAD. See Sec. 3.2.

FiLM. FiLM [27] is a convenient baseline situated between hypernetworks and the standard architecture. In FiLM, the hypernetwork generates biases for each layer, but only point-wise scales the activations instead of generating weight matrices. In this case, the base network has its own weights. Bias-HyperInit can easily be adapted to FiLM; details presented in supplementary materials.

HFI. HFI [3] is a state-of-the-art initialization method developed tested in the supervised learning setting. While we do compare to HFI, our methods are simpler and more general. Specifically, our methods work with arbitrary base network target initialization (as opposed to being tied to Kaiming) and our methods work with arbitrary base network architectures (without additional variance analysis). Moreover, our approach is straightforward to apply to additional methods, which we show by applying it to FiLM. Finally, although both HFI and Weight-HyperInit do make assumptions about the input to the hypernetwork, our strongest method, Bias-HyperInit, does not.

5 Experiments

In this section, we compare our hypernetwork architecture and initialization methods to baselines on 2D navigation [5], MuJoCo [1], ML1 [1], and ML10 [1] benchmarks. MuJoCo is a common meta-RL benchmark [7, 5, 28, 29, 12], as are toy 2D navigation tasks [7, 5, 28, 29]. These two benchmarks allow us to demonstrate that hypernetworks with default initialization methods fail to learn, whereas our proposed methods learn reliably. ML1 and ML10 are benchmarks in Meta-World [1]. These two benchmarks have greater room for improvement with state-of-the-art methods [5], which allow us to demonstrate improvement over the baseline architectures. Finally, we use two MuJoCo environments to investigate the performance of hypernetworks against standard architectures in terms of the number of parameters in the model overall.

Throughout our evaluation, we use two-tailed t-tests with $p = 0.05$ to determine significance. Details on model tuning and implementation are presented in supplementary materials.

5.1 Navigation and MuJoCo

Here we evaluate on the grid-world variant from Zintgraf et al. [5] as our 2D navigation task and MuJoCo [6].

Table 1: Comparison of return on grid-world and MuJoCo tasks over five random seeds (mean $\pm$ standard error). Entries in bold have insignificant difference from the highest-performing result.

Method	**Grid-World**	**Cheetah-Dir**	**Walker**	**Ant-Dir**	**Humanoid**
Standard	35.5 ± 0.4	$\mathbf{2104 \pm 87}$	$\mathbf{1828 \pm 38}$	1167 ± 16	$\mathbf{1842 \pm 233}$
Kaiming	32.1 ± 1.2	378 ± 169	331 ± 37	253 ± 86	266 ± 23
Normc	32.2 ± 0.5	356 ± 134	357 ± 60	264 ± 106	249 ± 18
Orthogonal	34.9 ± 0.5	1379 ± 310	1687 ± 98	1127 ± 93	1126 ± 76
HFI	$\mathbf{36.8 \pm 0.2}$	$\mathbf{2218 \pm 80}$	1618 ± 130	$\mathbf{1370 \pm 9}$	$\mathbf{1323 \pm 57}$
Weight-HyperInit	$\mathbf{36.1 \pm 0.5}$	$\mathbf{2066 \pm 119}$	1748 ± 57	$\mathbf{1346 \pm 7}$	1048 ± 21
Bias-HyperInit	$\mathbf{36.7 \pm 0.2}$	$\mathbf{2300 \pm 32}$	$\mathbf{1994 \pm 67}$	$\mathbf{1328 \pm 23}$	$\mathbf{1678 \pm 162}$

In Table 1 we see that default initializations frequently fail to learn while Bias-HyperInit learns reliably. Specifically, Kaiming and Normc initializations used with hypernetworks achieve far lower returns than all other methods. Orthogonal initialization is more competitive, however it is still significantly outperformed by Bias-HyperInit in every environment.

We also compare hypernetworks with Bias-HyperInit to the standard architecture and Bias-HyperInit to HFI. We see that Bias-HyperInit matches or exceeds the performance of HFI, with a significant improvement in Walker. Hypernetworks with Bias-HyperInit also significantly outperform the standard architecture on grid-world and Ant-Dir. In fact, Bias-HyperInit is not significantly outperformed by any other method. However, the standard architecture, HFI, and Bias-HyperInit all achieve near optimal performance, motivating an evaluation on Meta-World, on which the standard architecture has greater room for improvement [5].

5.2 ML1 and ML10

Here we evaluate on the more challenging Meta-World ML1 and ML10 benchmarks [1]. Note that we only test on the Pick-Place task from ML1, since VariBAD already achieves a 100% success rate on all other tasks [5].

Table 2: Comparison of meta-test success percentage on the Pick-Place ML1 task (ten seeds) and ML10 (three seeds).

Method		**Pick-Place**	**ML10**
Standard		4.4 ± 2.4[1]	5.6 ± 2.3
FiLM	Normc	5.5 ± 4.8	—
	Bias-HyperInit	$\mathbf{34.2 \pm 15.9}$	—
Hypernetwork	HFI	$\mathbf{25.5 \pm 14.5}$	$\mathbf{19.1 \pm 6.8}$
	Bias-HyperInit	$\mathbf{42.9 \pm 16.3}$	$\mathbf{19.0 \pm 5.3}$

In Table 2 we see significant improvement of hypernetworks with Bias-HyperInit over baseline architectures, as well as the efficacy of Bias-HyperInit on FiLM. Specifically, Bias-HyperInit significantly outperforms the standard architecture with greater than 9-fold and 3-fold increases in test success percentage on Pick-Place and ML10 respectively. On Pick-Place, although the difference is not statistically significant, hypernetworks with Bias-HyperInit also greatly exceed the performance of HFI and FiLM with Bias-HyperInit. In addition, Bias-HyperInit significantly improves the performance of FiLM, compared to Normc initialization. This further validates the efficacy and generality of Bias-HyperInit.

[1] Zintgraf et al. [5] report a success percentage of 29% for Pick-Place; however, we were not able to replicate this result. The difference is likely due to the fact that we use version two of the Meta-World benchmark. Regardless, we still achieve a greater return with hypernetworks.

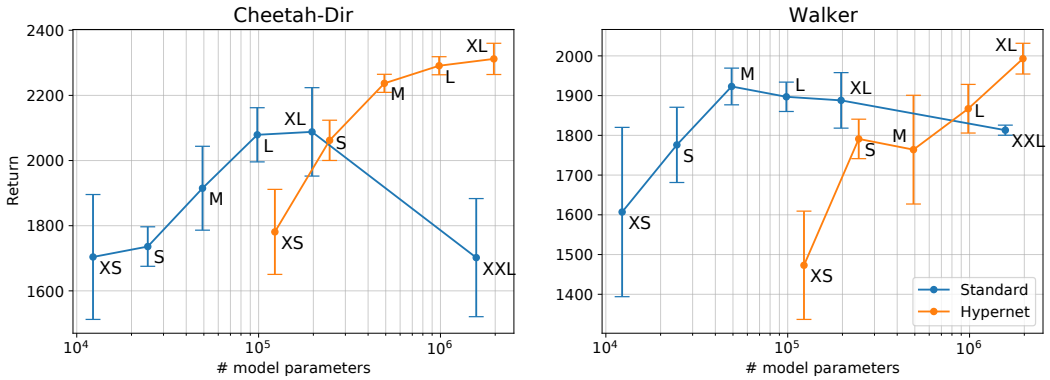


Figure 3: Performance of standard and hypernetwork models over a range of base policy architecture sizes on Cheetah-Dir and Walker. Architectures are presented in supplementary materials.

5.3 Network Size Comparison

Because hypernetworks learn a mapping from a task embedding to the parameters of a base network, they require significantly more parameters in the entire model than a standard architecture with the equivalent base network. For a fair comparison, we evaluate return over both a range of base network sizes and total number of parameters in the model on the Cheetah-Dir and Walker tasks (Figure 3). We find that hypernetworks consistently equal or outperform the standard architectures with the equivalent base network size. We also find that hypernetworks likewise outperform standard architectures for equivalent number of parameters in the entire model, when the total number of parameters in the model is sufficiently large.

6 Limitations

While our proposed methods are general, we build our experiments on top of VariBAD so cannot guarantee that these results transfer to other meta-RL methods with standard architectures. We chose VariBAD since it is the state of the art, but given additional time and computational resources, future work could investigate application to other meta-RL methods. Furthermore, as in any empirical study, there is no guarantee that our results hold on real robots or other meta-RL benchmarks. However, we have tested on seven standard meta-RL environments in total, including Meta-World, which was proposed specifically for addressing robotics. As much as is possible from simulated meta-RL experiments, these results give us confidence in a significant improvement over previous methods.

7 Conclusion

We used hypernetworks to improve a state-of-the-art method in meta-RL, evaluating over a range of benchmarks. In doing so, we demonstrated that hypernetworks are a promising path forward for meta-RL. Moreover, we showed that the initialization of the hypernetwork is crucial, as default initialization methods fail. To overcome this difficulty, we presented two novel initialization methods: Bias-HyperInit and Weight-HyperInit. Bias-HyperInit matched or exceeded the performance of existing methods from the supervised learning setting, while also being simpler and more general – applying to arbitrary base network initializations, base network architectures, and also improving FiLM. Using Bias-HyperInit, we showed that hypernetwork performance improves substantially over standard architectures. Finally, we demonstrated that hypernetworks outperform the standard VariBAD architecture for equivalently sized base policies, and outperform it at any size given sufficiently many parameters in the entire model. This paper additionally opens the path for future research extending meta-SL methods using hypernetworks [18, 22] and multi-task RL methods with separate parameters for each task [30, 31, 32] to meta-RL.

References

[1] 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, 2, 5, 6, 7)

[2] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *International Conference on Learning Representation (ICLR)*, 2017. (cited on p. 1, 2, 3, 4)

[3] O. Chang, L. Flokas, and H. Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. (cited on p. 1, 2, 3, 4, 6)

[4] 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, 4)

[5] 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. 2, 3, 4, 6, 7)

[6] 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. 2, 6)

[7] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017. (cited on p. 2, 6)

[8] R. Vuorio, S. Sun, H. Hu, and J. J. Lim. Multimodal model-agnostic meta-learning via task-aware modulation. *NeurIPS*, 2019. (cited on p. 2)

[9] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *arxiv*, 2017. (cited on p. 2)

[10] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *ICLR*, 2018. (cited on p. 2, 3)

[11] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv*, 2016. (cited on p. 2, 3)

[12] E. Najarro and S. Risi. Meta-learning through hebbian plasticity in random networks. *NeurIPS*, 2020. (cited on p. 2, 6)

[13] T. Miconi, K. Stanley, and J. Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3559–3568, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. (cited on p. 2)

[14] E. Sarafian, S. Keynan, and S. Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9301–9312. PMLR, 18–24 Jul 2021. (cited on p. 2)

[15] M. Peng, B. Zhu, and J. Jiao. Linear representation meta-reinforcement learning for instant adaptation. *arxiv*, 2021. (cited on p. 2)

[16] Z. Xian, S. Lal, H.-Y. Tung, E. A. Platanios, and K. Fragkiadaki. Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks. In *International Conference on Learning Representations*, 2021. (cited on p. 3)

[17] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Multi-task reinforcement learning without interference. *NeurIPS*, 2019. (cited on p. 3)

[18] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. (cited on p. 3, 8)

[19] T. Munkhdalai and H. Yu. Meta networks. *ICML*, 2017. (cited on p. 3)

[20] M. Przewiezlikowski, P. Przybysz, J. Tabor, M. Zieba, and P. Spurek. Hypermaml: Few-shot adaptation of deep models with hypernetworks, 2022. (cited on p. 3)

[21] D. Zhao, S. Kobayashi, J. Sacramento, and J. von Oswald. Meta-learning via hypernetworks. In *MeurIPS Workshop on Meta-Learning*, 2020. (cited on p. 3)

[22] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. (cited on p. 3, 8)

[23] K. Li and J. Malik. Learning to optimize. *ICLR*, 2017. (cited on p. 3)

[24] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. (cited on p. 4)

[25] A. M. Saxe, J. L. Mcclelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*, 2014. (cited on p. 4)

[26] 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. 4)

[27] 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. 6)

[28] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess. Meta reinforcement learning as task inference. *arXiv*, 2019. (cited on p. 6)

[29] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019. (cited on p. 6)

[30] Y. Teh, V. Bapst, W. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *NeurIPS*, 2017. (cited on p. 8)

[31] E. Parisotto, J. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *ICLR*, 2016. (cited on p. 8)

[32] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. *CVPR*, 2016. (cited on p. 8)