Meta-Learning via Hypernetworks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent developments in few-shot learning have shown that during fast adaption, gradient-based meta-learners mostly rely on embedding features of powerful pre-trained networks. This leads us to research ways to effectively adapt features and utilize the meta-learner's full potential. Here, we demonstrate the effectiveness of hypernetworks in this context. We propose a soft row-sharing hypernetwork architecture and show that training the hypernetwork with a variant of MAML is tightly linked to meta-learning a curvature matrix used to condition gradients during fast adaptation. We achieve similar results as state-of-art model-agnostic methods in the overparametrized case, while outperforming many MAML variants without using different optimization schemes in the compressive regime. Furthermore, we empirically show that hypernetworks do leverage the inner loop optimization for better adaptation, and analyse how they naturally try to learn the shared curvature of constructed tasks on a toy problem when using our proposed training algorithm.

1 Introduction

The ability to generalize previous knowledge and adapt quickly to novel environments has been subject of intense machine learning research in the past few years. One of the cornerstones of recent progress of meta-learning algorithms are gradient-based methods such as Model-Agnostic Meta-Learning (MAML) [1], which takes the initial parameters of a model as its meta-parameters. MAML recreates few-shot learning scenarios and trains the meta-parameters directly on how well they can solve new tasks after a few gradient steps, see Section 2.1.

Recent work has shown that MAML is mostly learning general features rather than finding fast-adaptable features deep inside its model. In fact, it was demonstrated that few-shot learning the hidden layers of a network has little to no effect on the performance of MAML [2, 3]. Furthermore, powerful models trained on rich enough data and without explicit meta-learning were shown to outperform most gradient-based meta-learning methods [4]. This is coherent with huge models being few-shot learners without explicit training at all [5]. These previous findings point to the untapped potential of fast adaptation as a promising area of improvement for such meta-learning methods.

One promising scalable approach is to separate the model into shared meta-parameters and context parameters [6]. Here, the context parameters are the only parameters updated in the inner loop. This way, the context parameters can learn task specific information and quickly adapt while the meta-parameters are used as general features.

Other approaches attempt to explicitly modulate the inner-loop training procedure by meta-learning learning rates or factorised preconditioning matrices [7–9]. Here, the actual model stays untouched and additional parameters are learned only to modulate the gradient with respect to the model parameters while learning new tasks.

Submitted to the 3rd Meta-Learning Workshop of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Here, we provide new insights on how hypernetworks [10, 11] implicitly combine these two seemingly different approaches. When trained with a variant of MAML, we show that hypernetworks are able to naturally modulate the inner loop optimization and adapt hidden layer features in a task-dependent manner. More generally, we demonstrate that hypernetworks implicitly learn features that directly support fast adaptation without any hand-designed add-ons or optimization variants. We propose a specific row-sharing hypernetwork architecture and show that it achieves state-of-the-art results compared to other gradient-based methods. Our method performs comparable with MAML even if the number of parameters is drastically compressed. Our main contributions are as follows:

- We demonstrate the effectiveness of hypernetworks for fast adaptation – in the compressed and overparametrized regime.
- By proposing a simple row-sharing hypernetwork architecture, we outperform most MAML variants without any explicit add-ons or optimization algorithm changes. Furthermore, we show empirically that hypernetworks can indeed learn useful inner-loop adaptation information and are not simply learning better network features.
- We show theoretically that in an simplified toy problem, hypernetworks can learn to model the shared structure that underlies a family of tasks. Specifically, its parameters model a preconditiong matrix equal to the inverse of the tasks' shared curvature matrix.

2 Background and Related Work

Our algorithm is intimately related to MAML, in particular to two of its variants, namely CAVIA [6] and Meta-Curvature [7]. We now briefly reintroduce both algorithms.

2.1 Model-Agnostic Meta-Learning

In the supervised learning setting, MAML optimizes the initial parameters of a model by minimizing the validation loss obtained after a few gradient steps. To do so, the training data is separated into training $\mathcal{D}^{\text{train}}$ and validation $\mathcal{D}^{\text{val}}$ sets. The validation loss is evaluated after one or more steps of stochastic gradient descent with respect to the training tasks on the meta-parameters. For task $\mathcal{T}_i \in \mathrm{T}$,

$$\theta_i = \theta - \alpha \nabla_\theta \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}\left(f_\theta(x), y\right). \tag{1}$$

The initialization parameters are then updated to minimize the validation loss:

$$\theta \leftarrow \theta - \gamma \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathrm{T}} \sum_{(x,y)\in\mathcal{D}_i^{\text{val}}} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i}(x), y\right). \tag{2}$$

2.2 Meta-Curvature

Recently, it has been shown that meta-learning a (compressed) preconditioning matrix $\mathbf{M}$ to modulate gradients used in the inner loop yields state-of-the-art performance [7, 9]. Interestingly, Meta-Curvature does not affect the model itself. Unlike MAML, this algorithm only adapts parameter gradients through a learnable matrix of meta-parameters:

$$\theta_i = \theta - \alpha \mathbf{M} \nabla_\theta \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}\left(f_\theta(x), y\right), \tag{3}$$

$$\theta \leftarrow \theta - \gamma \nabla_\theta \frac{1}{N} \sum_{T_i \in \mathrm{T}} \sum_{(x,y)\in\mathcal{D}_i^{\text{val}}} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i}(x), y\right), \tag{4}$$

$$\mathbf{M} \leftarrow \mathbf{M} - \gamma \nabla_{\mathbf{M}} \frac{1}{N} \sum_{\mathcal{T}_i \in \mathrm{T}} \sum_{(x,y)\in\mathcal{D}_i^{\text{val}}} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i}(x), y\right). \tag{5}$$

We note that the authors explore various interesting ways to construct the meta-parameter matrix $\mathbf{M}$.

2.3 Hypernetworks

For our approach, we draw direct inspiration from ideas related to *fast-and-slow weights* [10, 12, 13] and the more recently introduced *hypernetworks* [11, 14, 15]. One specific approach to implement these ideas is to express each layer as a learnable linear combination of templates. Since the templates are the same for each layer, the linear coefficients contain information on how the templates are shared across the model. Furthermore, it was shown that compression by using a small template bank is possible without affecting generalization [11]. More formally, the learnable parameters are the templates $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(k)}$ and each layer is fully determined by embedding parameters α as

$$\mathbf{W}^{(i)} := \sum_{j=1}^{k} \alpha_j^{(i)} \mathbf{T}^{(j)} \tag{6}$$

It is shown in [14] that the coefficient parameters α learn to reuse the various $\mathbf{T}^{(j)}$ and to share feature information.

More generally, a hypernetwork is any network that generates the weights of another network, given an input. Numerous research studies show the usefulness of hypernetworks and their respective variants in various areas such as visual-reasoning [16], continual learning [17, 18], transfer learning [19] and few-shot learning [3]. We point the reader to [20] for a broad analysis of hypernetworks and other multiplicative interactions within neural networks.

3 Meta-learning via Hypernetworks

Given the success of hypernetworks in these settings, we test here their abilities in the context of meta-learning. Other than in [21**?**], in the following we focus on the hypernetwork's capability to implicitly modulate the inner-loop optimization. More concretely, we suggest that the hypernetwork parameters θ play a role similar to the matrix $\mathbf{M}$ in Meta-Curvature. Our meta-learning algorithm is a variant of CAVIA which learns a general initialization for *both* task embeddings and hypernetwork parameters. Thus, the hypernetwork as well as an initialization for the task embedding are learned in the outer loop of our algorithm.

3.1 Soft Row-Sharing Architecture

In the following we investigate a simple linear soft row-sharing architecture for our hypernetwork and leave more complicated deep hypernetworks for future research. Each row of the generated network weight matrix will be a linear combination of the hypernetwork template's. More precisely, given a convolution layer from the output model $W^l \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}} \times K \times K}$, we construct, for each input channel c, some weights $R_c^l \in \mathbb{R}^{1 \times C_{\text{out}} \times K \times K}$ as a linear combination of templates $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(k)}$.

Conceptually, this can be seen as soft-sharing the rows of the parameter matrices across all layers. We then express R_c for a specific input channel as:

$$R_c^l := \sum_{j=1}^{k} \alpha_j^{(l,c)} \mathbf{T}^{(j)} \tag{7}$$

and concatenate each channel to create the layer weights:

$$W^l = [R_1^l, ..., R_{C_{in}}^l] \tag{8}$$

In this setting, the coefficients α represent the task embedding and $\theta = (\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(k)})$ the hypernetwork weights. For conciseness, we denote the network generated from a task embedding by $f_\theta(\alpha)$. In the inner loop, we first adapt the task embedding:

$$\alpha_i = \alpha - \beta \nabla_\alpha \sum_{(x,y) \in \mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i} \left(f_\theta(\alpha)(x), y \right). \tag{9}$$

Then, in the outer-loop, we train the hypernetwork as well as an initialization for the coefficients using the validation loss, evaluated at the network generated using the adapted task embedding:

$$\theta \leftarrow \theta - \gamma \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathrm{T}} \sum_{(x,y) \in \mathcal{D}_i^{\text{val}}} \mathcal{L}_{\mathcal{T}_i} \left(f_\theta(\alpha_i)(x), y \right). \tag{10}$$

$$\alpha \leftarrow \alpha - \gamma \nabla_{\alpha} \frac{1}{N} \sum_{\mathcal{T}_i \in \mathrm{T}} \sum_{(x,y) \in \mathcal{D}_i^{\mathrm{val}}} \mathcal{L}_{\mathcal{T}_i} \left(f_{\theta}(\alpha_i)(x), y \right) \tag{11}$$

We refer to a hypernetwork meta-learned in this fashion as a Meta-Hypernetwork (MH).

Note that due to the multiplicative interaction between α and θ within f, θ modulates ∇_{α} in the inner-loop similar to a block-wise conditioning matrix. The rank of the block matrix is determined by the number of templates shared across the layers of the network.

4 Experiments

To show the hypernetwork's effectivness in fast adaption, we conduct experiments on standard few-shot regression and classification benchmarks.

4.1 Few-shot regression on Sinusoidal Task

Our first experiment follows the K-shot regression protocol outlined in [1, 8]. Given K input-output pairs $(\mathbf{x}, f(x))$, with x uniformly sampled from $[-5, 5]$ and f a sinusoidal determined by random phase and amplitude in [0,π] and $[0.1, 5.0]$ respectively, the goal is to quickly reconstruct $f(x)$ from these few examples.

Table 1: Few-shot Regression MSE results

Method	5-shot	10-shot
MAML	$0.686^{\pm 0.070}$	$0.435^{\pm 0.039}$
Meta-SGD [8]	$0.482^{\pm 0.061}$	$0.258^{\pm 0.026}$
LayerLR [22]	$0.528^{\pm 0.068}$	$0.269^{\pm 0.027}$
MC2 [7]	$0.405^{\pm 0.048}$	$0.201^{\pm 0.020}$
MH	$0.501^{\pm 0.082}$	$0.281^{\pm 0.072}$

For fair comparison, we use the same neural network architecture proposed in [1], which consists of a fully-connected network with 2 hidden layers of size 40. For each of the rows of the hidden layers, we generate the weights with a linear combination of 40 templates. We use our Meta-Hypernetwork (MH) method to train the meta-learner. MH performs comparable with current state-of-the-art MAML-variants, see Table 1. Results from methods we compare to are taking from [7]. Details about the experiment can be found in appendix A.1.

4.2 Few-Shot Classification on MiniImageNet

To further demonstrate the effectiveness of our MH approach, we conduct experiments on the classic few-shot classification benchmark MiniImagenet. For fair comparison, we again use the model proposed in [1] for all experiments, which consists of four convolutional layers with 64 filters, followed by a fully connected layer. Each row of these convolutional layers is represented following our hypernetwork architecture described above.

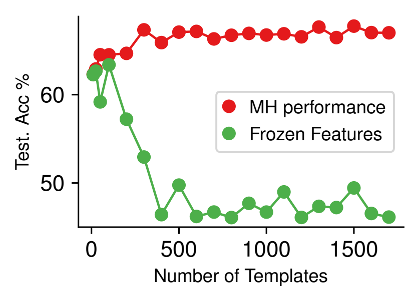

Figure 1: Difference in performance for 5-shot 5-way classification on MiniImageNet when adapting and not adapting the hypernetworks templates.

The number of templates of the hypernetwork controls the expressiveness of the inner-loop gradients modulation and therefore is unsurprisingly an important impact on algorithm's performance. In the following, we investigate three versions of MH with different number of parameters compared to the original model: compressed (**MH-C**), comparable (**MH**) and overparametrized (**MH-O**), with 50, 600 and 1500 templates respectively.

As expected, we observe a performance difference for our three variants in Table 2 in the 1-shot and 5-shot setting. We emphasise the effectiveness our approach in all scenarios compared to similar gradient based methods using other architectures or explicit gradient modulation.

Table 2: Few-shot classification results for MiniImagenet

Method	1-shot 5-way	5-shot 5-way
MAML	$48.07^{\pm 1.75}$	$63.15^{\pm 0.91}$
CAVIA$^{(32)}$ [6]	$47.24^{\pm 0.65}$	$59.05^{\pm 0.54}$
Meta-SGD [8]	$50.47^{\pm 1.87}$	$64.03^{\pm 0.94}$
MH-C	$48.64^{\pm 0.33}$	$64.52^{\pm 0.51}$
REPTILE [23]	$49.97^{\pm 0.32}$	$65.99^{\pm 0.58}$
CAVIA$^{(128)}$	$49.84^{\pm 0.68}$	$64.63^{\pm 0.54}$
CAVIA$^{(512)}$	$51.82^{\pm 0.65}$	$65.85^{\pm 0.55}$
MH	$49.41^{\pm 0.96}$	$67.16^{\pm 0.42}$
MH-O	$52.50^{\pm 0.61}$	$67.76^{\pm 0.34}$
MC [7]	$54.23^{\pm 0.88}$	$68.47^{\pm 0.69}$

4.3 Hypernetwork effects on fast adaptation

To investigate the ability of MHs to shape inner-loop learning of its embeddings in the neural network hidden layers, we perform the experiments described in [2]. After training the hypernetwork, we freeze the layers during fast adaptation and only update the fully-connected layer (Frozen). This allows us to investigate the contribution of the meta-learned hypernetwork to the inner-loop optimization. Indeed, we observe (see Table 3) a dramatic effect on performance when disabling the training of hypernetwork embeddings. This effect is reinforced for larger hypernetwork sizes, see Figure 1.

5 Theoretical analysis on a toy problem

Recent work has shown that given a local smooth and convex optimization landscape, linearizing a network around some weights and then taking the second-order Taylor expansion of the loss function gives an accurate enough quadratic objective [24]. This motivates the study of noisy quadratic models (NQMs) as analytically-tractable simplifications of real-world problems [25].

Table 3: Usefulness of feature adaptation for 5-shots on MiniImagenet

Methods	Adapting	Frozen
MAML	$63.15^{\pm 0.91}$	$61.50^{\pm 0.50}$
MH-C	$64.52^{\pm 0.51}$	$59.18^{\pm 0.49}$
MH	$67.16^{\pm 0.42}$	$46.21^{\pm 0.27}$
MH-O	$67.76^{\pm 0.34}$	$49.44^{\pm 0.71}$

In this section, we formulate a toy few-shot regression learning problem using a noisy quadratic model and show analytically that an optimal linear hypernetwork trained with our proposed algorithm seeks to learn the underlying structure of the different tasks.

5.1 Problem Definition

We define a linear hypernetwork of the form

$$W = \theta(\alpha_0 + \alpha) \tag{12}$$

with α the task specific embedding (set to 0 at the beginning of each task), α_0 and θ, the hypernetwork parameters.

To adapt the NQM to our setting, we make the simplifying assumption that all tasks consist of the minimization of a quadratic scalar loss which share a common underlying curvature H. The tasks will differ only in the location of the global minimum of the quadratic loss, and can thus be uniquely identified by their optimal vector. We assume these vectors follow a Gaussian distribution $\mathcal{N}(W^*, \Sigma)$.

The loss $\mathcal{L}$ of a given task t identified by the weight vector ϵ_t is therefore defined as

$$\mathcal{L}(W) = \frac{1}{2}(W - \epsilon_t)^T H (W - \epsilon_t) \tag{13}$$

where the Hessian matrix H describes the curvature common to all tasks.

5.2 Hypernetworks learn underlying task similarities

Fast adaptation in the given context is measured on the number of steps necessary to minimize (up to a given error) the loss given a new task ϵ.

We begin by computing the gradient w.r.t. the hypernetwork embeddings,

$$\frac{\partial L}{\partial \alpha} = \theta^T H(W - \epsilon) \tag{14}$$

resulting in the following change of W when taking a gradient step

$$\Delta W = W - \theta(\alpha_0 + \alpha - \gamma \frac{\partial L}{\partial \alpha}) = -\gamma\theta \frac{\partial L}{\partial \alpha} = -\gamma\theta\theta^T H(W - \epsilon). \tag{15}$$

We observe that this leads to an optimal single step if $\gamma\theta\theta^T = H^{-1}$. Indeed, this condition ensures that the updated model will be equal to ϵ, which is the optimal weight vector for the given task. Note that the weight vector θ acts as the optimal preconditioner for the task embeddings α, allowing for fast and efficient adaptation to new tasks. The remainder of this section will outline how the hypernetwork may implicitly learn such weight vector when meta-learning over the tasks of this NQM.

During training, the inner loop's task embedding updates are followed by updates in the hypernetwork's parameters in the outer loop. Since the inner and outer loop optimization are performed on the same task, the noise ϵ of their respective loss is identical. Note that the choice of reusing the same task in the inner as well as outer loop is crucial for our algorithm to learn the common structure of the tasks (see appendix for further justification).

The outer loop loss $\mathcal{L}$ is therefore the same as the inner loop loss $\mathcal{L}$, but evaluated with the updated task embedding after a single step, $\alpha = -\gamma\frac{\partial L}{\partial \alpha}$:

$$\tilde{\mathcal{L}} = \mathcal{L}(\theta, \alpha_0, -\gamma \frac{\partial L}{\partial \alpha}) = \frac{1}{2}(\theta\alpha_0 - \gamma\theta\frac{\partial L}{\partial \alpha} - \epsilon)^T H(\theta\alpha_0 - \gamma\theta\frac{\partial L}{\partial \alpha} - \epsilon) \tag{16}$$

$$= \frac{1}{2}((I - \gamma\theta\theta^T H)(\theta\alpha_0 - \epsilon))^T H((I - \gamma\theta\theta^T H)(\theta\alpha_0 - \epsilon)) \tag{17}$$

In the outer loop, the hypernetwork weights θ, α_0 are optimized based on the stochastic validation loss $\tilde{\mathcal{L}}$ such that the expectation of the gradient reaches 0.

$$\mathbb{E}_\epsilon \frac{\partial \tilde{L}}{\partial \alpha_0} = 0 \tag{18}$$

$$\mathbb{E}_\epsilon \frac{\partial \tilde{L}}{\partial \theta} = 0 \tag{19}$$

The system of equation (18) and (19) is solved when $(I - \gamma H\theta\theta^T) = 0$. In other words, when $\gamma\theta\theta^T = H^{-1}$ (see appendix for details).

6 Conclusion

We showed that meta-learning a hypernetwork and adapting its embedding is a natural candidate method to create good few-shot learners. Our method's performance demonstrates that hypernetworks offer a good combination of feature acquisition and quick adaptation. MH is scalable and simple enough to be adapted to different model architectures and its effectiveness shed light on the role of overparametrization in meta-learning.

Broader Impact

Fast adaptation and generalization on a wide range of environments is key to improve future artificial intelligent technologies. Although neural networks are highly flexible function approximators, they often do not generalize well to unseen tasks. In this work, we shed light on this problem and offer solutions to mitigate this problem. This line of research can have widespread impact in fields such as robotics, data analytics and artificial intelligence.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, July 2017.

[2] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

[3] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2019.

[4] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. volume 97, pages 7693–7702, 2019.

[7] Eunbyung Park and Junier B. Oliva. Meta-curvature. In *Advances in Neural Information Processing Systems 32*, pages 3309–3319, 2019.

[8] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[9] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2020.

[10] Jürgen Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, January 1992.

[11] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[12] Christoph von der Malsburg. The correlation theory of brain function. *Models Neural Netw.*, 2, 01 1994.

[13] Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, pages 177–186. Erlbaum, 1987.

[14] Pedro Savarese and Michael Maire. Learning implicitly recurrent CNNs through parameter sharing. In *International Conference on Learning Representations*, 2019.

[15] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[16] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

[17] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.

[18] Vikranth Dwaracherla, Xiuyuan Lu, Morteza Ibrahimi, Ian Osband, Zheng Wen, and Benjamin Van Roy. Hypermodels for exploration. In *International Conference on Learning Representations*, 2020.

[19] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. *arXiv preprint arXiv:1803.10082*, 2018.

[20] Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020.

[21] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes, 2020.

[22] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019.

[23] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

[24] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*, pages 8580–8589, 2018.

[25] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems 32*, pages 8194–8205, 2019.

[26] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[27] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638, 2016.