Appendix

A Theoretical Analysis of Linear Models for Meta-Learning

This section provides more details to §4 in the main text.

A.1 Analytic Solution of 1D Linear Regression

We use

$$\theta_\alpha = \theta - \alpha \nabla \ell_\tau(\theta) \tag{27}$$

as a short-hand. The gradient of the MAML loss $\mathcal{L}^{\text{MAML}}$ is given by

$$\frac{\partial \mathcal{L}^{\text{MAML}}}{\partial \theta} = \mathbb{E}_\tau(\boldsymbol{I} - \alpha H_\tau(\theta)) \left.\frac{\partial \ell_\tau}{\partial \theta}\right|_{\theta - \alpha \nabla \ell_\tau} = \mathbb{E}_\tau(I - \alpha H_\tau(\theta)) \nabla \ell_\tau(\theta_\alpha) \tag{28}$$

We setup the problem as follows. Let the task parameter $\theta_\tau \sim N(0,1)$ be a normal distributed scalar. Let the covariate $x \sim N(0,1)$ be a normal distributed scalar and the outcome be $y \sim N(\theta_\tau x, 1)$. We investigate two models for their meta-learning performance:

$$\text{SHALLOW:} \quad \hat{y} = cx, \qquad \text{DEEP:} \quad \hat{y} = abx \tag{29}$$

Note that the "deep" model is overparameterized. For each task, we use the least-square loss

$$\ell_\tau = \frac{1}{2}\, \mathbb{E}_{p(x,y|\theta_\tau)} (y - cx)^2, \quad \text{or} \quad \ell_\tau = \frac{1}{2}\, \mathbb{E}_{p(x,y|\theta_\tau)} (y - abx)^2 \tag{30}$$

A.1.1 Shallow Model

For the shallow model, the gradient is

$$\nabla \ell_\tau = \mathbb{E}_{p(x,y|\theta_\tau)} (cx - y)x = c - \theta_\tau \tag{31}$$

Thus, the loss for the τ-th task is given by

$$\ell_\tau(c - \alpha \nabla \ell_\tau) = \mathbb{E}_{p(x,y|\theta_\tau)} (y - [(1-\alpha)c + \alpha\theta_\tau]x)^2 \tag{32}$$

$$= [(1-\alpha)c + \alpha\theta_\tau]^2 - 2[(1-\alpha)c + \alpha\theta_\tau]\theta_\tau \tag{33}$$

Simplifying and completing the square on the θ_τ^2 term with constants, we arrive at

$$\ell_\tau(c - \alpha \nabla \ell_\tau) = (1-\alpha)^2 (c - \theta_\tau)^2 + \text{CONST} \tag{34}$$

Taking the expectation of the loss with respect to $p(\tau)$, we have

$$\mathcal{L}^{\text{MAML}}_{\text{SHALLOW}} = 2(1-\alpha)^2 c^2 + \text{CONST} \tag{35}$$

A.1.2 Deep Model

The gradients of the linear regression loss w.r.t a, b are given by:

$$\frac{\partial l_\tau}{\partial a} = ab^2 - b\theta_\tau \tag{36}$$

$$\frac{\partial l_\tau}{\partial b} = a^2 b - a\theta_\tau \tag{37}$$

The post-adaptation parameters are obtained by taking one step of gradient descent with learning rate α:

$$a \leftarrow a - \alpha b\,(ab - \theta_\tau) \tag{38}$$
$$b \leftarrow b - \alpha a\,(ab - \theta_\tau) \tag{39}$$

This gives us the expression for the MAML loss:

$$\begin{aligned}
\mathcal{L}^{\text{MAML}}_{\text{DEEP}} &= \frac{1}{2}\mathop{\mathbb{E}}_{\tau}\mathop{\mathbb{E}}_{x,y}\left[y - (a - \alpha b(ab-\theta_\tau)) \cdot (b - \alpha a(ab-\theta_\tau))x\right]^2 && (40)\\
&= \frac{1}{2}\mathop{\mathbb{E}}_{\tau}\mathop{\mathbb{E}}_{x,y}\left[y - f(a,b,\theta_\tau)x\right]^2 && (41)\\
&= \frac{1}{2}\mathop{\mathbb{E}}_{\tau}\left[\theta_\tau^2 - 2\theta_\tau + f^2(a,b,\theta_\tau)\right] && (42)\\
&= \frac{1}{2} + \frac{1}{2}\mathop{\mathbb{E}}_{\tau} f^2(a,b,\theta_\tau) - \frac{1}{2}\mathop{\mathbb{E}}_{\tau}\theta_\tau f(a,b,\theta_\tau) && (43)
\end{aligned}$$

where $f(a,b,\theta_\tau) = (a - \alpha b(ab-\theta_\tau)) \cdot (b - \alpha a(ab - \theta_\tau))$.
We now take a closer look at the term $\mathbb{E}\, f^2$:

$$\begin{aligned}
f^2(a,b,\theta_\tau) &= \left[ab - \alpha a^2(ab-\theta_\tau) - \alpha b^2(ab-\theta_\tau) + \alpha^2 ab(ab-\theta_\tau)^2\right]^2 && (44)\\
&= \left[ab - \alpha(a^2+b^2)(ab-\theta_\tau) + \alpha^2(ab-\theta_\tau)^2\right]^2 && (45)\\
&= \left[ab - \alpha(a^2+b^2)ab + \alpha^2a^3b^3 + (\alpha(a^2+b^2) - 2\alpha^2a^2b^2)\theta_\tau + \alpha^2ab\theta_\tau^2\right]^2 && (46)\\
&= \left[p_1 + p_2\theta_\tau + p_3\theta_\tau^2\right]^2 && (47)\\
& && (48)
\end{aligned}$$

where we let:

$$p_1 = ab - \alpha(a^2+b^2)ab + \alpha^2a^3b^3, \qquad p_2 = \alpha(a^2+b^2) - 2\alpha^2a^2b^2, \qquad p_3 = \alpha^2 ab \tag{49}$$

We then obtain

$$\mathbb{E}\, f^2(a,b,\theta_\tau) = p_1^2 + p_2^2 + 3p_3^2 + 2p_1p_3 \tag{50}$$

and where the last equality is obtained be remembering the expectation properties of θ_τ:

$$\mathbb{E}_{p(\tau)}\,\theta_\tau^2 = 1, \qquad \mathbb{E}_{p(\tau)}\,\theta_\tau^3 = 0, \qquad \mathbb{E}_{p(\tau)}\,\theta_\tau^4 = 3. \tag{51}$$

Furthermore, we obtain

$$\mathbb{E}\,\theta_\tau f(a,b,\theta_\tau) = p_2 \tag{52}$$

This gives us a final expression for the MAML loss:

$$\mathcal{L}^{\text{MAML}}_{\text{DEEP}} = \frac{1}{2}(1 + p_1^2 + p_2^2 + 3p_3^2 + 2p_1p_3 - 2p_2). \tag{53}$$

We use the Matlab Symbolic Toolbox to help us simplify the rest of the calculation. The code is given below

```
%% We use sa, sb, salpha as the "symbolic version" of a, b, and alpha
syms sa sb salpha

pp1 = sa*sb*(1- salpha*(sa^2+sb^2)+ salpha^2*sa^2*sb^2);
pp2 = salpha*(sa^2+sb^2) - 2 *salpha^2 *sa^2*sb^2;
pp3 = salpha^2*sa*sb;

L = pp1*pp1 + pp2*pp2 + 3*pp3*pp3 + 2*pp1*pp3 -2 *pp2;
diffa = diff(L, sa);
diffb = diff(L, sb);

H = [diff(diffa, sa) diff(diffa, sb);diff(diffb, sa) diff(diffb, sb)];
```

Stationary Points The gradients are complicated polynomials in a and b. Instead, we give the specific cases:

$$\left.\frac{\partial \mathcal{L}_{\text{DEEP}}^{\text{MAML}}}{\partial a}\right|_{b=0} = 4\alpha a(\alpha a^2 - 1), \qquad \left.\frac{\partial \mathcal{L}_{\text{DEEP}}^{\text{MAML}}}{\partial b}\right|_{b=0} = 0 \tag{54}$$

We immediately can derive that there are at least 5 stationary points:

- $(a = 0, b = 0)$, with Hessian
$$H = \begin{bmatrix} -4\alpha & 0 \\ 0 & -4\alpha \end{bmatrix} \tag{55}$$
- $(a = \pm\frac{1}{\sqrt{\alpha}}, 0)$, with Hessian
$$H = \begin{bmatrix} 8\alpha & 0 \\ 0 & 6\alpha^3 \end{bmatrix} \tag{56}$$
- $(a = 0, b = \pm\frac{1}{\sqrt{\alpha}})$, with Hessian
$$H = \begin{bmatrix} 6\alpha^3 & 0 \\ 0 & 8\alpha \end{bmatrix} \tag{57}$$

B Supplementary Experiments

This section provides additional experimental evidence to complement the evidence presented in the main text.

B.1 Binary Logistic Regression

To resonate more with our experiments of using (multinominal) logistic regression models on Omniglot, CIFAR and MNIST datasets, we also analyze binary logistic regression models on synthetic data.

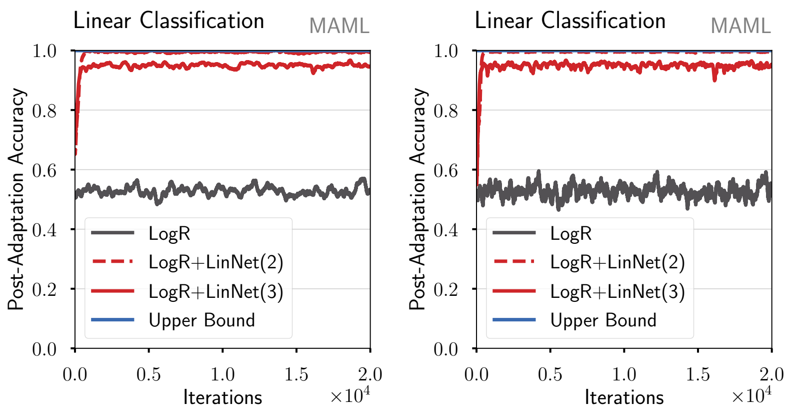

Figure B.1: Meta-learning of linear models on synthetic data. For linear separable data, MAML fails on logistic regression but succeed on logistic regression augmented with 2 or 3 linear layers. **(Left)** Meta-train accuracies. **(Right)** Meta-test accuracies.

B.1.1 MAML on Logistic Regression and Logistic Regression with Linear Layers

We randomly sample a set of 2-dimensional task parameters $\theta_\tau \in \mathbb{R}^2$ from a standard multivariate spherical Gaussian and use each of them to define a linear decision boundary of a binary classification task. We sample inputs from a 2-dimensional multivariate spherical Gaussian and the binary outputs for each task are sampled from $y \sim \sigma(\theta_\tau^{\mathrm{T}} \boldsymbol{x})$, where $\sigma()$ is the sigmoid function.

By construction, a logistic regression (LR) is sufficient to achieve very high accuracy on any task. But can MAML learn a linear classifier from a randomly sampled subset of training tasks that is able to adapt quickly to the test tasks? It is intuitive to see the minimizer for the MAML loss $\mathcal{L}^{\text{MAML}}$ is the origin due to the rotation invariances of both the task parameters and the inputs. The origin thus provides the best initialization to adapt to new tasks by not favoring any particular task.

Fig. B.1(Left) reports the 1-step post-adaptation accuracy on the test tasks for the meta-learned logistic regression model. Equally surprising as the previous results, the model fails to perform better than chance. However, adding linear layers "rescues" the meta-learning; LR with 2 or 3 linear layers attains nearly perfect classification on any task.

B.2 Logistic Regression Failure Modes

As argued in §5.1, shallow logistic regression models fail to meta-learn on Omniglot and CIFAR-FS. Fig. B.2 displays this same failure mode on the mini-ImageNet dataset. To simplify computations, we downscaled the original mini-ImageNet images (84x84 pixels, with RGB channels) to the same size as CIFAR-FS. (32x32 pixels, with RGB channels)

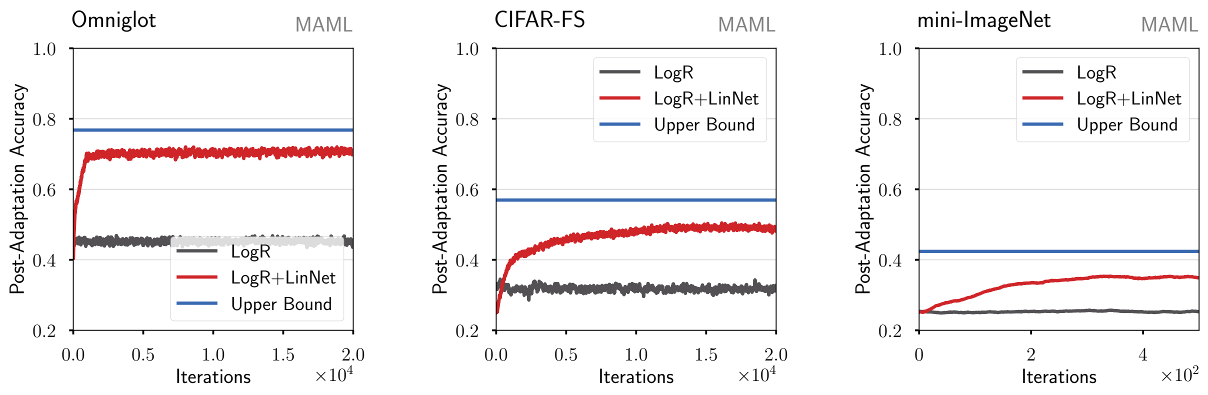

Figure B.2: Meta-training logistic regression models with MAML on Omniglot, CIFAR-FS, and mini-ImageNet led to poor performances. Adding linear nets improves meta-learning significantly, *without* changing the model's capacity.

B.3 Linear Layers Cannot Be Collapsed

Our previous results have shown that when MAML cannot meta-train well, adding multiple linear layers helps to meta-learn, cf. Fig. B.1 and Fig. 1 of the main text.

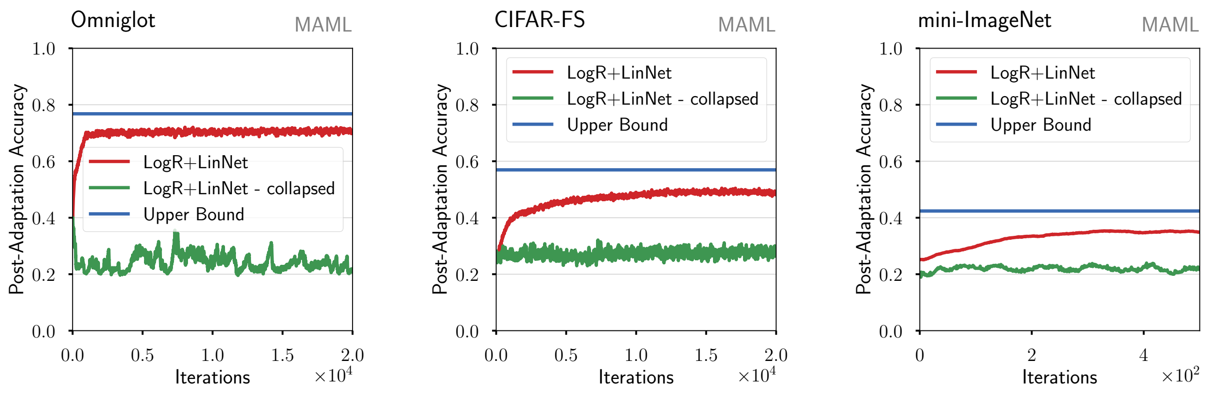

Figure B.3: Collapsing multiple linear layers into a single layer before adapting to new tasks (green curves) leads to poor adaptation results.

But after meta-training, can we collapse those multiple linear layers into a single linear layer so as to reduce the model size?

Fig. B.3 shows that collapsing erases the model's ability to adapt to new tasks. This suggests that **the solutions identified in models with additional linear layers need to stay in the overparameterized space to be effective and they cannot be collapsed before adaptation**.

B.4 Does SGD-induced implicit regularization for linear networks matter for meta-learning?

Implicit regularization refers to the bias towards solutions with different generalization abilities when stochastic gradient descent (SGD) is used to optimize overparameterized models Gidel et al. (2019).

One might want to argue that implicit regularization explains why additional linear layers in our experiments (Fig. 1 of the main text, Fig. B.1) help to meta-learn. This is an interesting hypothesis, though we do not think it can fully explain what is observed.

First, as those experiments have shown, for models augmented with linear nets, there is very little difference between training accuracies and meta-testing accuracies. Moreover, for models without linear nets, there is very little difference between the two phases either. Thus, while SGD could lead to solutions with different generalization abilities in regular supervised learning settings, our experiments do not show there is an overfitting issue in the experiments here for meta-learning.

The difference between parsimonious and overparameterized models (with linear layers) could be due to the difference in solutions. However, the solutions are in space of different dimensionality. Fig. B.3 shows that **after** we collapse the overparameterize models such that the solutions are in the same space, the solutions from the collapsing becomes as ineffective (in terms of leading to poor adaptation results) as the original models. This suggests that SGD is **not** identifying a solution in the overparameterized space that could have been identified in the original space. In other words, SGD is not biasing towards any solution in the overparameterized space that could have made difference in the original space.

B.5 ANIL with Meta-Optimizers

In §5.3, we showed that MAML augmented with meta-optimizers was able to meta-learn in shallow non-linear networks. Table B.4 shows similar results when using ANIL as the meta-learning algorithm. (Note: this table omits T-Nets, as it is not compatible with ANIL.) We observe that ANIL always benefits from the combination with a meta-optimizer, and specifically that it benefits most from KFO.

Table B.4: Meta-Optimizers Improve ANIL Meta-Learnability on CNN(2)

Dataset	ANIL	ANIL w/		
		MSGD	MC	KFO
Omniglot	91.00	92.47	92.67	**94.40**
CIFAR-FS	66.10	67.55	67.43	**68.81**
mini-ImageNet	56.42	57.45	59.07	**62.65**

B.6 Effect of Number of Layers

Figure B.4 complements the results in §5.3 when varying the number of layers in the convolutional network. (We include again results on Omniglot and CIFAR-FS for convenience.) As we observed in the main text, shallower models greatly benefit from the additional meta-optimization parameters which alleviates the burden of learning how to adapt. But, this benefit dampens as we increase the number of convolutional layers – the gap between META-KFO and MAML shrinks – since the additional upper layers implicitly transform the gradient of lower layers, thus enabling successful meta-learning.

C Details on KFO

C.1 Other factorization schemes

A technical issue arises when expressing the meta-optimizer U_ξ as a neural network: the dimensionality of modern (model) network architectures ranges in the tens of thousands, if not more. To address those computational

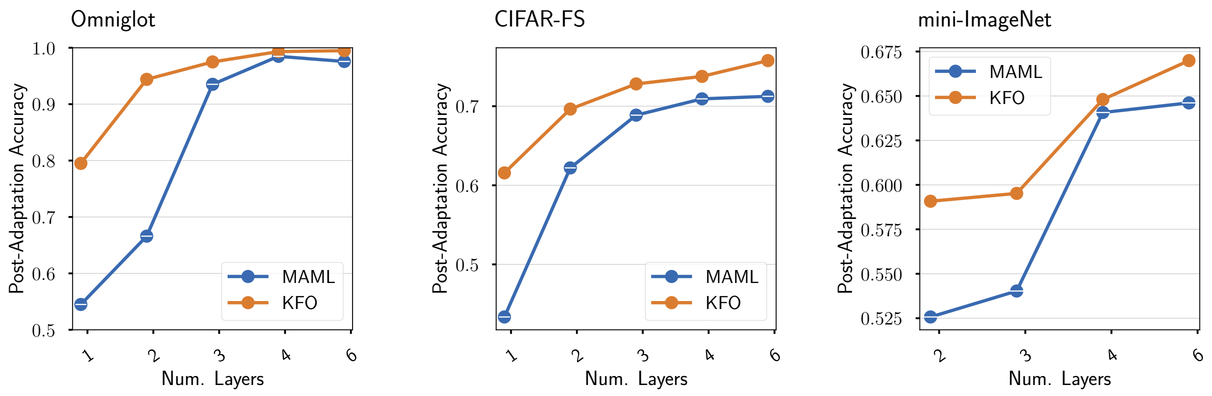

Figure B.4: Complementing Fig. 4, we vary the number of convolutional layers in the model. As the model size increases the performance of both methods improves, which can be attributed to the model's increased capacity to learn the target task. However, the net gain from META-KFO over MAML has diminishing returns as the number of layers increases since the benefit of the external meta-optimizer reduces as the upper layers of the larger models have more capacity to meta-learn to control their own bottom layers.

and memory issues, we learn one meta-optimizer network per matrix parameter in the model network and express the weights of the optimizer neural network as a Kronecker factorization such that for each weight $W \in \mathbb{R}^{m \times n} : W = R^\top \otimes L$, where $R \in \mathbb{R}^{m \times m}$ and $L \in \mathbb{R}^{n \times n}$.

Many matrix factorization schemes could be used and would result in different modeling and computational trade-offs. For example, using a low-rank Cholesky factorization $A = LL^T$ where $L \in \mathbb{R}^{k \times r}$ allows to interpolate between computational complexity and decomposition rank by tuning the additional hyper-parameter r. The Cholesky decomposition might be preferable to the Kronecker one in memory-constrained applications, since r can be used to control the memory requirements of the meta-optimizer. Moreover, such a decomposition imposes symmetry and positiveness on A, which might be desirable when approximating the Hessian or its inverse.

In this work, we preferred the Kronecker decomposition over alternatives for three reasons: (1) the computational and memory cost of the Kronecker-product are acceptable, (2) $R^\top \otimes L$ is full-rank whenever L, R are full-rank, and (3) the identity matrix lies in the span of Kronecker-factored matrices. In particular, this last motivation allows meta-optimizers to recover the gradient descent update by letting R, L be the identity.

C.1.1 Schematics and Pseudo-code

Pseudo-code for meta-optimizers is provided in Algorithm 1, and a schematic of the model-optimizer loop in Figure C.5.

Algorithm 1 Meta-Learning with Meta-Optimizers

```
Require: Fast learning rate $\alpha$, Initial parameters $\theta^{(\text{init})}$ and $\xi^{(\text{init})}$, Optimizer Opt
 1: while $\theta^{(\text{init})}, \xi^{(\text{init})}$ not converged do
 2:     Sample task $\tau \sim p(\tau)$
 3:     $\theta_1 = \theta^{(\text{init})}, \quad \xi_1 = \xi^{(\text{init})}$
 4:     for step $t = 1, \ldots, T$ do
 5:         Compute loss $\ell_\tau(\theta_t)$
 6:         Compute gradients $\nabla_{\theta_t} \ell_\tau(\theta_t)$ and $\nabla_{\xi_t} \ell_\tau(\theta_t)$
 7:         Update the meta-optimizer parameters $\xi_{t+1} = \xi_t - \alpha \nabla_{\xi_t} \ell_\tau(\theta_t)$
 8:         Compute the model update $U_{\xi_{t+1}}(\nabla_{\theta_t} \ell_\tau(\theta_t))$
 9:         Update the model parameters $\theta_{t+1} = \theta_t - U_{\xi_{t+1}}(\nabla_{\theta_t} \ell_\tau(\theta_t))$
10:     Update model and meta-optimizer initializations
11:         $\theta^{(\text{init})} \leftarrow \theta^{(\text{init})} - \texttt{Opt}(\nabla_{\theta^{(\text{init})}} \ell_\tau(\theta_T))$
12:         $\xi^{(\text{init})} \leftarrow \xi^{(\text{init})} - \texttt{Opt}(\nabla_{\xi^{(\text{init})}} \ell_\tau(\theta_T))$
```

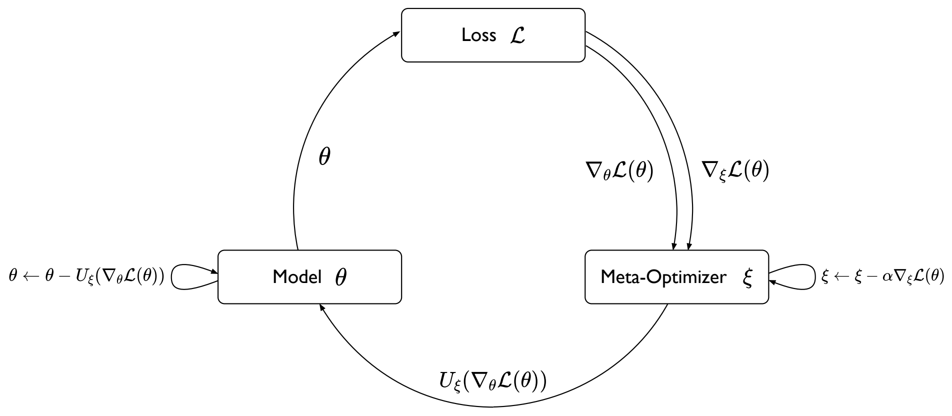
Loss $\mathcal{L}$
θ
$\nabla_\theta \mathcal{L}(\theta)$
$\nabla_\xi \mathcal{L}(\theta)$
$\theta \leftarrow \theta - U_\xi(\nabla_\theta \mathcal{L}(\theta))$
Model θ
Meta-Optimizer ξ
$\xi \leftarrow \xi - \alpha \nabla_\xi \mathcal{L}(\theta)$
$U_\xi(\nabla_\theta \mathcal{L}(\theta))$

Figure C.5: Schematic of model-optimizer loop

Table D.5: **Adaptation learning rates** for the linear model experiments in the main text and the Suppl. Material

Dataset	Model	Meta learning rate	Adaptation learning rate
Synthetic	LR	0.01	0.900
Synthetic	LR+LinNet(2)	0.1	0.900
Synthetic	LR+LinNet(3)	0.1	0.900
Omniglot	LR	0.0005	1.0
Omniglot	LR+LinNet	0.0005	0.08
CIFAR-FS	LR	0.0005	0.01
CIFAR-FS	LR+LinNet	0.0001	0.02
mini-ImageNet	LR	0.0005	0.01
mini-ImageNet	LR+LinNet	0.0005	0.01

D Hyperparameters and details for reproducibility

We report the hyper-parameter values for the experiments of the main text. Each experimental setup is tuned independently based on post-adaptation accuracies obtained on validation tasks.

All experiments are implemented on top of PyTorch Paszke et al. (2019) and learn2learn Arnold et al. (2020). Moreover, our work relied on tools from the Python scientific ecosystem Van Rossum and Drake (1995), including numpy Hunter (2007), matplotlib Harris et al. (2020), and scipy Virtanen et al. (2020). We provide example implementations of Meta-KFO at: `https://github.com/Sha-Lab/kfo`

Linear Models On the 1D linear regression, both SHALLOW and DEEP models use a fast-adaptation learning rate set to 0.1. Their parameters are optimized by SGD with a meta learning rate set to 0.01, and a momentum term set to 0.9. On the vision datasets (Omniglot, CIFAR-FS, mini-ImageNet), the meta and adaptation learning rates are given in Table D.5. For those experiments, the added linear network consists of 256-128-64-64 hidden units. To simplify computations on mini-ImageNet, we downsample images to 32x32 pixels.

Nonlinear Models All methods in Tables 1, 2, and 3 are tested on the standard 5-ways 5-shots setting. For Omniglot and CIFAR-FS, the networks are the original CNN from Finn et al. (2017) with 32 hidden units, while for mini-ImageNet we used 64 hidden units. We use Adam to optimize the networks, with default values of (0.9, 0.999) for β and 1e-8 for ϵ, and process tasks by batches of 16. Meta and adaptation learning rates for all methods are reported in Tables D.6, D.7.

Table D.6: **Adaptation learning rates** for the two-layer CNN experiments in the main text and the Suppl. Material

	MAML w/					ANIL w/				
Dataset	(null)	LinNet	MSGD	MC	KFO	(null)	LinNet	MSGD	MC	KFO
Omniglot	0.08	0.05	0.5	0.5	0.1	0.5	0.5	0.05	0.05	0.1
CIFAR-FS	0.07	0.01	0.7	0.7	0.1	0.5	0.5	0.1	0.1	0.1
mini-ImageNet	0.001	0.001	0.1	0.05	0.001	0.5	0.5	0.1	0.1	0.1

Table D.7: **Meta learning rates** for the two-layer CNN experiments in the main text and the Suppl. Material

	MAML w/					ANIL w/				
Dataset	(null)	LinNet	MSGD	MC	KFO	(null)	LinNet	MSGD	MC	KFO
Omniglot	0.003	0.003	0.0005	0.0005	0.001	0.001	0.003	0.001	0.001	0.001
CIFAR-FS	0.003	0.003	0.001	0.003	0.003	0.002	0.001	0.001	0.001	0.001
mini-ImageNet	0.0005	0.0005	0.001	0.001	0.0001	0.001	0.001	0.001	0.001	0.001

References

Arnold, S. M. R., Mahajan, P., Datta, D., Bunner, I., and Zarkias, K. S. (2020). learn2learn: A library for meta-learning research.

Balcan, M.-F., Khodak, M., and Talwalkar, A. (2019). Provable guarantees for gradient-based meta-learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 424–433. PMLR.

Baxter, J. (2000). A model of inductive bias learning. *J. Artif. Intell. Res.*, 12:149–198.

Behl, H., Baydin, A. G., and Torr, P. H. (2019). Alpha maml: Adaptive model-agnostic meta-learning. In *6th ICML Workshop on Automated Machine Learning, Thirty-sixth International Conference on Machine Learning (ICML 2019), Long Beach, CA, US.*

Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 969 vol.2–.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. (2019). Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations.*

Caruana, R. (1997). Multitask learning. *Mach. Learn.*, 28(1):41–75.

Denevi, G., Ciliberto, C., Grazzi, R., and Pontil, M. (2019). Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, pages 1566–1575. PMLR.

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Finn, C. and Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations.*

Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R. (2020). Meta-learning with warped gradient descent. In *International Conference on Learning Representations.*

Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E., and Whiteson, S. (2018). Dice: The infinitely differentiable monte carlo estimator. In *International Conference on Machine Learning*, pages 1524–1533.

Gidel, G., Bach, F., and Lacoste-Julien, S. (2019). Implicit regularization of discrete gradient dynamics in linear neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations.*

Guiroy, S., Verma, V., and Pal, C. (2019). Towards understanding generalization in Gradient-Based Meta-Learning. *arXiv preprint arXiv:1907.07287.*

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science Engineering*, 9(3):90–95.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665.

Lee, Y. and Choi, S. (2018). Gradient-based meta-learning with learned layerwise metric and subspace. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2927–2936, Stockholmsmässan, Stockholm Sweden. PMLR.

Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-SGD: Learning to learn quickly for Few-Shot learning. *arXiv preprint arXiv:1707.09835*.

Nichol, A., Achiam, J., and Schulman, J. (2018). On First-Order Meta-Learning algorithms. *arXiv preprint arXiv:1803.02999*.

Park, E. and Oliva, J. B. (2019). Meta-curvature. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, High-Performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alche Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2020). Rapid learning or feature reuse? towards understanding the effectiveness of {maml}. In *International Conference on Learning Representations*.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. (2019). ProMP: Proximal meta-policy search. In *International Conference on Learning Representations*.

Saunshi, N., Zhang, Y., Khodak, M., and Arora, S. (2020). A sample complexity separation between non-convex and convex meta-learning. In *International Conference on Machine Learning*, pages 8512–8521. PMLR.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2019). A mathematical theory of semantic development in deep neural networks. *Proc. Natl. Acad. Sci. U. S. A.*, 116(23):11537–11546.

Schmidhuber, J. (1987). *Evolutionary Principles in Self-Referential Learning*. PhD thesis.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

Van Rossum, G. and Drake, Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Vanschoren, J. (2019). Meta-Learning. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Automated Machine Learning: Methods, Systems, Challenges*, pages 35–61. Springer International Publishing, Cham.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods*, 17(3):261–272.

Wu, Y., Ren, M., Liao, R., and Grosse., R. (2018). Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*.