DECOUPLING ADAPTATION FROM MODELING WITH META-OPTIMIZERS FOR META LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Meta-learning methods, most notably Model-Agnostic Meta-Learning (Finn et al., 2017) or MAML, have achieved great success in adapting to new tasks quickly, after having been trained on similar tasks. The mechanism behind their success, however, is poorly understood. We begin this work with an experimental analysis of MAML, finding that deep models are crucial for its success, even given sets of simple tasks where a linear model would suffice on any individual task. Furthermore, on image-recognition tasks, we find that the early layers of MAML-trained models learn task-invariant features, while later layers are used for adaptation, providing further evidence that these models require greater capacity than is strictly necessary for their individual tasks. Following our findings, we propose a method which enables better use of model capacity at inference time by separating the adaptation aspect of meta-learning into parameters that are only used for adaptation but are not part of the forward model. We find that our approach enables more effective meta-learning in smaller models, which are suitably sized for the individual tasks.

1 INTRODUCTION

Meta-learning or *learning to learn* is an appealing notion due to its potential in addressing important challenges when applying machine learning to real-world problems. In particular, learning from prior tasks but being able to to adapt quickly to new tasks improves learning efficiency, model robustness, etc. A promising set of techiques, Model-Agnostic Meta-Learning (Finn et al., 2017) or MAML, and its variants, have received a lot of interest (Nichol et al., 2018; Lee & Choi, 2018; Grant et al., 2018). However, despite several efforts, understanding of how MAML works, either theoretically or in practice, has been lacking (Finn & Levine, 2018; Fallah et al., 2019).

For a model that meta-learns, its parameters need to encode not only the common knowledge extracted from the tasks it has seen, which form a task-general inductive bias, but also the capability to adapt to new test tasks (similar to those it has seen) with task-specific knowledge. This begs the question: *how are these two sets of capabilities represented in a single model and how do they work together?*

In the case of deep learning models, one natural hypothesis is that while knowledge is represented distributedly in parameters, they can be localized – for instance, lower layers encode task-general inductive bias and the higher layers encode adaptable task-specific inductive bias. This hypothesis is consistent with one of deep learning's advantages in learning representations (or feature extractors) using its bottom layers.

Then we must ask, *in order for a deep learning model to meta-learn, does it need more depth than it needs for solving the target tasks?* In other words, is having a large capacity to encode knowledge that is unnecessary *post*-adaptation the price one has to pay in order to be adaptable? Is there a way to have a smaller (say, less deep) meta-learnable model which still adapts well? This question is of both scientific interest and practical importance – a smaller model has a smaller (memory) footprint, faster inference and consumes less resources.

In this work, through empirical studies on both synthetic datasets and benchmarks used in the literature, we investigate these questions by analyzing how well different learning models can meta-learn and adapt. We choose to focus on MAML due to its popularity.

Our observations suggest depth is indeed necessary for meta-learning, despite the tasks being solvable using a shallower model. Thus, applying MAML to shallower models does not result in successful meta-learning models that can adapt well. Moreover, our studies also show that higher layers are responsible more for adapting to new tasks while the lower layers are responsible for learning task-general features.

Our findings prompt us to propose a new method for meta-learning. The new approach introduces a meta-optimizer which learns to guide the (parameter) optimization process of a small model. The small model is used for solving the tasks while the optimizer bears the burden of extracting the knowledge of how to adapt. Empirical results show that despite using smaller models, the proposed algorithm with small models attains similar performance to larger models which use MAML to meta-learn and adapt.

We note that a recent and concurrent work to ours addresses questions in this line of inquiry (Raghu et al., 2019). They reach similar conclusions through different analysis and likewise, they propose a different approach for improving MAML. We believe our work is complementary to theirs.

2 Related Work

Meta-learning, or learning-to-learn, is a vibrant research area with a long and rich history, lying at the intersection of psychology (Maudsley, 1980; Biggs, 1985), neuroscience (Hasselmo & Bower, 1993), and computer science(Schmidhuber, 1987; Thrun & Pratt, 1998) Of particular interest to this manuscript is the line of work concerned with optimization-based meta-learning (OBML) algorithms in the few-shot regime, of which MAML is a particular instance. (Finn & Levine, 2018; Finn et al., 2017; Finn, 2018)

While popular in practical applications, relatively few works have analysed the convergence and modelling properties of those algorithms. Finn & Levine (2018) showed that, when combined with deep architectures, OBML is able to approximate arbitrary meta-learning schemes. Fallah et al. (2019) recently provided convergence guarantees for MAML to approximate first-order stationary points for non-convex loss surfaces, under some assumptions on the availability and distribution of the data. Other analyses (empirical or theoretical) have attempted to explain the generalization ability of OBML (Guiroy et al., 2019; Nichol et al., 2018), the bias induced by restricting the number of adaptation steps (Wu et al., 2018), or the effect of higher-order terms in the meta-gradient estimation(Foerster et al., 2018; Rothfuss et al., 2019)

Closely related to our proposed methods are works attempting to improve the adaptation mechanisms of OBML. Meta-SGD (Li et al., 2017) meta-learns per-parameter learning rates, while Alpha MAML (Behl et al., 2019) adapts those learning rates during adaptation via gradient-descent. Meta-Curvature (Park & Oliva, 2019) propose to learn a Kronecker-factored pre-conditioning matrix to compute fast-adaptation updates. Their resulting algorithm is a special case of one of our methods, where the linear transformation is not updated during adaptation. Another way of constructing pre-conditioning matrices is to explicitly decompose all weight matrices of the model in two separate components, as done in T-Nets (Lee & Choi, 2018). The first component is only updated via the evaluation loss, while the second is also updated during fast-adaptation. Warped Gradient Descent (Flennerhag et al., 2019) further extends T-Nets by allowing both components to be non-linear functions. Instead of directly working with gradients, (Chebotar et al., 2019; Xu et al., 2018) suggest to directly learn a loss function which is differentiated during fast-adaptation and results in faster and better learning. Additionally, meta-optimizers have also been used for meta-descent (Sutton, 1981; Jacobs, 1988; Sutton, 1992b;a; Schraudolph, 1999). They can be learned during a pre-training phase (Andrychowicz et al., 2016; Li & Malik, 2017a;b; Metz et al., 2019b;a;c; Wichrowska et al., 2017) or online (Kearney et al., 2018; Jacobsen et al., 2019; Ravi & Larochelle, 2017).

Our work differentiates from the the above by diagnosing and attempting to address the entanglement between modelling and adaptation in the meta-learning regime. We uncover a failure mode of MAML with linear and smaller models, and propose an effective solution in the form of expressive meta-optimizers.

3 ANALYSIS OF MAML

3.1 BACKGROUND

In MAML and its many variants (Lee & Choi, 2018; Nichol et al., 2018; Li et al., 2017), we have a model whose parameters are denoted by θ. We would like to optimize θ such that the resulting model can adapt to new and unseeen tasks fast. To this end, we are given a set of (meta)training tasks, indexed by τ. For each such task, we associate with a loss $\mathcal{L}_\tau(\theta)$.

Distinctively, MAML minimizes the expected task loss after an *adaptation* phase, consisting of a few steps of gradient descent from the model's current parameters. Since we do not have access to the target tasks to which we wish to adapt to, we use the expected loss over the training tasks,

$$\mathcal{L}^{\text{META}} = \mathop{\mathbb{E}}_{\tau \sim p(\tau)} \left[\mathcal{L}_\tau \left(\theta - \alpha \nabla \mathcal{L}_\tau(\theta) \right) \right] \tag{1}$$

where the expectation is taken with respect to the distribution of the training tasks. α is the learning rate for the adaptation phase. The right-hand-side uses only one step gradient descent such that the aim is to adapt *fast*: in one step, we would like to reduce the loss as much as possible. In practice, a few more steps are often used.

3.2 ANALYSIS

Shallow models can be hard to meta-learn Many intuitive explanations for why MAML works exist. One appealing suggestion is that the minimizer of the meta-learning loss $\mathcal{L}^{\text{META}}$ is chosen in such a way that it provides a very good initialization for the adaptation phase; however, if the model is shallow such that $\mathcal{L}_\tau$ is convex in its parameters, then any initialization that is good for fast adapting to one subset of tasks could be bad for another subset of tasks since all the tasks have precisely one global minimizer and those minimizers can be arbitrarily far from each other. When the test tasks are distributed similar to the training tasks, the ideal initialization point has to be the "mean" of the minimizers of the training tasks — the precise definition of the mean is not important, as we will see below.

We illustrate a surprising challenge by studying MAML on a synthetic dataset and the Omniglot task (Lake et al., 2015). Specifically, for the former study, we construct a set of binary classification tasks by first randomly sampling datapoints $w \in \mathbb{R}^{100}$ from a standard Gaussian and use each of them to define a linear decision boundary of a binary classification task. We assume the boundaries pass through the origin and we sample training, validation and testing samples by randomly sampling data points from both sides of the decision boundaries.

By construction, a linear model such as logistic regression is sufficient to achieve very high accuracy on any task. But can MAML learn a logistic regression model from a subset of training tasks that is able to adapt quickly to the test tasks?

Note that due to the random sampling of the training tasks, the average of the minimizers (ie, the samples from the Gaussian distribution) is the origin. Likewise, for a set of test tasks randomly sampled the same way, the origin provides the best initialization by not favoring any particular task.

Fig. 1 reports the 1-step post-adaptation accuracy on the test tasks for the meta-learned logistic regression model. Surprisingly, the model fails to perform better than chance. Despite the simplicity of the tasks, logistic regression models are unable to find the origin as an initialization that adapts quickly to a set of test tasks that are similar to the training tasks.

Fig. 1 also reports how *being deep* can drastically change the behavior of MAML. There, we add a 4-layer linear network (LinNet) to the logistic regression model (before the sigmoid activation). Note that while the model has the same representational capacity as a linear logistic regression, it is overparameterized and has many local optimizers. As such, MAML can train this model such that its 1-step adaptation accuracy reaches 92% on average.

We observe the same phenomena on meta-learning with MAML on the Omniglot dataset (details of the dataset are given in the Section 5). The shallow logistic regression model achieves 45% accuracy on average (for the 5-way classification tasks) after 2-step adaptation from the meta-learned initialization. However, with a linear network, the adapted model achieves significantly higher accuracy – 70% on average, while having the same modeling capacity as the logistic regression.

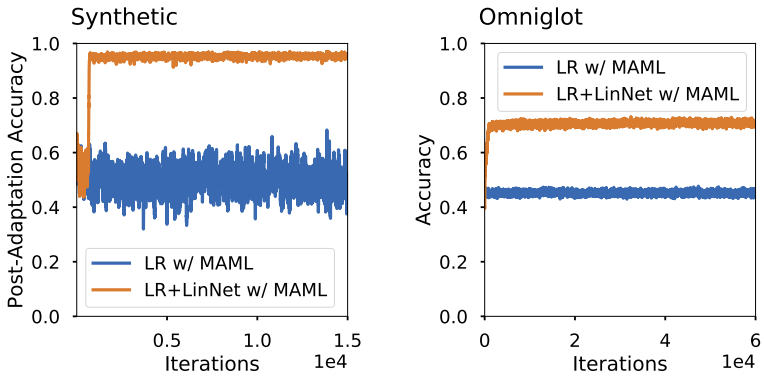


Figure 1: Meta learning results on two datasets: Synthetic binary classification (left) and Omniglot (right). The models to be meta-learned are shallow ones (logistic regression models for binary classification and multinomial logistic regression) and deep ones – the shallow models with linear networks (LinNet) added to them. While the shallow models do not adapt and achieve chance level accuracies on average, the overparameterized versions attain significantly better results after adaptation.

In summary, these experiments suggest that even for tasks that are solvable with shallow models, a model needs to have enough depth in order to be meta-learnable and to adapt. We postpone the description of our experiments on nonlinear models to section 5, where we also show that having sufficient depth is crucial for models to be meta-learnable, even when the tasks require fewer layers.

A natural question arises: if *being deep* is so important for meta-learning on even very simple tasks, what different roles, if any, do different layers of a deep network play?

Depth enables task-general feature learning and fast adaptation We hypothesize that for deep models meta-learned with MAML, lower layers learn task-invariant features while higher layers are responsible for fast-adaptation.

To examine this claim, we meta-train a model consisting of four convolutional layers (C1 - C4) and a final fully-connected layer (FC) on Omniglot (Lake et al., 2015) and CIFAR-FS (Bertinetto et al., 2019). (Experimental setups are detailed in Section 5.) Once the model has finished meta-training, we perform a layer-wise ablation to study each layer's effect on adaptation. In particular, we iterate over each layer and perform two sets of experiments. In the first, we freeze the weights of the layer such that it does not get updated during fast-adaptation – we call it *freezing only this*. In the second experiment, we freeze all layers but the layer such that this layer is updated during fast-adaptation – we call it *adapting only this*. The left two plots in Figure 2 report the average accuracy over 100 testing tasks from both datasets.

We observe that freezing only the first few lower layers (C1-C3) does not cause noticeable degradation to the post-adaptation accuracy. In fact, as long as the last convolutional layer (C4) is not frozen, post-adaptation accuracy remains unaffected. This indicate that C1-C3 provide information that is task-invariant, while C4 is crucial for adaptation.

This does not mean FC is not important — since adapting C4 requires gradients passing through the FC layer, it cannot be arbitrary. In fact, in the rightmost plot of the figure, C1-C3 are held fixed during adaptation. While C4 and FC are allowed to adapt, and FC is perturbed with noise. When the noise is strong, the performance degrades significantly. Thus, we conclude that both C4 and FC play important roles in the mechanism for fast adaptation. We note that the recent work by Raghu et al. (2019) concurently reached similar conclusions on the *mini*-ImageNet dataset, using feature similarity-based analyses of the model's representations.

These observations highlight a fundamental issue: the property of being meta-learnable entails more model capacity than being learnable for a specific task. Thus, *MAML can fail on models that lack the capacity to encode both task-general features and adaptation information, even when the models themselves are powerful enough to perform well on each of the tasks used for the meta-learning procedure*. For example, with linear models (e.g. logistic regression), the parameters are forced to

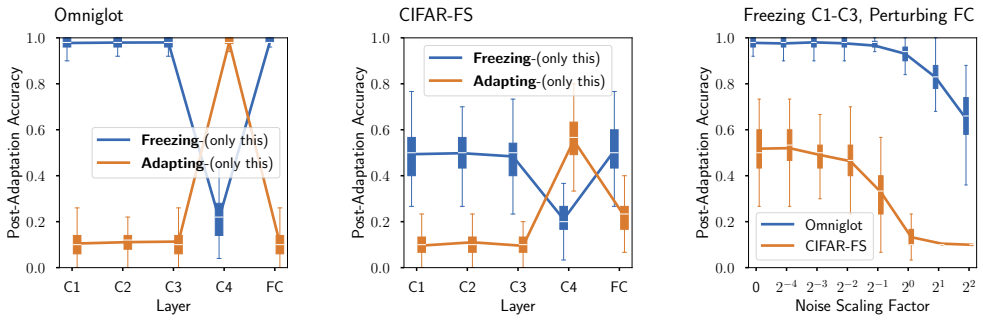


Figure 2: Layer-wise study of each layer's effect on adaptation, in the left and the middle plots. The rightmost plot shows that both C4 and FC have to work together to adapt. See texts for details.

overlap and serve both modelling and adaptation purposes. However, as soon as the models are over-parameterized, the extra layers enable meta-learnability. In section 5, we show that this observation also applies to nonlinear models where MAML-trained models quickly lose their performance when the number of layers is reduced.

4 Meta-Optimizer for Meta Learning

The previous section has shown that MAML, when applied to deep learning models, meta-learns both task-general features and task-specific adaption parameters at the same time. Since both are represented in a single model where strong dependencies exist between the lower layers and the higher layers, the model needs to be large enough and all its parameters have to be used to solve the test tasks. In some cases, the model has a modelling capacity that is bigger than what the test tasks require. For a model where linear network layers exist, the layers can be collapsed after adaptation so that the actual model used for inference on teh test tasks is small. However, it is not clear how to do so for typical deep models where nonlinearities prevent collapsing the model into smaller ones.

Can we have a different adaptation mechanism such that a smaller model, whose modelling capacity is still adequate for the test tasks, can be adapted to find the minimizers of its loss on the aforementioned tasks?

In this section, we describe a new approach of learning meta-optimizers for meta-learning. The meta-optimizer aims to separate modelling from adaptation. The meta-optimizer transforms the parameter update process for the model so the model can converge fast to its minimizer. How to transform is, however, learned from meta-training tasks. Because of this separation, the model for the task does not have to know how to adapt; it only needs to have enough capacity (roughly, big enough to represent the target task).

We note that classical tools could be used to compress a large model for a given task. However, in the scenario where meta-learning is often used, such tools are unlikely effective. For example, distillation requires a lot of labeled data from the target task, which is not suitable for few-shot learning tasks. Pruning often degrades performance.

4.1 Learnable Meta-Optimizers

A meta-optimizer – or learnable optimizer – is a parameterized function U_ξ defining the model's parameter updates. For example, a linear meta-optimizer might be defined as:

$$U_\xi(g) = Ag + b, \tag{2}$$

where $\xi = (A, b)$ is the set of parameters of the linear transformation. The objective is to jointly learn the model and optimizer's parameters in the hope of accelerating the minimization of $\mathcal{L}_\tau$. Concretely, we alternate between model and optimizer updates such that their values at adaptation step $t + 1$ are given by:

$$\theta_{t+1} = \theta_t - U_{\xi_t}\left(\nabla_{\theta_t}\mathcal{L}_\tau\left(\theta_t\right)\right) \tag{3}$$
$$\xi_{t+1} = \xi_t - \alpha\nabla_{\xi_t}\mathcal{L}_\tau\left(\theta_{t+1}\right). \tag{4}$$

While this change to the classical optimization pipeline might seem innocuous at first, it bears its own set of computational challenges. First, it is common for modern machine learning models to have millions of parameters. In this case, simply storing the optimizer's parameters becomes infeasible even for the simplest of models, such as the linear one outlined above. Because of this dimensionality issue most of the current literature considers a per-parameter independent update function, thus greatly limiting the expressivity of the meta-optimizer. We propose to address the issue of parameter dimensionality via factorizations of the optimizer's parameters, which we detail in the following subsection.

4.2 Factorized Meta-Optimizers

To tackle the issue of parameter dimensionality we propose to factorize the optimizer's parameters. To demonstrate this, let us revisit the linear optimizer example above: we assume $g \in \mathbb{R}^k$ to be the vectorization of a matrix gradient $G \in \mathbb{R}^{n\times m}$, such that $\text{vec}(G) = g$ and $m \cdot n = k$. By expressing $A = R^\top \otimes L$ as the Kronecker product of small matrices $R \in \mathbb{R}^{m\times m}$ and $L \in \mathbb{R}^{n\times n}$, the linear update above can be efficiently computed as:

$$U_\xi(g) = \text{vec}\left(L \cdot G \cdot R\right) + b, \tag{5}$$

where $b \in \mathbb{R}^k$ and $\xi = (L, R, b)$. In the best case scenario where $m = n = \sqrt{k}$, the above factorization requires $\mathcal{O}\left(k\right)$ memory as opposed to $\mathcal{O}\left(k^2\right)$ for the non-factored case. Similarly, the matrix-product takes $\mathcal{O}\left(k\sqrt{k}\right)$ time complexity, while the non-factored version takes $\mathcal{O}\left(k^2\right)$.

Note that this linear transformation can be used as a building block to implement more expressive and non-linear meta-optimizers. For example, a fully-connected network meta-optimizer U_ξ is the composition of linear transformations U_{W_i} interleaved with activation function σ. If the weight matrices $(W_1, \ldots, W_h)$ of the network admit a Kronecker-factorization $W_i = R_i^\top \otimes L_i$, the fully-connected meta-optimizer is imlemented as:

$$U_\xi(g) = U_{W_h} \circ \cdots \circ \sigma \circ U_{W_1} \circ g \tag{6}$$
$$= \text{vec}\left(L_h\sigma\left(\ldots\sigma\left(L_1 \cdot G \cdot R_1\right)\ldots\right)R_h\right), \tag{7}$$

where $\xi = (W_1, \ldots, W_h)$. Such a scheme can be used to implement arbitrary meta-optimizers – such as convolutional or recurrent ones – so long as the architecture involves a composition of linear maps.

We emphasize that the choice of a Kronecker factorization is arbitrary; many matrix decompositions work equivalently well and 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 prefered the Kronecker decomposition over alternatives for three reasons: (1) the computational and memory cost of the Kronecker-product are acceptable, $R^\top \otimes L$ 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.

Table 1: Accuracies and standard deviation of Meta-Learning of Linear Models

Dataset	LR w/ MAML	LR + LINNET w/ MAML	LR W/ KLIN	LR W/ KFC
Synthetic	51.38% $\pm$ 3.91	95.77% $\pm$ 0.41	**99.00% $\pm$ 0.08**	**98.98% $\pm$ 0.10**
Omniglot	46.50% $\pm$ 2.80	**71.31% $\pm$ 4.52**	63.88% $\pm$ 3.93	**71.56% $\pm$ 4.40**

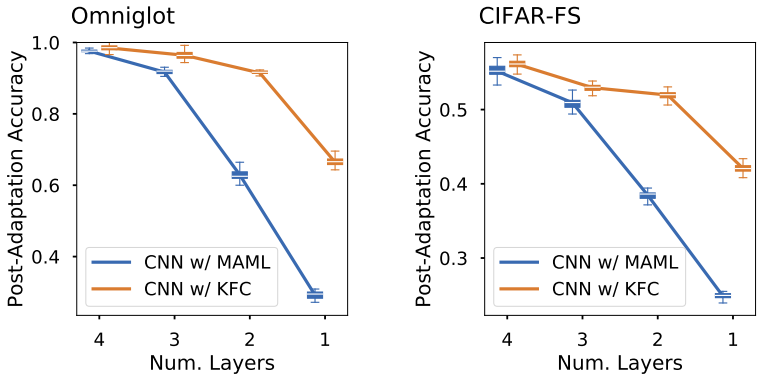


Figure 3: The effect of the number of convolutional layers on adaptation performance. While the model becomes smaller, the performance drops. However, our approach degrades in a slower pace, as the adaptation is shared by both the model and the meta-optimizer, thus leaving more freedom for the model to represent the tasks.

5 EXPERIMENTS

In our experiments, we complement our empirical studies and analysis from Section 3 with further results and address the following question: can we meta-learn smaller (or shallower) models that perform as well as larger (or deeper) ones? To this end, we apply the proposed approach of learning meta-optimizers to the example synthetic dataset, as well as popular benchmark datasets: Omniglot (Lake et al., 2015), *mini*-ImageNet (Ravi & Larochelle, 2017), and CIFAR-FS (Bertinetto et al., 2019). All hyper-parameters and additional experimental details are available in Appendix A.1.

Unless otherwise noted, all models are trained using MAML. For models that use learnable deep optimizers, we meta-learn both model and optimizer parameters. Similarly, both sets of parameters are adapted during fast-adaptation.

To differentiate, we add the name of the optimizer to the model. For example, LR W/ KLIN corresponds to a logisitic regression model with the Kronecker-factorized linear transformation from the previous section. Similarly, W/ KFC indicates a model optimized by **K**ronecker-factored **F**ully-**C**onnected network in the previous section.

5.1 META-LEARNING LINEAR MODELS

We adopt the same setting as in section 3 where we study the synthetic data and the Omniglot dataset. For the Omniglot dataset, we consider the 1-shot, 5-way setup of Finn et al. (2017) and let the model adapt for 2 steps. Recall that the logistic regression model does not meta-learn while overparameterized logistic regression models (ie, models with linear networks) do.

Table 1 reports the final post-adaptation testing accuracies and their standard deviations. As can be observed, while logistic regression (LR) cannot be meta-learned, overparameterized models (LINNET) as well as LR with our meta-optimizers (LR W/ KLIN and LR W/ KFC) can all meta-learn while our approaches lead to similar or better results compared to LinNet.

Note that since linear networks can always be collapsed into a single weight matrix and bias term, any increase in performance is the consequence of better adaptation, rather than better modelling.

Table 2: Post-adaptation accuracies by different meta-optimizers adapting a small CNN

		Meta-Learner Architecture			
Task	UPPER BOUND	W/ MAML	W/ MSGD	W/ MC	W/ KFC
Omniglot	95.09% $\pm$ 2.32	60.94% $\pm$ 1.54	69.25% $\pm$ 0.81	85.70% $\pm$ 0.91	**91.13% $\pm$ 0.93**
CIFAR-FS	63.94% $\pm$ 3.89	38.44% $\pm$ 0.87	45.96% $\pm$ 1.73	**52.89% $\pm$ 1.43**	**52.94% $\pm$ 1.33**
mini-ImageNet	58.45% $\pm$ 5.58	18.55% $\pm$ 0.46	26.59% $\pm$ 1.43	**29.74% $\pm$ 0.74**	27.40% $\pm$ 0.60

5.2 NON-LINEAR MODELS

We examine whether our meta-optimizers can meta-learn successfully with smaller but deep nonlinear models to match bigger deep models. We focus on two datasets: the Omniglot where the setting is 10-way classification with 5-shots and 4 adaptation steps, using the original 4-layer convolutional network (CNN) of Finn et al. (2017), and the CIFAR-FS dataset (Bertinetto et al., 2019), doing 10-way classification with 3-shots and 2 adaptation steps. The model is a CNN similar to the one for the Omniglot, but only uses 32 hidden units as opposed to 64.

We report numerical results in detail in Table A1 in the Appendix, which are graphically presented in Fig. 3. We vary the number of convolutional layers. As the number of the layers decrease, the adaptation performance by both MAML and our approach (KFC) decrease. However, our approach has a much slower degradation rate. In fact, our approach is generally able to adapt a smaller model to the same level of performance as a MAML-trained model with an additional layer.

5.3 COMPARISON TO OTHER META-OPTIMIZERS

We also compare our approach of learning meta-optimizers to other approaches, notably Meta-SGD (Li et al., 2017) and Meta-Curvature (Park & Oliva, 2019). Both of these methods attempt to approximate the expected task-loss landscape curvature and can be seen as ablations of our methods; Meta-Curvature corresponds to `KLin` without adapting the optimizer, while Meta-SGD approximates Meta-Curvature with a diagonal matrix.

To ease the computational burden, we use a smaller CNN with 2 convolutional layers (SCNN) model for adaptation. As an upper bound on adaptation performance, we train the model on individual tasks to convergence and report the average final accuracy. Additionally, we include results on the *mini*-ImageNet dataset for all methods, in the 10-way 1-shot with 5 adaptation steps setting (Ravi & Larochelle, 2017).

Table 2 reports post-adaptation testing accuracies. All methods of learning optimizers are able to adapt better than SCNN w/ MAML. In particular, our approachs perform the best and comes closest to approaching the upper bound of performance (SCNN).

6 CONCLUSION

We introduce our approach by analyzing the success and failure modes of optimization-based meta-learning methods. Namely, we find that, when successful, these methods tend to learn task-general features in early layers and adaptable parameters/update functions in the later layers. Moreover, we find that this learning fails when model size is reduced, indicating that optimization-based meta-learning methods rely on the ability to encode task-general features and/or adaptable parameters, even when the model itself is adequate for learning on the individual tasks. As such, we introduce our method for decomposing modelling from adaptation using factored meta-optimizers. These meta-optimizers enable the forward model to use more capacity on learning task-specific features, while the expressiveness of their updates allows the forward model to adapt quickly to different tasks. We find that our approach is able to enable successful meta-learning in models that do not work with traditional optimization-based meta-learning methods.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3981–3989. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6461-learning-to-learn-by-gradient-descent-by-gradient-descent.pdf.

Harkirat Singh Behl, Atılım Güneş Baydin, and Philip H S Torr. Alpha MAML: Adaptive Model-Agnostic Meta-Learning. *arXiv preprint arXiv:1905.07435*, May 2019.

Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7.

J B Biggs. THE ROLE OF METALEARNING IN STUDY PROCESSES. *Br. J. Educ. Psychol.*, 55 (3):185–212, November 1985.

Yevgen Chebotar, Artem Molchanov, Sarah Bechtle, Ludovic Righetti, Franziska Meier, and Gaurav Sukhatme. Meta-Learning via learned loss. *arXiv preprint arXiv:1906.05374*, June 2019.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of Gradient-Based Model-Agnostic Meta-Learning algorithms. *arXiv preprint arXiv:1908.10400*, August 2019.

Chelsea Finn. *Learning to Learn with Gradients*. PhD thesis, UC Berkeley, July 2018.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyjC5yWCW.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, 2017. PMLR.

Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. Meta-Learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, August 2019.

Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte carlo estimator. In *International Conference on Machine Learning*, pp. 1524–1533, 2018.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ_UL-k0b.

Simon Guiroy, Vikas Verma, and Christopher Pal. Towards understanding generalization in Gradient-Based Meta-Learning. *arXiv preprint arXiv:1907.07287*, July 2019.

M E Hasselmo and J M Bower. Acetylcholine and memory. *Trends Neurosci.*, 16(6):218–222, June 1993.

Robert A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Netw.*, 1 (4):295–307, January 1988.

Andrew Jacobsen, Matthew Schlegel, Cameron Linke, Thomas Degris, Adam White, and Martha White. Meta-descent for online, continual prediction. In *AAAI Conference on Artificial Intelligence*, 2019.

Alex Kearney, Vivek Veeriah, Jaden B Travnik, Richard S Sutton, and Patrick M Pilarski. TIDBD: Adapting temporal-difference step-sizes through stochastic meta-descent. *arXiv preprint arXiv:1804.03334*, April 2018.

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

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2927–2936, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lee18a.html.

Ke Li and Jitendra Malik. Learning to optimize. In *International Conference on Learning Representations*, 2017a. URL https://openreview.net/forum?id=ry4Vrt5gl.

Ke Li and Jitendra Malik. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, March 2017b.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for Few-Shot learning. *arXiv preprint arXiv:1707.09835*, July 2017.

Donald B Maudsley. A THEORY OF META-LEARNING AND PRINCIPLES OF FACILITATION: AN ORGANISMIC PERSPECTIVE. 1980.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=HkNDsiC9KQ.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4556–4565, Long Beach, California, USA, 09–15 Jun 2019b. PMLR. URL http://proceedings.mlr.press/v97/metz19a.html.

Luke Metz, Niru Maheswaranathan, Jonathon Shlens, Jascha Sohl-Dickstein, and Ekin D Cubuk. Using learned optimizers to make models robust to input noise. *arXiv preprint arXiv:1906.03367*, June 2019c.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning algorithms. *arXiv preprint arXiv:1803.02999*, March 2018.

Eunbyung Park and Junier B Oliva. Meta-Curvature. *arXiv preprint arXiv:1902.03356*, February 2019.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. *arXiv preprint arXiv:1909.09157*, September 2019.

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

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal meta-policy search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkxXCi0qFX.

Juergen Schmidhuber. *Evolutionary Principles in Self-Referential Learning*. PhD thesis, 1987.

Nicol N Schraudolph. Local gain adaptation in stochastic gradient descent. 1999.

Rich Sutton. Adaptation of learning rate parameters. *Goal Seeking Components for Adaptive Intelligence: An Initial Assessment*, 1981.

Richard S Sutton. Gain adaptation beats least squares. In *Proceedings of the 7th Yale workshop on adaptive and learning systems*, volume 161168, 1992a.

Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pp. 171–176, 1992b. URL http://www.aaai.org/Library/AAAI/1992/aaai92-027.php.

Sebastian Thrun and Lorien Pratt (eds.). *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3751–3760, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/wichrowska17a.html.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1MczcgR-.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2396–2407. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7507-meta-gradient-reinforcement-learning.pdf.

A APPENDIX

A.1 EXPERIMENTAL DETAILS AND HYPER-PARAMETERS

The following paragraphs describe the data-generation process, model architectures, and learning hyper-parameters for experiments in Section 5. Meta-optimizers's parameter matrices are always initialized to the identity.

A.1.1 LINEAR MODEL EXPERIMENTS

Synthetic Binary Classification The task decision boundaries $w \sim \mathcal{N}(0.0, I_{100})$ are sampled from standard Gaussian. Each one of the 1,000 data point $x \sim \mathcal{N}(0.0, I_{100})$ is also sampled from the standard Gaussian, and is assigned a label from $\{-1, 1\}$ based on the sign of $w^\top x$. All models are trained for 30,000 iterations of stochastic gradient descent (without momentum) so as to minimize the 1-step MAML loss $\mathcal{L}^{\text{MAML}}$. The task loss $\mathcal{L}(\theta)$ is the binary cross-entropy. Each stochastic gradient is estimated on a single task. (i.e. the meta-batch size is 1.)

Due to the argument presented in the main text, any hyper-parameter setup will replicate the logistic regression (LR) results, but we used meta and adaptation learning rates of 0.01 and 0.5. The linear network (LR+LinNet) consists of 3 hidden layers of 64 units, and is trained with both meta- and fast-adaptation learning rates set to 1.9. Experiments using the Kronecker-factored linear meta-optimizer also use the same meta- and fast-adaptation learning rate, set to 3.33. The meta-optimizer itself consists of 10,001 parameters. $(n = 100, n = 1)$ The Kronecker-factored fully-connected meta-optimizer uses a single hidden layer with rectified linear-units (ReLU) activation functions, for a total of 20,002 parameters. $(m = 100, n = 1)$ It has both learning rates set to 3.9. None of the meta-optimizers use a bias term.

Omniglot Our Omniglot experiments exactly replicate the setup of Finn et al. (2017). Of the 1623 classes, we designate 1,200 for meta-training, 100 for validation, and 423 for testing. We then generate 20,000 training tasks, 600 validation/testing tasks, each consisting of five classes containing a single character image, possibly rotated by 90, 180, or 270 degrees. All models are trained using Adam for 60,000 iterations, an iteration consisting of 32 tasks for which the model is allowed 2 steps of fast-adaptaiton.

The logistic regression model uses a meta-learning rate of 0.0005 and an adaptation learning-rate set to 1.0. The linear network uses 4 hidden layers of 256, 128, 64, and 64 units and flattens the character images into a 784-dimensional vector before processing them. The meta-learning rate was set to 0.0005 and the adaptation learning rate to 0.08. For the KLin experiment (LR w/ KLin), we use a meta-learing rate of 0.003 and set the adaptation learning rate to 0.9. The KLin optimizer consists of 614k parameters, with $m = 784, n = 5$. The KFC meta-optimizer (LR w/ KFC) consists of 4 hidden layers (2.5M parameters) with ReLU activations, whose meta and adaptation learning rates are set to 0.001 and 0.01, respectively. Again, none of the meta-optimizer use biases.

A.1.2 NONLINEAR MODEL EXPERIMENTS

For the non-linear model experiments, we learn one KFC optimizer per layer of the model. Each KFC optimizer is based on the same architecture, but learns its own set of parameters. Moreover, we separate processing of the magnitude and direction of the model's gradient as follows: the gradient is initially scaled so as to have unit norm before being fed to the KFC network. Once the normalized gradient has been processed, it is rescaled by the initial normalizing constant times a learnable factor. (This learnable factor is initialized to 1.) We found this architecture to help in situations where the model is allowed several adaptation steps. For convolutional models, we flatten the height and width weight dimensions such that the optimizer is shared across filters of a same layer. (i.e. if the convolutional layer has shape NxCxHxW, we have $m = C, n = HW$.) Note that, again, none of the meta-optimizers use a bias term.

Omniglot The dataset replicates the processing of linear model experiments, detailed above. Instead of 5 ways and 1 shot, we use a 10 ways and 5 shots. All models are allowed 4 steps of fast adaptation.

The 4-layer CNN network (CNN w/ MAML) uses a meta-learning rate of 0.003, and an adaptation learning rate of 0.5. Its architecture replicates the one described in Finn et al. (2017). The 2-layer CNN (SCNN w/ MAML) uses the same first two layers and same input dimensionality to the FC layer as the 4-layer CNN, with meta and adapation learning rates set to 0.003 and 0.8, respectively. The 4-layer CNN has a total of 112,586 parameters, while the 2-layer only 38,474. We used the same learning rates for the Meta-SGD, Meta-Curvature, and KFC optimizers: a meta learning rate of 0.003, and an adaptation learning rate of 0.5. The KFC architecture consists of 4 layers, such that the meta-optimizers contains a total of 134,171 parameters for the 2-layer CNN model and 267,451 parameters for the 4-layer CNN.

CIFAR-FS We obtained the splits created by Bertinetto et al. (2019) and exactly reproduced their preprocessing setting for our experiments on CIFAR-FS. We also split the 100 classes in 64 training, 16 validation, and 20 testing classes and generate 20,000 training tasks, and 600 validation and testing tasks. We consider the 10-ways 3-shots setup with 2 fast-adaptation steps. As opposed to Bertinetto et al. (2019), our model closely resembles the one of our Omniglot experiments. Its main difference with the model described in prior work is the absence of max-pooling layers, and averaging of the last two convolutional feature dimensions before the fully-connected layer. Doing so allows us to conduct our ablation studies on the effect of our meta-optimizers with different number of convolutional layers, while keeping the input dimensionality of the fully-connected layer constant.

The 4- and 2-layer CNNs use meta- and fast-adaptation learning rates of 0.003 and 0.7, respectively. The 4-layer CNN has a total of 29,226 parameters, while the 2-layer only 10,602. Meta-SGD, Meta-Curvature, and KFC also use a 0.003 meta-learning rate, but the adaptation learning rate is decreased to 0.5. The KFC optimizer consists 4 layers with ReLU activations, for a total of 69,325 parameters for the 4-layer CNN model and 35,117 parameters for the 2-layer CNN.

***mini*-ImageNet** As for the Omniglot experiment, we replicate the *mini*-ImageNet setting and model from Finn et al. (2017). We consider the 10-ways, 1-shot and 5 fast-adaptation steps setting.

The 4-layer CNN and 2-layer CNN both use a meta-learning rate of 0.0005 and a fast-adaptation learning rate of 0.07. The 4-layer CNN has a total of 36,906 parameters, while the 2-layer only 18,282. The Meta-SGD, Meta-Curvature, and KFC meta-optimizer use the same meta-learning rate, but a smaller fast-adaptation learning rate set to 0.1 for Meta-SGD, 0.09 for Meta-Curvature, and 0.005 for KFC. For the 2-layer CNN model, the 4-layer KFC optimizer consists of approximately 2.60M parameters, while for the 4-layer CNN model it consists of 2.63M parameters.

A.2 Additional Experimental Results

In this section, we provide complementary results to the ones presented in the main text.

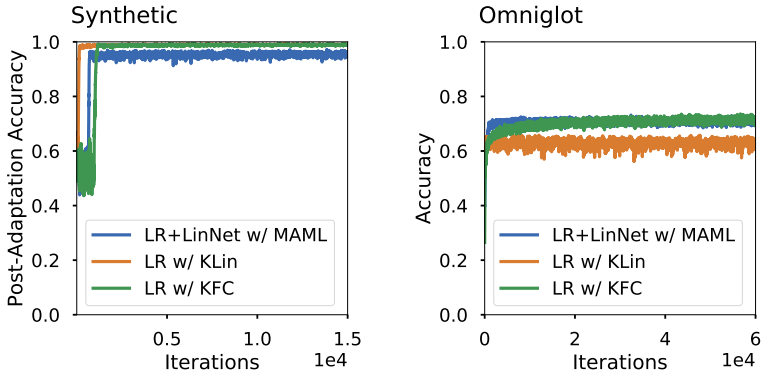
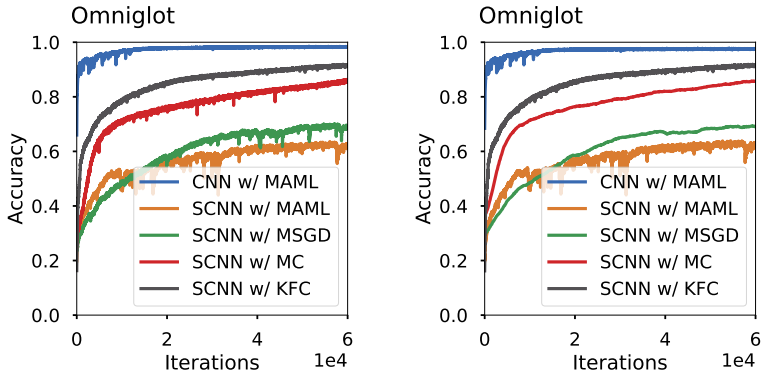


Figure A.1: Evolution of test accuracies along training for our methods with linear models on the synthetic data binary classification (left) and Omniglot (right).

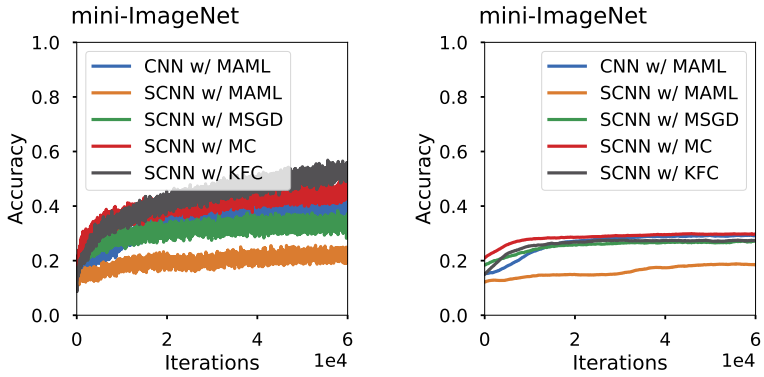
Table A1: Accuracies and standar deviation of Meta-Learning of Non-Linear Models

	Omniglot		CIFAR-FS	
Num. Layers	CNN	CNN w/ KFC	CNN	CNN w/ KFC
1	29.20% $\pm$ 1.13	**66.45% $\pm$ 1.42**	24.86% $\pm$ 0.47	**42.02% $\pm$ 0.52**
2	60.94% $\pm$ 1.54	**91.13% $\pm$ 0.93**	38.44% $\pm$ 0.87	**52.94% $\pm$ 1.33**
3	91.69% $\pm$ 0.70	**96.57% $\pm$ 0.86**	50.91% $\pm$ 0.80	**53.07% $\pm$ 0.58**
4	97.56% $\pm$ 0.45	**98.49% $\pm$ 0.41**	**55.03% $\pm$ 1.23**	**55.97% $\pm$ 0.59**



(a) Omniglot - Train Tasks (b) Omniglot - Test Tasks

Figure A.2: Evolution for the training (left) and testing (right) accuracies of the non-linear models on Omniglot.



(a) miniImageNet - Train Tasks (b) miniImageNet - Test Tasks

Figure A.3: Evolution for the training (left) and testing (right) accuracies of the non-linear models on *mini*-ImageNet .