Investigating parallelization of MAML*

Jan Bollenbacher[1,2][0000−0001−7180−1129], Florian Soulier[2],
Beate Rhein[2], and Laurenz Wiskott[1]

[1] Institut für Neuroinformatik, Ruhr-University Bochum, Bochum D-44801,
Germany laurenz.wiskott@ini.rub.de
[2] TH Köln (University of Applied Sciences), Köln D-50678, Germany
{jan.bollenbacher, beate.rhein}@th-koeln.de

Abstract. We propose a meta-learning framework to distribute Model-Agnostic Meta-Learning (DMAML), a widely used meta-learning algorithm, over multiple workers running in parallel. DMAML enables us to use multiple servers for learning and might be crucial if we want to tackle more challenging problems that often require more CPU time for simulation. In this work, we apply distributed MAML on supervised regression and image recognition tasks, which are quasi benchmark tasks in the field of meta-learning. We show the impact of parallelization w.r.t. wall clock time. Therefore, we compare distributing MAML over multiple workers and merging the model parameters after parallel learning with parallelizing MAML itself. We also investigate the impact of the hyperparameters on learning and point out further potential improvements.

Keywords: meta-learning · distributed learning · few-shot learning · parallel computing · MAML.

1 Introduction

In machine-learning, models are traditionally learned by applying a vast dataset on a specific task and train a model from scratch. Artificial learners perform poorly when only a small amount of data is available, or the task is changing, and they need to adapt.

In contrast, humans can abstract problems and effectively utilize prior knowledge to learn new skills quickly and can learn with just a few samples. Meta-learning is an emerging trend of research and tackles the problem of learning to learn. Meta-learning can be seen as a generalization of using experience and knowledge acquired earlier and is related to techniques for fine-tuning a model on a task or hyperparameter optimization.

Model-Agnostic Meta-Learning (MAML) is a popular and influential meta-learning algorithm. On a broad set of tasks, the MAML algorithm estimates a set of parameters, which is used as a starting point for fast adaptation so that new tasks can be learned quickly. MAML also has the benefit of being model-agnostic

[*] Accepted as a short-paper at 23rd International Conference on Discovery Science, will be published in LNCS

in a way, that it can be applied to many different instances of structurally similar tasks, solvable using gradient descent.

In this paper, we apply MAML in a distributed manner using multiple workers and compare it to a parallelization of MAML itself. We provide an empirical hyperparameter study and show the impact on the learning outcome. We can show that the benefit of parallelization highly depends on how we can sample the tasks for learning, and we recommend how to train MAML accordingly.

2 Related Work

MAML is a highly popular meta-learning algorithm for few-shot learning problems and achieves competitive performance on common benchmark few-shot learning problems [10, 20, 19, 18, 14, 16]. It is an optimization-based meta-learning algorithm learning the parameters of a task-specific classifier. Other approaches in the same family of meta-learning are [12, 7, 3, 11, 21]. Where, for example, in [21], a concept generator is learned parallel with a meta-learner, the meta-learner learns in a high-level concept space instead of the original representation.

Of all the algorithms in the family of optimization-based meta-learning algorithms, MAML is especially influential and inspired many direct extensions in literature recently [1, 6, 8, 17, 13, 2, 22]. These extensions critically rely on the core structure of the MAML algorithm, using an outer-learning loop (for meta-training) and an inner-learning loop (for task-specific adaptation). In [13, 2] an approach for implementing regimes to learn the inner learning rate α is proposed and in [15] in the inner-learning loop only the last layer of the model is adapted leading to less parameter updates. [22] divides the model parameters which are learned into *context* and *shared parameters* aiming to make MAML easier to parallelize and more interpretable.

In this work, we also aim to parallelize MAML by distributing learning over multiple workers. This approach is similar to [4], where the learning is performed by individual workers in a federated learning setting. Federated learning aims at training on heterogeneous datasets, whereas distributed learning aims on parallelizing computing power. In this work, we are performing an empirical investigation of MAML in the distributed learning setting.

3 Distributed Model-Agnostic Meta-Learning

3.1 Model-Agnostic Meta-Learning

The main goal of MAML introduced by [5] is to find model parameters f_θ which serve as a good starting point for training the model on new instances of tasks by applying just a few gradient steps.
We sample training tasks $\mathcal{T}_i$ from a distribution of tasks $p(\mathcal{T})$. A task $\mathcal{T}_i$ consists of a support and query set $\mathcal{T}_i = (\mathcal{D}_i^s, \mathcal{D}_i^q)$:

1. The support set is used to fit the model parameters for solving the task by using a high inner-loop learning rate α and

2. the query set is used for slow training the meta-model parameters using a lower meta-learning rate β.

In MAML, we apply learning by two nested loops:

Inner-learning loop: Using the support set $\mathcal{D}_i^s$ of a task $\mathcal{T}_i$, we calculate the gradient of the loss function $\mathcal{L}$ w.r.t. the parameters of f_θ and store these in γ_i (Eq. 1).

$$\gamma_i \leftarrow \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_i^s}(f_\theta) \tag{1}$$

Outer-learning loop: Using the Adam optimizer, we update the meta-model parameters θ in a way that the error loss using the query set $\mathcal{D}_i^q$ is minimized for all task specific values γ_i (Eq. 2).

$$\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i} \nabla_\theta \mathcal{L}_{\mathcal{D}_i^q}(f_{\theta-\gamma_i}) \tag{2}$$

The quantity of data points and the structure of the sets are highly dependent on the task domain. We evaluate the domains supervised regression and image recognition; the tasks are described in detail in Section 4.

3.2 Distributed Learning

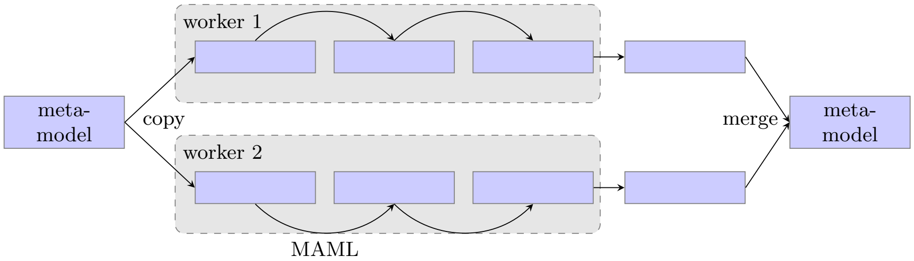

Fig. 1: One complete DMAML run: The central learner sends a copy of the meta-model to both workers. The workers perform 2 MAML steps and send the model parameters back to the central learner where they are merged.

We parallelize learning by distributing the learning over several workers. The central learner holds the meta-model parameters and sends a copy to the workers. Each worker is applying MAML steps to its model parameters in parallel. One MAML step is one update of the model parameters, as described in the previous Section 3.1. After the workers performed their MAML steps, the model parameters are sent back to the central learner. The central learner averages all parameters and starts over again. We call this procedure from copying the model parameters to the workers until averaging the adapted parameters one run. One Distributed Model-Agnostic Meta-Learning (DMAML) run is shown on Figure1.

The central learner receives the updated model parameters of each worker along with the following metrics: loss and accuracy. Therefore, strategies can be implemented on how to merge the model parameters. It is possible to drop model parameters of underperforming workers or give parameters of a well-performing worker a greater weight.

Workers can be run in parallel and are not increasing wall clock time used, while the consecutive MAML steps are executed sequentially and increase wall clock time linearly.

4 Experiments

The goal of our experimental evaluation is to answer the following questions:

1. Which impact does parallelization have on learning?
2. Which impact do the number of workers and MAML steps have, and is it beneficial to use more tasks during one MAML step?
3. Which parameters are best if tasks are either perfectly parallelized or serialized?

We compare our model to plain MAML, which is equivalent to DMAML using only one worker. In all of our experiments, we use TensorFlow 2 (without GPU support). All parameters used for training are listed in the Appendix B.

4.1 Regression Task

The task domain of tasks T_i is to perform a regression to data points produced by a sine function $y(x) = A \cdot sin(\omega x - b)$ with constant frequency $\omega = 1$ but random amplitude $A \in [0.1, 5.0]$ and phase $b \in [0, \pi]$.

The regressor $f_\theta(x)$ is a simple neural network with parameters θ consisting of one input neuron, 2 hidden layers of size 40, and ReLU nonlinearities followed by an output layer with size one following [5]. The loss is the mean-squared error between $f_\theta(x)$ and the true value $y(x)$.

For each experiment we apply 10,000 consecutive MAML steps on the model parameters θ and use $n \in \{1, 4, 16, 64\}$ workers. Validation is performed by optimizing the model parameter θ for a given validation task T_i and applying 5 gradient steps. As stated in [5], we use $K = 10$ points for training, randomly chosen from $x \in [-5.0, 5.0]$ and $K \in \{5, 10, 20\}$ for validation.

Results show that all models trained with more than one worker perform better (Figure 2). Table 1 shows that we achieve similar results for more than 4 workers and comparable results with even more data points for validation. We compare the mean loss and standard deviation after 600 tasks and with a confidence interval of 95%. We assume that the increase in performance is a result of the quantity of tasks the meta-learner observes.

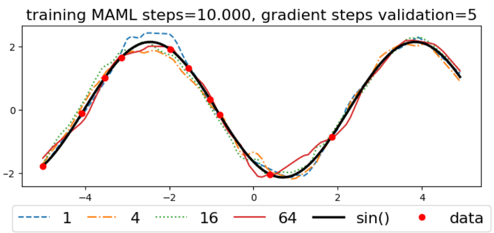

Fig. 2: The data points used for the gradient steps for validation are all sampled on the left side, so the model needs to extrapolate the values on the right side.

n	K		
	5	10	20
1	1.24 ± 0.10	0.78 ± 0.05	0.52 ± 0.04
4	1.02 ± 0.08	0.63 ± 0.05	0.43 ± 0.04
16	$\mathbf{0.83 \pm 0.07}$	$\mathbf{0.53 \pm 0.05}$	$\mathbf{0.32 \pm 0.03}$
64	0.90 ± 0.08	$\mathbf{0.54 \pm 0.05}$	$\mathbf{0.33 \pm 0.03}$

Table 1: Validation loss for n workers after training a meta-learner for 10,000 MAML steps and one gradient step w.r.t. the validation task

4.2 Image recognition task

We evaluate DMAML on Omniglot, and miniImageNet image recognition tasks, which are the most common recently used few-shot learning benchmarks [20, 18, 16].

We split the 50 available Omniglot alphabets into 30 alphabets for training and 20 for testing, as proposed by [18]. The dataset is augmented by rotating each instance by 90 degrees, and every rotated class is treated as a new class. The training dataset consists of 3.760 classes, the validation dataset of 2.732 classes. Every class consists of 20 images, which are resized to 28x28 pixels and are treated as grayscale images.

The miniImageNet dataset proposed by [16] consists of 64 training, 12 validation, and 24 test classes. As we are just doing validation, we aggregate the validation and test classes.

As proposed by [20], we apply the N-way, K-shot task as follows:

1. We apply fast learning using a support-set consisting of N (5 or 20) different classes and K (1 or 5) instances per class and
2. evaluate the models' ability to classify new instances of the N classes using a query-set consisting of K (Omniglot $K = 1$, miniImagenet $K = 15$) unseen instances of the N classes.

The models used for training have a simple architecture following [20]:

For Omniglot image recognition task we use a model with an input shape of (28, 28, 1), four modules with a 3x3 convolution with 64 filters, and a stride of 2. Each module followed by batch normalization [9] and a ReLU nonlinearity.

The miniImageNet model follows a similar architecture: The model has an input shape of (84, 84, 3) and consists of four modules with a 3x3 convolution with 32 filters, followed by a batch normalization, ReLU nonlinearity, and 2x2 max-pooling.

For all models, the output layer has a size of N and a softmax nonlinearity. The loss function is the cross-entropy error between the predicted and true class.

The following experiments were performed on the Omniglot image recognition task as a 5-way, 1-shot problem. In our experiments, we focus on the first 1,000 runs, which show a comparable trend in learning without training the algorithm exceedingly long. All experiments run with 3 different seeds, recognizing the need for more runs for a confidential result. The validation accuracy is the average over 10 different validation tasks, taken after fitting the model parameters to each task for 3 (Omniglot) and 10 (miniImageNet) gradient steps. We show the standard deviation of all 3 models and the output of a polynomial model fitted to the experiment data. The detailed results for both data sets are shown in the Appendix A.

4.3 Parallel workers

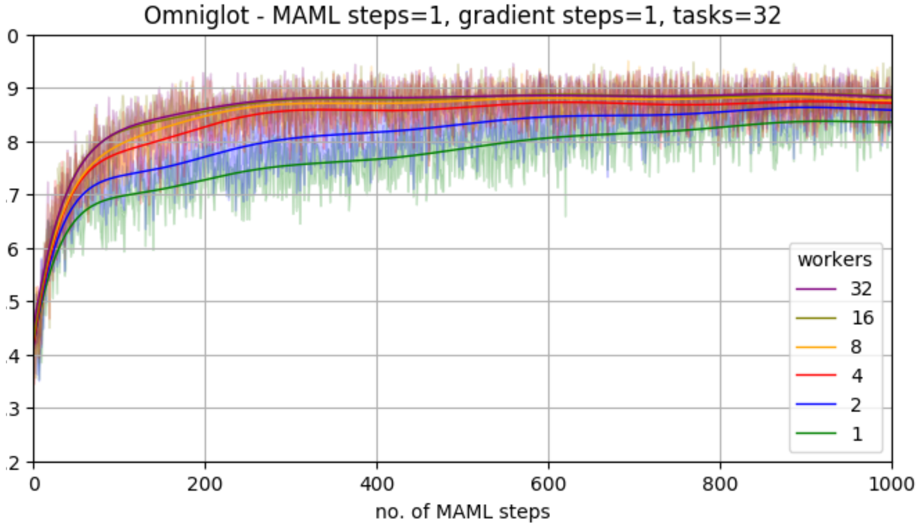

Fig. 3: Validation accuracy using n workers for learning. Learning is accelerated, when more workers are used. There also seems to be a breakeven point where more workers do not result in faster learning.

We run DMAML using $n \in \{1, 2, 3, 8, 16, 32\}$ parallel workers for learning. On the image recognition task, a small increase of workers leads to a steeper slope and faster convergence on a higher accuracy level (Figure 3). The difference in accuracy between 16 and 64 workers is much smaller than the difference between 1 and 4 workers. This small difference leads us to the assumption that there must be a breakeven point, where increasing the parallel workers n does not increase the outcome significantly. The maximum tradeoff between CPU-time and performance in this example is 2 or 4 workers. We run our experiments also for 10,000 MAML steps and can observe an acceleration of learning for the first 1,000 MAML steps. If n is higher, we can observe a divergence of the validation accuracy after 1,500 MAML steps (Appendix A). This divergence reinforces our assumption that 2 or 4 workers might be optimal for this setting. Also, a form of schedule to decrease n over time or a strategy to drop the parameters of underperforming workers might be helpful.

4.4 MAML steps per run

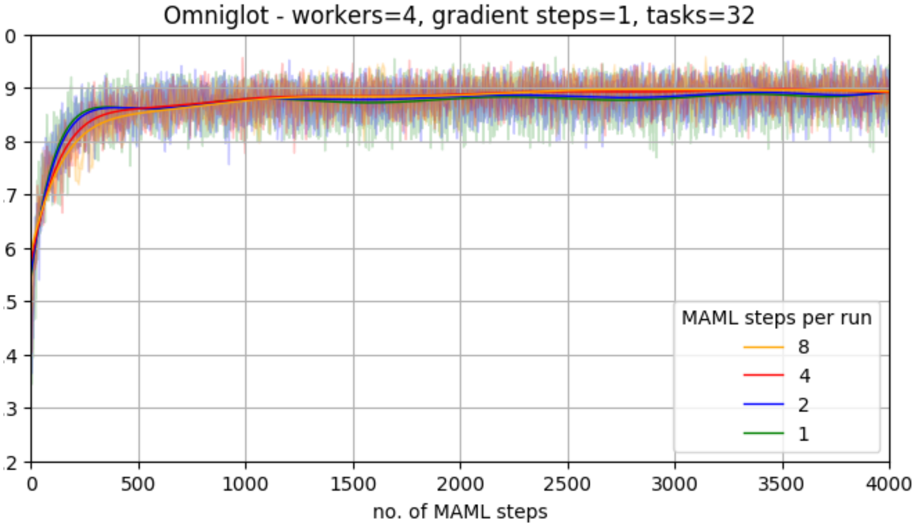

Fig. 4: The performance is shown depending on MAML steps. There seems to be no advantage in increasing the consecutive MAML steps in one run.

In this experiment, we increase the number of consecutive MAML steps per run. One worker performs multiple MAML steps before the model parameters are merged in the central learner. The resulting graphs for 1, 2, 4 and 8 consecutive MAML steps are shown on Figure 4. Less consecutive MAML steps increase the acceleration of learning at first but converge to the same level of accuracy. This convergence leads to the conclusion that there is no advantage in increasing the number of MAML steps in one run.

4.5 Tasks per MAML step

In the meta-learning setting, tasks are costly, as most of the time, time-consuming simulations need to be run. Therefore, we try to minimize overall tasks needed

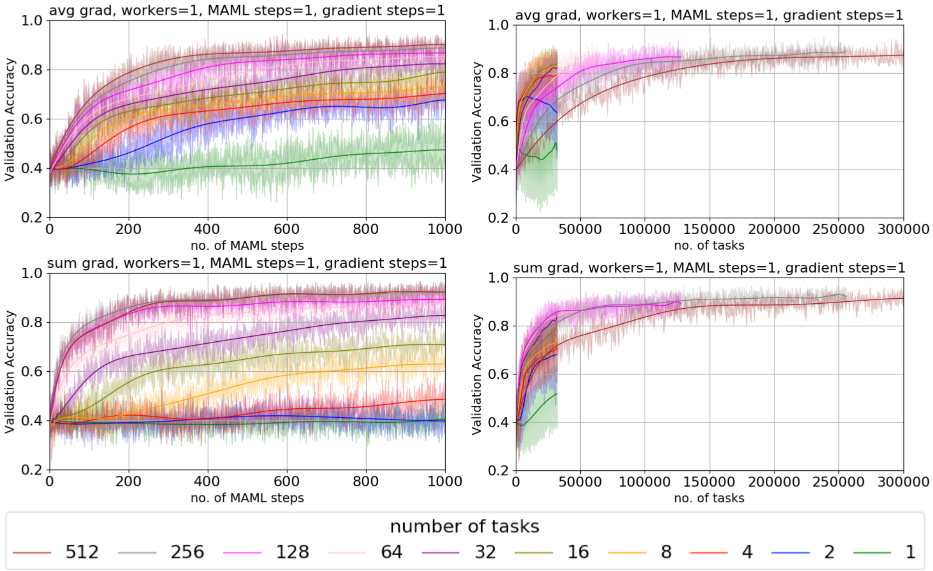

Fig. 5: Applying on Omniglot dataset (left) perfect parallelization: an increase of tasks perceived in one MAML step increases performance. (right) perfect serialization: an increase of tasks perceived in one MAML step increases wall clock time linearly. If we use fewer tasks per MAML step, we achieve a higher performance in a shorter amount of time. (top) we average the gradient over all tasks and apply the same learning rate β. (bottom) we sum over the gradients and apply a learning rate scheme.

for training. In this experiment, we also want to determine how we can obtain the greatest output out of one single task.

On Figure 5 (left), we assume a perfect level of parallelization of tasks and show that the accuracy rises with an increasing number of tasks observed in one MAML step. If we assume complete serialization of tasks, it is favorable to use less tasks per MAML step (Figure 5 (right)). In the top section, we show the results when averaging the gradients in the outer-learning loop over all tasks and apply the same learning rate β, resulting in a similar step size for each MAML step.

In contrast, we sum the gradients of all tasks in the outer-learning loop and apply a learning rate scheme (Figure 5, bottom). We set the learning rate for 32 tasks $\beta = 0.001$ as proposed by [5]. If we double the tasks used in one MAML step, we can be more certain that the gradient is more accurate, so we also double the learning rate and vice versa. Due to this learning rate scheme, learning is stabilized using less than 32 tasks, and learning is accelerated when using more tasks. As more tasks are used to determine the gradient, we can be more confident of the gradient's accuracy and perform a more significant update on the model parameters.

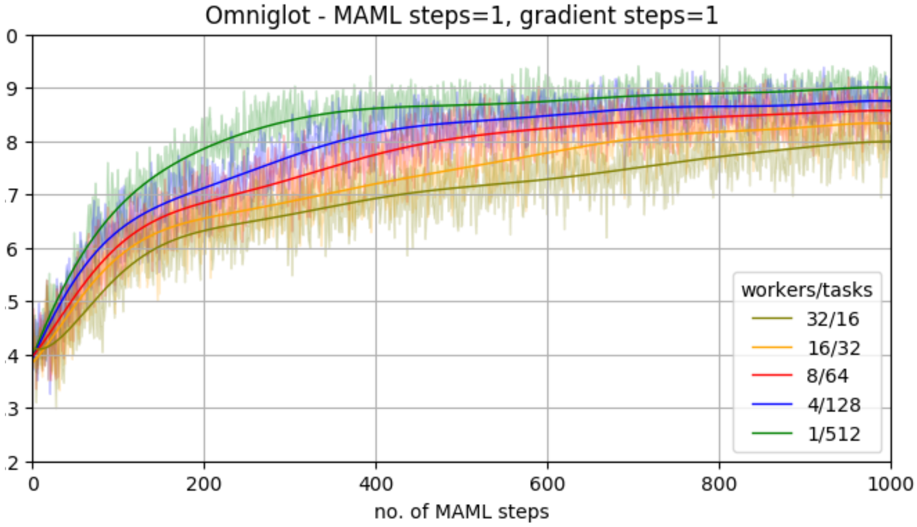

Fig. 6: The same amount of tasks distributed over multiple workers does not result in the same performance as using just one worker.

In one MAML step, the meta-model parameters are adapted to perform better for all observed tasks. With the same total number of tasks across all workers, the meta-learner achieves a higher accuracy if a single worker observes more tasks in one MAML step instead of distributing them to more workers (Figure 6). This observation leads to the conclusion that we should aim at assigning as many tasks per MAML step as possible.

5 Conclusion

We found two ways to parallelize MAML: 1. Parallelize the computation of the tasks or 2. distribute learning over multiple workers, DMAML.
In DMAML, every worker receives a copy of the model parameters and updates the parameters by applying MAML. We have investigated the impact of DMAML and the influence of hyperparameters on learning (consecutive MAML steps, number of workers and tasks).

In this work, we have shown:

1. If perfect parallelization of tasks is possible, meaning every MAML step takes the same amount of wall clock time, we should aim at using as many tasks as possible in one MAML step to increase performance.
2. If we need to serialize tasks, meaning the sum of computing all training tasks results in the same amount of wall clock time, we should aim for many MAML steps with fewer tasks per step. This results in an acceleration of learning.

3. Distributing learning over multiple workers does not improve learning compared to parallelizing the inner-learning loop of MAML.

Further research can investigate if other strategies of merging the model parameters in the central learner improves learning and suppresses the divergence (e.g., dropping the parameters of a worker with a high loss).

As mentioned, using more tasks in one MAML step is beneficial and, due to the averaging, can be expected to lead to a more accurate gradient. Therefore, it might be possible to increase the learning rate. We also worked with a constant learning rate β, however, if we optimize the learning rate individually, the optimal solution might be shifted. This requires further investigation.

References

1. Antoniou, A., Storkey, A., Edwards, H.: How to train your MAML. In: 7th International Conference on Learning Representations, ICLR 2019. International Conference on Learning Representations, ICLR (2019), http://arxiv.org/abs/1810.09502
2. Behl, H.S., Baydin, A.G., Torr, P.H.S.: Alpha MAML: Adaptive Model-Agnostic Meta-Learning (may 2019), http://arxiv.org/abs/1905.07435
3. Bertinetto, L., Torr, P.H., Henriques, J., Vedaldi, A.: Meta-learning with differentiable closed-form solvers. In: 7th International Conference on Learning Representations, ICLR 2019. International Conference on Learning Representations, ICLR (may 2019), http://arxiv.org/abs/1805.08136
4. Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated Meta-Learning with Fast Convergence and Efficient Communication (feb 2018), http://arxiv.org/abs/1802.07876
5. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. 34th International Conference on Machine Learning, ICML 2017 **3**, 1856–1868 (2017)
6. Finn, C., Xu, K., Levine, S.: Probabilistic model-agnostic meta-learning. In: Advances in Neural Information Processing Systems. vol. 2018-Decem, pp. 9516–9527. Neural information processing systems foundation (jun 2018), http://arxiv.org/abs/1806.02817
7. Gordon, J., Bronskill, J., Nowozin, S., Bauer, M., Turner, R.E.: Meta-learning probabilistic inference for prediction. In: 7th International Conference on Learning Representations, ICLR 2019. International Conference on Learning Representations, ICLR (may 2019), http://arxiv.org/abs/1805.09921
8. Grant, E., Finn, C., Levine, S., Darrell, T., Griffiths, T.: Recasting gradient-based meta-learning as hierarchical bayes. 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings (jan 2018), http://arxiv.org/abs/1801.08930
9. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd International Conference on Machine Learning, ICML 2015. vol. 1, pp. 448–456. International Machine Learning Society (IMLS) (feb 2015)
10. Koch, G.: Siamese neural networks for one-shot image recognition. ICML Deep Learning Workshop (2015), http://www.cs.toronto.edu/ gkoch/files/msc-thesis.pdf

11. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 10649–10657 (apr 2019). https://doi.org/10.1109/CVPR.2019.01091, http://arxiv.org/abs/1904.03758
12. Lee, Y., Choi, S.: Gradient-based meta-learning with learned layerwise metric and subspace. 35th International Conference on Machine Learning, ICML 2018 pp. 4574–4586 (jan 2018), http://arxiv.org/abs/1801.05558
13. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-SGD: Learning to Learn Quickly for Few-Shot Learning (jul 2017), http://arxiv.org/abs/1707.09835
14. Nichol, A., Achiam, J., Schulman, J.: On First-Order Meta-Learning Algorithms (mar 2018), http://arxiv.org/abs/1803.02999
15. Raghu, A., Raghu, M., Bengio, S., Vinyals, O.: Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML (sep 2019), http://arxiv.org/abs/1909.09157
16. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings (nov 2019)
17. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. 7th International Conference on Learning Representations, ICLR 2019 (jul 2019), http://arxiv.org/abs/1807.05960
18. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-Learning with Memory-Augmented Neural Networks. 33rd International Conference on Machine Learning, ICML 2016 pp. 2740–2751 (2016), http://proceedings.mlr.press/v48/santoro16.pdf
19. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in Neural Information Processing Systems pp. 4078–4088 (2017), http://arxiv.org/abs/1703.05175
20. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. Advances in Neural Information Processing Systems pp. 3637–3645 (2016), http://arxiv.org/abs/1606.04080
21. Zhou, F., Wu, B., Li, Z.: Deep Meta-Learning: Learning to Learn in the Concept Space (feb 2018), http://arxiv.org/abs/1802.03596
22. Zintgraf, L., Shiarlis, K., Kurin, V., Hofmann, K., Whiteson, S.: Fast context adaptation via meta-learning. 36th International Conference on Machine Learning, ICML 2019 pp. 13262–13276 (oct 2019), http://arxiv.org/abs/1810.03642

A Detailed experiment results

The validation accuracy after training DMAML using n workers for a number of MAML steps is calculated using the model with the highest validation accuracy after 10.000 steps out of three trained models, each using a different seed. We use a confidence interval of 95% and use 600 tasks for validation.
The result for Omniglot few-shot tasks are shown in Tables 2, 3 and 4, and the results for miniImageNet few-shot tasks are shown in Table 5.

Table 2: Comparison of n workers training for a number of MAML steps on the Omniglot image recognition task (5-way, 1-shot)

n	500	1000	2000	4000	8000	10000
1	0.81 ± 0.006	0.82 ± 0.007	0.87 ± 0.005	0.88 ± 0.006	0.85 ± 0.007	0.86 ± 0.007
2	0.84 ± 0.006	0.86 ± 0.006	0.87 ± 0.006	0.89 ± 0.005	0.89 ± 0.005	0.9 ± 0.005
4	0.87 ± 0.005	0.88 ± 0.005	0.88 ± 0.005	0.89 ± 0.006	0.91 ± 0.005	0.93 ± 0.004
8	0.88 ± 0.005	0.88 ± 0.005	0.86 ± 0.006	0.83 ± 0.007	0.88 ± 0.006	0.91 ± 0.005
16	0.88 ± 0.005	0.89 ± 0.005	0.84 ± 0.006	0.77 ± 0.01	0.86 ± 0.006	0.87 ± 0.007
32	0.89 ± 0.005	0.89 ± 0.005	0.85 ± 0.006	0.81 ± 0.009	0.75 ± 0.016	0.9 ± 0.005

Table 3: Comparison of n workers training for a number of MAML steps on the Omniglot image recognition task (5-way, 5-shot).

n	500	1000	2000	4000	8000	10000
1	0.96 ± 0.002	0.96 ± 0.002	0.97 ± 0.002	0.97 ± 0.002	0.97 ± 0.002	0.97 ± 0.003
2	0.95 ± 0.003	0.95 ± 0.003	0.96 ± 0.003	0.97 ± 0.002	0.97 ± 0.002	0.97 ± 0.002
4	0.96 ± 0.002	0.97 ± 0.002	0.97 ± 0.002	0.94 ± 0.005	0.97 ± 0.002	0.97 ± 0.004
8	0.96 ± 0.002	0.97 ± 0.002	0.97 ± 0.003	0.92 ± 0.009	0.85 ± 0.016	0.72 ± 0.021
16	0.96 ± 0.002	0.97 ± 0.002	0.97 ± 0.002	0.92 ± 0.008	0.84 ± 0.015	0.95 ± 0.007
32	0.97 ± 0.002	0.97 ± 0.002	0.97 ± 0.002	0.94 ± 0.005	0.67 ± 0.020	0.71 ± 0.019

Table 4: Comparison of n workers training for a number of MAML steps on the Omniglot image recognition task (20-way, 1-shot).

n	500	1000	2000	4000	8000	10000
1	0.59 ± 0.004	0.68 ± 0.004	0.73 ± 0.004	0.75 ± 0.004	0.76 ± 0.004	0.76 ± 0.004
2	0.66 ± 0.004	0.73 ± 0.004	0.77 ± 0.004	0.78 ± 0.004	0.78 ± 0.004	0.79 ± 0.004
4	0.71 ± 0.004	0.79 ± 0.004	0.81 ± 0.004	0.81 ± 0.004	0.81 ± 0.003	0.78 ± 0.004
8	0.74 ± 0.003	0.82 ± 0.003	0.84 ± 0.003	0.85 ± 0.002	0.75 ± 0.004	0.73 ± 0.003
16	0.77 ± 0.003	0.83 ± 0.003	0.83 ± 0.003	0.8 ± 0.004	0.77 ± 0.004	0.77 ± 0.003
32	0.76 ± 0.003	0.82 ± 0.003	0.81 ± 0.003	0.71 ± 0.004	0.76 ± 0.004	0.77 ± 0.003

Table 5: Comparison of n workers training for a number of MAML steps on the miniImageNet image recognition task (5-way, 1-shot and 5-way, 5-shot).

n	5-way, 1-shot			5-way, 5-shot		
	500	1000	2000	500	1000	2000
1	0.21 ± 0.003	0.21 ± 0.003	0.23 ± 0.004	0.3 ± 0.004	0.35 ± 0.005	0.39 ± 0.006
2	0.27 ± 0.005	0.27 ± 0.005	0.29 ± 0.005	0.38 ± 0.005	0.43 ± 0.006	0.46 ± 0.006
4	0.31 ± 0.005	0.32 ± 0.005	0.34 ± 0.006	0.44 ± 0.006	0.46 ± 0.006	0.5 ± 0.006
8	0.33 ± 0.005	0.36 ± 0.006	0.37 ± 0.006	0.46 ± 0.006	0.49 ± 0.006	0.52 ± 0.006
16	0.34 ± 0.006	0.37 ± 0.006	0.39 ± 0.006	0.47 ± 0.006	0.5 ± 0.006	0.53 ± 0.006
32	0.35 ± 0.006	0.36 ± 0.006	0.4 ± 0.006	0.48 ± 0.006	0.51 ± 0.006	0.53 ± 0.006

B Training parameters

The parameters used for training DMAML are shown in Table 6.

Table 6: Experiment parameters

Experiment parameter	Omniglot	MiniImageNet	Regression
Support-set samples (k)	1	1	10
Query-set samples (k')	1	15	10
Inner-loop learning rate (α)	0.4	0.4	0.01
Meta-learning rate (β)	0.01	0.01	0.01
Tasks (i)	32	4	10
MAML steps	1	1	1
Gradient steps	1	5	1
Validation steps	3	10	1
Validation tasks	10	10	10