protANIL: a Fast and Simple Meta-Learning Algorithm

Alexander Arzhanov
Deep Multi-Task and Meta Learning (CS330), Stanford University
aarz@stanford.edu

Abstract

A broad recognition of important practical benefits inherent to meta-learning paradigm has recently elevated the few-shot learning problem into the spotlight of machine learning research. While the optimization-based model-agnostic meta-learning algorithms offer a variety of appealing properties, such as model architecture flexibility and generalization capabilities, they come at a cost of high computational complexity. On the other hand, the metric-based meta-learning algorithms, that are built on a simple inductive bias mechanism, have proven their high efficiency in the limit of very few training examples. We propose an algorithm that combines the complementary strengths of these two approaches, and, at the same time, significantly lowers the computational cost.

We benchmark the execution times of the proposed protANIL algorithm to demonstrate its computational benefits during both training and inference. We also analyze its few-shot classification performance and find that the results are on par with the original MAML algorithm.

1. Introduction

Much of the recent attention in the machine learning research community was drawn to the meta-learning algorithms that are specifically designed to quickly learn new tasks by leveraging the prior experience from a set of similar tasks [3, 4, 13, 17, 19, 23, 24, 27].

In the context of few-shot classification, the meta-learning problem can be formally framed by defining a collection of related tasks and splitting it into disjoint sets for meta-training and meta-testing. Given that each task usually contains only a small number of data samples, a meta-learning algorithm is designed with an objective to learn how to quickly adapt to the new tasks in the most efficient manner. Once trained, the generalization properties of the meta-learning algorithm are assessed on the new unseen tasks sampled from the meta-testing set.

Out of many proposed meta-learning methods (see, e.g. Ref. [8] for a recent review), two particularly successful groups of few-shot classification methods can be highlighted: the non-parametric algorithms that rely on mechanism of metric learning, and the so-called optimization-based methods. The most notorious examples of the former include Siamese networks [10], Prototypical networks [23], Matching networks [27], Relational networks [24], and Graph neural networks [4] - all of which aim to tackle the few-shot classification problem by *"learning to compare"* and are based on the notion of similarity between the input images. The second family of methods, on the other hand, addresses the meta-learning problem by *"learning to adapt"* and aims at finding a good initialization for model parameters so that the required model adaptation procedure to a new task is efficient both in the quantity of labeled examples and the number of gradient update steps. The most renowned among these methods is perhaps the MAML [3] algorithm, which has given rise to a number of different variations, such as Reptile [15] and LEO [21].

While the continuous scientific effort in meta-learning research have lately resulted in many novel and increasingly complex network architectures and algorithms [8], a number of recent studies of few-shot classification problem suggest that learning good features during meta-training and reusing them during adaptation is the dominant success factor [2, 18, 25]. Motivated by these findings, we propose a simple and fast meta-learning algorithm titled *protANIL* that

- integrates the highly efficient inductive bias mechanism of metric-based approaches into the versatile optimization-based MAML algorithm, which is expected to reduce the intra-class embedding variance [5] and thereby contribute to learning better features;
- addresses the problem of computational complexity of MAML approach by limiting the calculation of second-order derivatives to only task-specific head of the network [18].

Our few-shot classification benchmarks show that, by employing these two modifications, protANIL performs similarly well as the original MAML algorithm, while, at the same time, is able to show a 10-fold speedup during training, and almost a 5-fold speedup during inference.

2. Method

In this section, we first introduce our notation and formally define the few-shot classification problem. Then we briefly review the two meta-learning methods that underlie the proposed protANIL algorithm, which we present in detail at the end of this section.

2.1. Problem definition

The goal of N-way K-shot classification problem is to provide an algorithm that can efficiently learn to generalize from a relatively small set of N classes with K examples per class, so that it is capable of performing well on any unseen examples among these N classes.

We can formally approach this meta-learning problem by first defining a collection of related tasks as $\mathcal{T} = \{T_t\}_{t=1}^{M}$, where each task is represented by a tuple $T_t = (\mathcal{S}_t, \mathcal{Q}_t)$ that consists of a *support* data set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K}$ and a *query* data set $\mathcal{Q} = \{(\mathbf{x}_i^*, y_i^*)\}_{i=1}^{K}$. Here, the $(\mathbf{x}, y)$-pairs are sampled from the same distribution and represent input images $\mathbf{x} \in \mathbb{R}^D$ with the corresponding labels $y \in \{1, \ldots, N\}$.

A common meta-learning procedure [27] is to first disjointly split the task collection set $\mathcal{T}$ into meta-train $\mathcal{T}_{train}$, meta-validation $\mathcal{T}_{val}$, and meta-test $\mathcal{T}_{test}$ sets, so that they have no classes in common. The meta-training is then carried out by first drawing a training task $T = (\mathcal{S}, \mathcal{Q})$ from $\mathcal{T}_{train}$. Then, conditioned on the support set $\mathcal{S}$, a differentiable distribution over classes $p(y^*|\mathbf{x}^*, \mathcal{S})$ is computed for each query point $\mathbf{x}^* \in \mathcal{Q}$, whereupon the parameters of the model are learned end-to-end to minimize the classification loss with, for example, gradient descent algorithm. Having adapted the model parameters to the entire query set within a training task, the progress of meta-training can be monitored by evaluating the performance of meta-learner on a task sampled from the validation set $\mathcal{T}_{val}$. Once fully meta-trained, the generalizability of the meta-learning algorithm is assessed by its performance averaged across all tasks from the meta-test set $\mathcal{T}_{test}$.

The meta-learning approaches can vary by computation details of the predicted class distributions, as well as how they are conditioned on the support set. Before presenting our proposed meta-learning algorithm, we briefly summarize the two approaches that lie in its foundation.

2.2. Prototypical Networks

Prototypical networks [23] adopt *metric-based* approach and utilize an embedding module $f_\theta : \mathbb{R}^D \to \mathbb{R}^H$ (parametrized by θ) to compute a prototype $\mathbf{c}_k \in \mathbb{R}^H$ for each class by taking a mean of all representation vectors from the support set that bear the same class label k, i.e.

$$\mathbf{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_k} f_\theta(\mathbf{x}_i) \tag{1}$$

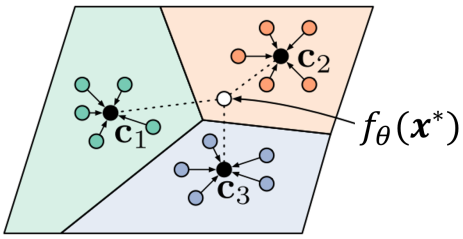

Figure 1: The class prototypes $\mathbf{c}_k$ in Prototypical networks are calculated as mean embedding vectors of all support points from the same class k. The query sample $f_\theta(\mathbf{x}^*)$ is classified by the closest prototype. Adapted from Ref. [23].

where $\mathcal{S}_k \subset \mathcal{S}$ denotes the set of samples of class k.

The query samples are then distance-compared to all of the prototypes, and classification is made based on the label of the prototype with the shortest Euclidean distance (see Fig. 1 for illustration). The class probability distribution is then calculated as

$$p(y^* = k|\mathbf{x}^*, \mathcal{S}) = \frac{\exp\left(-||f_\theta(\mathbf{x}^*) - \mathbf{c}_k||_2^2\right)}{\sum_{k'} \exp\left(-||f_\theta(\mathbf{x}^*) - \mathbf{c}_{k'}||_2^2\right)} \tag{2}$$

where the summation is over all classes in $\mathcal{Q}$.

2.3. MAML

The Model-Agnostic Meta-Learning (MAML) [3] algorithm aims to learn a good *initialization* for the parameters of the model in order to enable an efficient *optimization* of the meta-learner algorithm to a new task. The class probability distribution for a query sample $\mathbf{x}^* \in \mathcal{Q}$ is calculated as

$$p(y^* = k|\mathbf{x}^*, \mathcal{S}) = \sigma_k(g_{\varphi_i'} \circ f_{\theta_i'}(\mathbf{x}^*)) \tag{3}$$

where σ_k is a softmax function, f_θ is a embedding module and g_φ is a classifier head, s.t. $g_\varphi : \mathbb{R}^H \to \mathbb{R}^N$. In this case, the meta-learner is fully parameterized by $\phi = (\theta, \varphi)$. Then the parameters $\phi_i' = (\theta_i', \varphi_i')$ are obtained by an optimization of the initial *outer*-loop parameters ϕ (also known as *meta-initialization* parameters) via a so-called *inner*-loop adaptation by taking one (or more) gradient steps over the samples from the support set $\mathcal{S}$. This inner-loop adaptation is performed *separately for each training task* T_i. See Fig. 2 for an illustration. The MAML algorithm usually implements a linear classification head, so that $\varphi = (\mathbf{W}, \mathbf{b})$, but the architecture of the embedding module can differ [6, 13, 20, 26]. The meta-training is performed by calculating the cross-entropy loss on the query set $\mathcal{Q}$ after the inner loop adaptation, whereupon the error is backpropagated through *all* inner-loop steps and meta-initialization parameters ϕ are adjusted (Fig. 2). This procedure, however, imposes a considerable computational and memory burden due to necessity of computing the second-

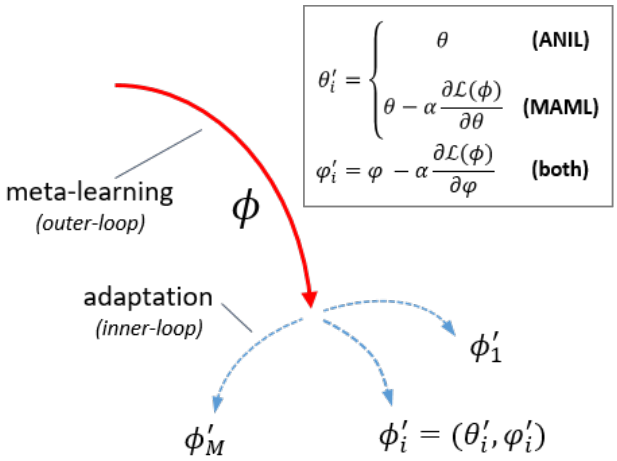

Figure 2: The principles of ANIL and MAML algorithms: the former does not tune the parameters of the embedding module during inner-loop adaptation, whereas the latter performs optimization of all parameters.

order derivatives and keeping track of the multiple inner-gradient paths. To this end, a number of different variations of MAML algorithm have been proposed [15, 21]. The simplest among them is perhaps the so-called *first-order* MAML (foMAML), which does not involve computation of second-order terms during training. This implies a significantly smaller computational footprint during meta-training, while still showing a surprisingly good performance on many few-shot classification tasks [3].

2.4. The protANIL algorithm

Motivated by the recent studies suggesting that learning good features during meta-training and reusing them during adaptation is the dominant success factor [2, 18, 25], we propose two modifications to the original MAML algorithm that are aimed to make it both simpler and faster without a noticeable accuracy sacrifice.

The prototype generator. First, in order to capitalize on the simple and effective inductive bias of Prototypical networks, we propose to incorporate the mechanism of metric-learning into the versatile approach of MAML algorithm. To this end, we swap the head of MAML meta-learner with a differentiable distance-based classifier, parametrized by a weight matrix $\mathbf{W} = [\mathbf{c}_1, \ldots, \mathbf{c}_N] \in \mathbb{R}^{H \times N}$, where each individual weight vector $\mathbf{c}_k \in \mathbb{R}^H$ can intuitively be interpreted as a class prototype. Then for each embedded sample $f_\theta(\mathbf{x})$ we compute its cosine-similarity with every weight vector $\mathbf{c}_k$ to obtain the corresponding similarity scores $\mathbf{s} = [s_1, \ldots, s_N]$, where each score is calculated as

$$s_k = \mathcal{K}(\mathbf{c}_k, f_\theta(\mathbf{x})) = \frac{\mathbf{c}_k \cdot f_\theta(\mathbf{x})}{||\mathbf{c}_k|| \, ||f_\theta(\mathbf{x})||}. \quad (4)$$

The class probability distribution for samples are then obtained by normalizing the similarity scores with a softmax function. See Fig. 3 for illustration.

Inspired by Ref. [2], we also incorporate a couple of simple tricks to stabilize and improve training with a prototypical head. First, we introduce scaling of similarity scores by a constant factor in order to strengthen the signal input to the softmax function. This introduced hyper-parameter helps to prevent the weight vectors from collapsing to zeros. The scaling can be thought of as an inverse of the temperature parameter [7] often used, for example, in RL or NLP domain. Besides this, we also use weight normalization [22] for the weight matrix $\mathbf{W}$. This reparameterization technique decouples the length of the weight vectors from their direction, which helps us to improve the conditioning of the optimization problem and speeds up convergence of stochastic gradient descent. Finally, we also note that an introduction of the prototype mechanism has been shown [5, 9] to help with the reduction of variance among the intra-class embedding, which is also expected to contribute to better feature learning.

The ANIL algorithm. We address the issue of MAML's computational complexity by suggesting to leverage the recently proposed ANIL algorithm [18] which removes the MAML's inner loop for all but the task-specific head of the network (see Fig. 2), while still showing performance on par with the original MAML implementation.

We would like to stress, that the idea of equipping MAML with a parametric prototype generator was originally motivated by discussions in Refs. [2, 17], and only once this project was underway, it came to our attention that similar ideas have been already proposed and recently published (as a secondary objective) in an ICLR paper [26]. The novelty of our contribution is in the thorough benchmark of the computationally beneficial ANIL algorithm augmented with a prototypical classifier head.

3. Experiments

We leverage the `learn2learn` [1] meta-learning research library and implement MAML, foMAML, and protANIL in PyTorch [16] to evaluate these algorithms on two established datasets for few-shot classification benchmarks: *Omniglot* and *mini-Imagenet*. First, in order to test our implementation, we replicate the reported results for the original MAML [3] and ANIL [18] algorithms. Then we perform a thorough benchmark of training and inference speedups of foMAML and protANIL relative to the computationally demanding MAML algorithm. Finally, we present a detailed performance comparison of MAML and protANIL algorithms on the mini-Imagenet few-shot classification problem.

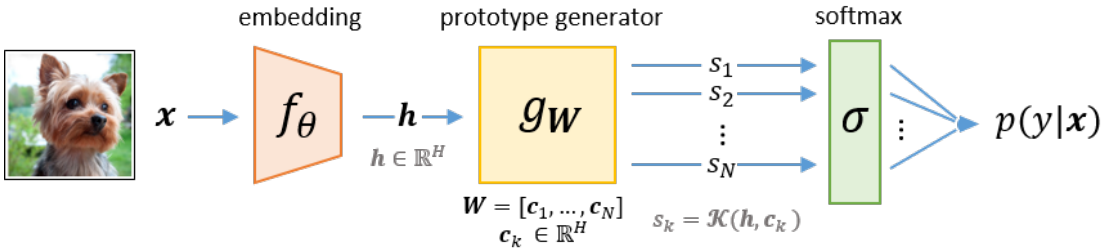

Figure 3: Schematic overview of the protANIL network architecture. The feature vector $\mathbf{h} = f_\theta(\mathbf{x})$ is cosine-compared with column vectors $\mathbf{c}_k$ of the classification head, whereupon softmax is used to produce a distribution of class probabilities.

3.1. Datasets

Omniglot [12] contains 1623 classes, where each class is a character coming from one of 50 different alphabets. For every class there is a total of 20 hand-written monochrome samples with 28×28 pixel resolution. We follow the procedures of Ref. [19] and take the first 1100 classes for training, the next 100 classes for validation, and the rest for testing. Each split is augmented with multiples of 90-degree rotations.

mini-Imagenet [27] was specifically derived for few-shot classification tasks from the original ILSVRC-12 dataset [11] and is comprised of 100 classes with a total of $60k$ RGB-images in 84×84 resolution. According to the established procedure [19], we split the dataset in 3 sets with 64 training, 16 validation, and 20 testing classes, with each class containing 600 examples.

3.2. Network architecture

In order to make direct comparison to previously published results, we implement a similar 4-layer CNN architecture of the embedding module as described in Ref. [3] and Ref. [19] for Omniglot and mini-Imagenet cases, respectively.

3.3. Replication of published results

The first part of the project involved replicating some of the previously published results to test our implementation of MAML and ANIL algorithms. To this end, we performed two benchmarks: one with Omniglot dataset on the 20-way 5-shot classification tasks, and another one with mini-Imagenet on 5-way 5-shot classification tasks. The models were trained with meta-batches of 32 tasks for $3k$ iterations and 1 inner gradient step in case of Omniglot, and for $12k$ iterations and 5 inner gradient steps in case of mini-Imagenet. The tests were performed on 600 randomly sampled tasks from the corresponding meta-test set. The results are shown in Tab. 1, where they are also compared to the previously published values. It is likely that our test accuracies could still improve a bit further with a longer training,

	Omniglot (20-way / 5-shot)	mini-Imagenet (5-way / 5-shot)
MAML [3]	98.9 ± 0.2	63.1 ± 0.9
MAML (ours)	96.8 ± 0.1	64.4 ± 0.6
ANIL [18]	98.0 ± 0.3	61.5 ± 0.5
ANIL (ours)	95.5 ± 0.2	61.0 ± 0.8

Table 1: Comparison of few-shot classification performance of our implementation and previously published results.

but given that a MAML training on Omniglot took around 8 hours on an Nvidia Tesla V100 GPU, and ran for more than 14 hours on mini-Imagenet, we chose to stop training once a large degree of convergence was seemingly reached.

3.4. Computational benefits

We benchmark the computation times of our implementations for MAML, foMAML, and protANIL algorithms. The results for Omniglot 20-way and mini-Imagenet 5-way classifications tasks are shown on Fig. 4. The x-axis indicate average execution times in seconds per one meta-batch of 32 tasks during training (Fig. 4, left side) and during inference (Fig. 4, right side). The results are averaged over 1000 such batches. The computations were performed on an Nvidia MX250 GPU. The labels at the edges of the bars indicate the observed speedup relative to MAML algorithm.

We can see that both foMAML and protANIL offer a significant computational benefit during meta-training. The boost to training times noticeably increases as the number of inner gradient updates is raised from 1 to 5. For example, in case of mini-Imagenet 5-way 5-shot tasks, the speedup offered by foMAML increases from 2.1x to 2.9x as we go from 1 to 5 inner updates. This noticeable computational advantage relative to MAML comes foremost from removing the necessity of using second-order derivatives when backpropagating the outer-loop gradient through the inner-loop gradient operator in the meta-objective. In the meantime, given the same classification task, the protANIL yields a speedup of 3.0x and 10.1x, re-

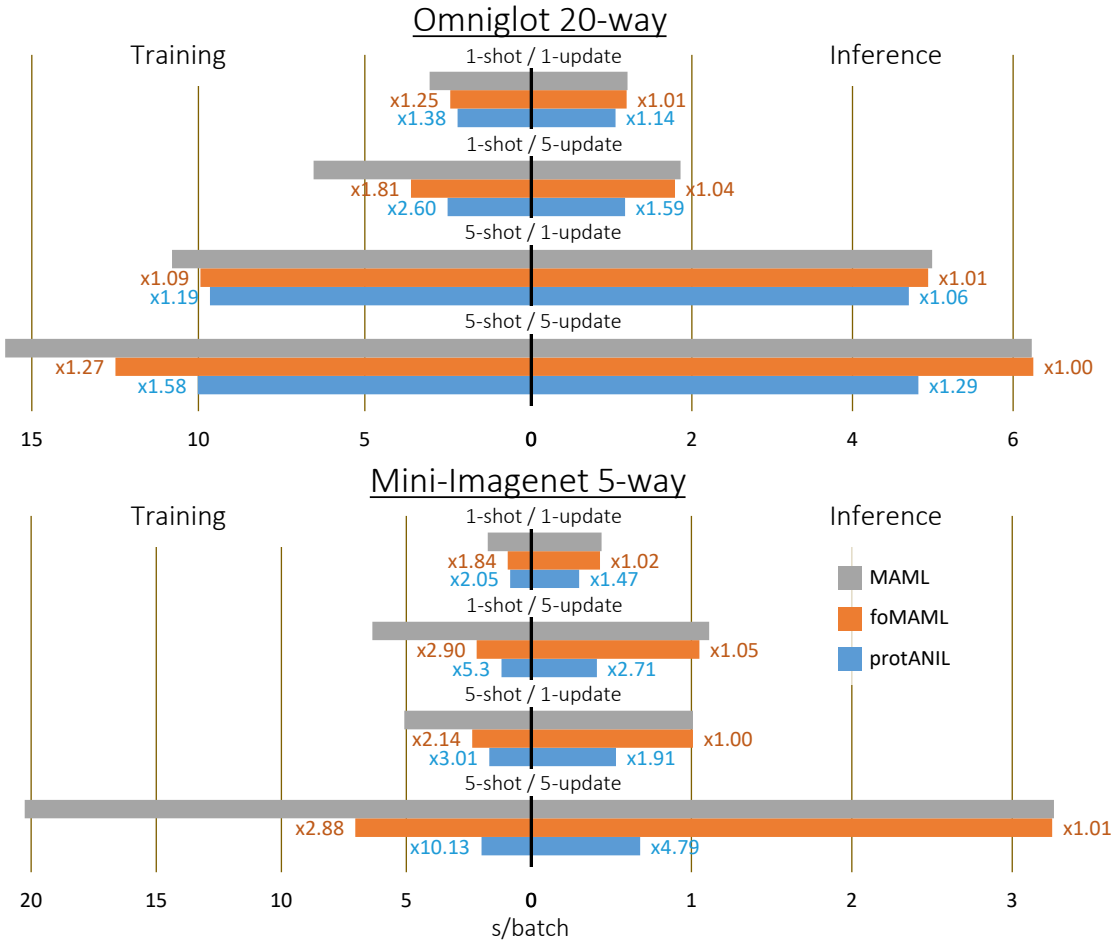

Figure 4: Comparison of estimated computational times for MAML, foMAML, and protANIL during training (left) and inference (right) on both Omniglot 20-way (above) and mini-Imagenet 5-way (below) datasets. The execution times are seconds per meta-batches of 32 tasks for either 1-shot or 5-shot classification problems with either 1 or 5 number of inner gradient updates. The values at the edges on the bar plots indicate a computational speedup relative to MAML algorithm.

spectively. The difference in computational boost between foMAML and protANIL, stems from the fact that, whereas foMAML removes the need for computing Hessian-vector products in an inner-loop backward pass, the protANIL removes the inner-loop almost completely. Furthermore, since foMAML is identical to MAML during the inference time, we expectedly do not see any noticeable difference here. At the same time, however, an absence of the full inner-update loop for protANIL offers a drastic up to 5-fold speedup during inference, if compared to both MAML and foMAML.

3.5. Classification performance

We perform a series of experiments with our implementations of MAML, foMAML, ANIL, and protANIL on mini-Imagenet 5-way 5-shot classification benchmark. The results are tabulated in Tab. 2. With some minor hyper-parameter tuning (such as adjusting inner-loop learning rate, number of inner gradient steps, and softmax temperature), we were able to replicate all the published re-

	Ours	Published	
MAML	64.4 ± 0.6	63.1 ± 0.9	[3]
foMAML	63.6 ± 0.7	63.2 ± 0.9	[3]
ANIL	61.0 ± 0.8	61.5 ± 0.5	[18]
protANIL	$\mathbf{64.1 \pm 0.8}$	—	

Table 2: Performance of MAML, foMAML, ANIL, and protANIL on mini-Imagenet 5-way 5-shot classification benchmark. Previously published results are shown for reference.

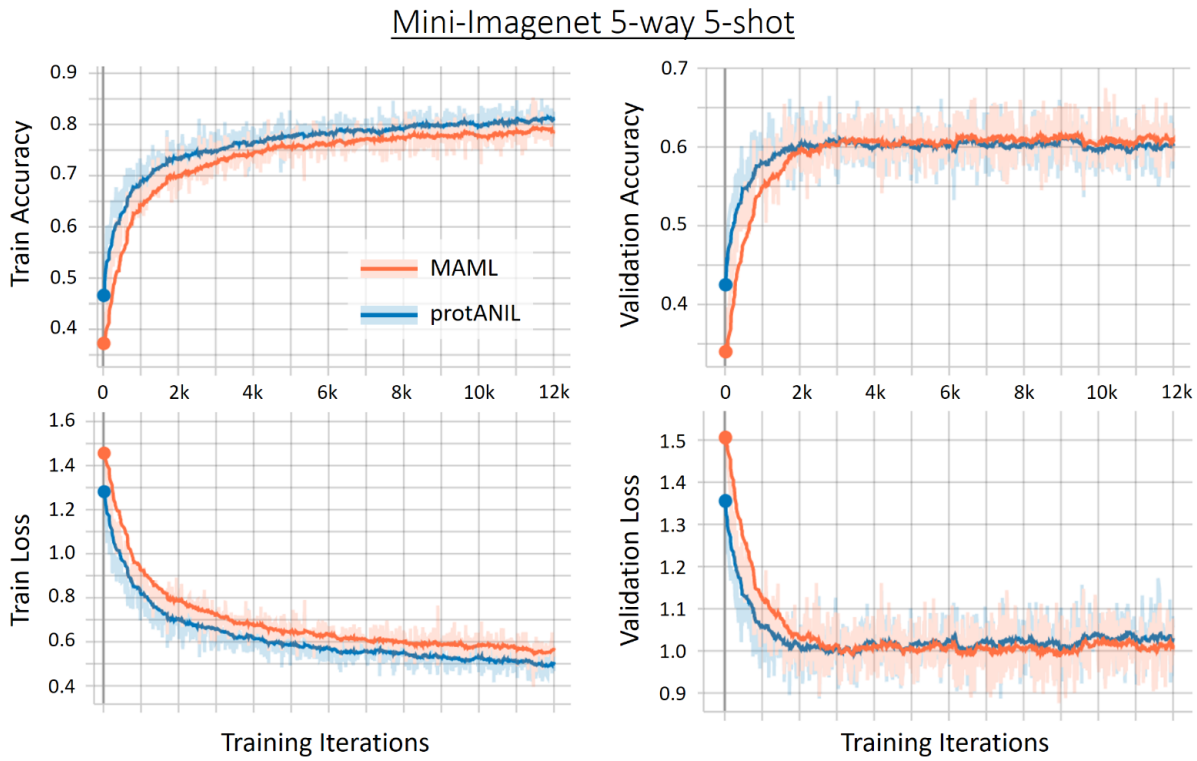

Figure 5: The loss and accuracy training curves for MAML and protANIL algorithms on mini-Imagenet 5-way 5-shot classification problem.

sults, and even saw a slight improvement over the reported MAML values in Ref. [3]. The Fig. 5 shows the loss and accuracy curves both for MAML and protANIL algorithms, which exhibit very similar dynamics during training. This, in turn, results in almost indistinguishable performance of our MAML implementation and protANIL with the test accuracies reaching 64.4% and 64.1%, respectively (Tab. 2). At the same time, we find that protANIL performs significantly better (64.4%) than its ANIL counterpart (61.0%), which can be indicative of the fact that introduction of prototypical head has indeed improved the quality of learned features.

4. Conclusions and Outlook

The gradient-based MAML algorithm offers a variety of appealing properties, such as model architecture flexibility and generalization capabilities, which, however, come at the cost of high computational complexity. Meanwhile, the metric-based Prototypical networks, built on a simple inductive bias mechanism, are found to be highly efficient in the limit of very few training examples. The proposed protANIL algorithm combined the complementary strengths of these two approaches and significantly lowered the computational cost at the same time.

Given the very limited time for this project, there are still many interesting directions to explore in the future work. For example:

- test the performance of protANIL on other established datasets for few-shot classification benchmarks;
- closely investigate the effect of the prototypical head on the quality of the learned features, by e.g. comparing the t-SNE [14] visualizations of the feature respresentations learned with the protANIL and other MAML-based variants;
- since most of meta-testing implementations for few-shot classification approaches operate in a *de facto* transductive mode [15] and are somewhat overly optimistic due to sharing of the batch-norm statistics between the test samples, it would be interesting to compare their performance in truly inductive (and more realistic) scenarios;
- investigate other network architectures for the embedding module;
- explore if protANIL can also be successfully employed for reinforcement learning applications.

References

[1] Sébastien M R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for Meta-Learning research. Aug. 2020.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[4] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[5] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[6] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[9] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Deep transfer metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 325–333, 2015.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[12] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[13] Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. *arXiv preprint arXiv:1905.06527*, 2019.

[14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[15] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[17] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[18] 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*, 2019.

[19] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[20] Marc Rußwurm, Sherrie Wang, Marco Korner, and David Lobell. Meta-learning for few-shot land cover classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 200–201, 2020.

[21] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[22] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pages 901–909, 2016.

[23] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[24] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Com-*

puter Vision and Pattern Recognition, pages 1199–1208, 2018.

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

[26] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[27] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.