Final Report

Huang Gaoang, Xu Jiafeng, Wang Jingnan

December 18, 2020

1 Introduction

It is well known that deep neural networks can perform well on some tasks using large-scale labeled data sets. However, powerful like it is, for example, deep neural networks do a quite hard job on small-scale image classification while a human child can learn a new class quickly through just one or few images. For a type of unknown or unusual image, people can quickly distinguish that this type of image does not belong to a known type of image based on the knowledge previously learned, and deep neural networks are likely to categorize it incorrectly.

To tackle this challenge, a research topic called few-shot image classification, aiming at recognizing new visual concepts with just a few amount of labeled samples in each class, is brought up. Certainly, fine-tuning a model based on trained data set on the novel labeled data is the most intuitive method. Unfortunately, it performs not very well since neural networks will suffer severe overfitting easily on the few given data. Data augmentation and regularization techniques can alleviate overfitting in such a limited-data regime, but they do not solve it. One promising direction to few-shot classification is the meta-learning where transferable knowledge is extracted and propagated from a collection of tasks to prevent overfitting and improve generalization. This is achieved by repeatedly sampling small subsets from the large pool of base images, effectively simulating the few-shot scenario. MAML Finn et al. [1] trains the meta-learner to provide a good initialization of the classifier parameters. Meta-SGD Li et al. [5]'s meta-learner creates an adaptive learning rate for classifier training. Ravi and Larochelle [7] replaces the gradient-based optimizer with a LSTM to train the classifier. Meta-learning may take the form of learning a shared metric [14], a common initialization for few-shot classifiers [7], or a generic inference network [10].

Among different meta strategies, gradient descent based methods, such as MAML approach, are particularly promising for today's neural networks [9, 11]. The successful MAML approach aimed to meta-learn an initial condition (set of neural network weights) that is good for fine-tuning on few-shot problems. The strategy here is to search for the weight configuration of a given neural network such that it can be effectively fine-tuned on a sparse data problem within a few gradient-descent update steps [4].

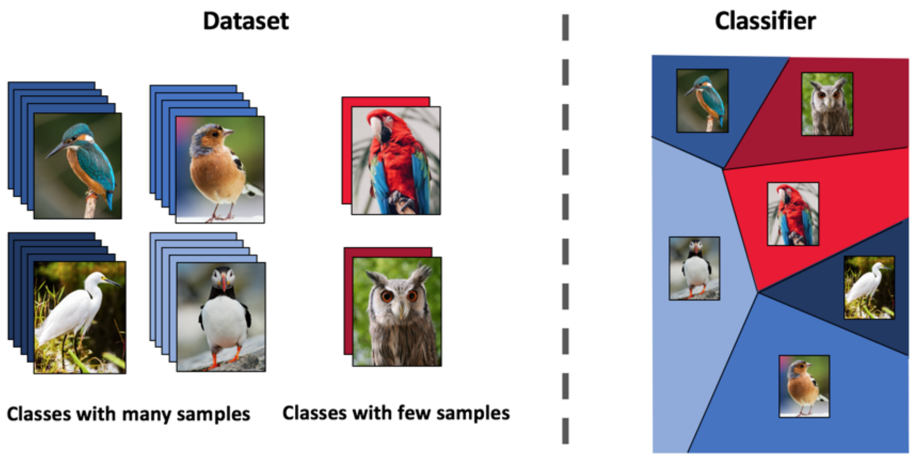

Figure 1: The basic form of few-shot classification

For the K-shot N-way classification task, we propose our method based on the famous MAML algorithm. We simply modified it by increasing the number of times of gradient descent of every iteration since the original MAML only perform once, which is still room for fine-tuning for better performance. Since MAML needs to differentiate through the optimization process, it's not a good match for problems where we need to perform a large number of gradient steps at test time [6]. Thus, we ignore the high derivative to avoid complicated implementation and high time complexity at the expense of losing some gradient information. Backtracking line search is typically used for gradient descent which is easy to implement, and applicable for very general functions. Considering that the previous MAML algorithm is easy to converge to the local optimal points in most cases, however, we will search several optimal values from different initial points and only keep the best one to avoid the bad local optimal points. Our algorithm will be tested on some famous few-shot classification datasets such as Omniglot, miniImagenet etc., and we will compare the accuracy of our algorithm with the previous baseline of MAML's in 5-way 1-shot, 5-way 5-shot, 20-way 1-shot and 20-way 5-shot classification experiments.

2 Related Work

Recent years,few-shot learning has become more and more important and powerful.Early research on few-shot learning mainly concentrate on image processing, and the few-shot learning model is classified into three categories which are model based, metric based and optimization based. Model based methods aims at updating parameters on few samples rapidly through special model directly mapping input to prediciton function, metric based method classify the test by using the idea of nearest neighbour which adopt the metric from samples in batch set and samples in support set, and optimization based method thinks the trivial gradient descent method is not suitable for few-shot learning, and it try to modify the optimization method to accomplish the few-shot classification task.

There are many creative and powerful method has been come up.

Observing the phenomenon many non-parametric allow novel examples to be rapidly assimilated, while not suffering from catastrophic forgetting.Matching Networks architecture which is a neural network uses recent advances in attention and memory that enable rapid learning, and it does not need any finite tuning on the classes it has never seen.[14]

Combining the features of few-shot and zero-shot learning, a Relation Network performs few-shot recognition by learning to compare query images against few-shot labeled sample images.Relation Network thinks that metric approach is very important in networks, so it trains a network like Convolutional neural network to learn the metric. This Relation Network use four convolutional blocks for embedding module, and the strategy here is to search for the weight configuration of a given neural network such that it can be effectively fine-tuned on a sparse data problem within a few gradient-descent update steps.[12]

Focusing on feature, few-shot classification via learned feature-wise transformation use feature-wise transformation layers to simulate various image feature distributions extracted from the tasks in different domains, and develope a learning-to-learn method to optimize the hyper-parameters of the feature-wise transformation layers.Contrary to the exhaustive parameter hand-tuning process ,they propose learning-to-learn algorithm to find the hyper-parameters for the feature-wise transformation layers to capture the variation of image feature distribution across various domain. [13]

Using the graph neural networks, DPGN extract the instance feature of support and query sample to obtain the distribution feature for each sample by calculating the instance-level similarity over all support samples, and DPGN devise a dual complete graph network that combines instance-level and distribution-level relations, where each node feature is concatenated with corresponding class label, then node features are updated via the attention mechanism of graph network to propagate the label information.[15]

There are many models which based on the MAML, that is Model-agnostic meta-learning, which addresses the general problem of meta-learning including few-shot learning, such as probabilistic model-agnostic meta-learning which injects noise into gradient descent at meta-test time, and Bayesian Model-Agnostic Meta-Learning which introduces Bayesian methods for fast adaption and meta-update.[2, 16]

Our main contributions are summarized as follows:

We devise a new loss function which doesn't cause overfiting problem.

Many experiments are conducted on three popular datasets for few-shot learning. By comparing with our method to the original MAML method, we discover a new prospective to consider the way which we can do some improvement.

3 Method

First, we will introduce several notations:

1. $p(\tau)$ is the task distribution. And τ denotes task sampled from $p(\tau)$.
2. $U^k_{\tau,A}(\theta)$ is the updated θ after k times of gradient descent using data sampled from training set A of τ.
3. $\mathcal{L}_{\tau,B}$ denotes the loss function w.r.t test set B of τ.

With these notations, the optimization goal of MAML can be written as

$$\min_\theta \mathbb{E}_{\tau\sim p(\tau)}[F_\tau(\theta)] = \mathbb{E}_{\tau\sim p(\tau)}[\mathcal{L}_{\tau,B}(U^1_{\tau,A}(\theta))]$$

where F_τ is a new definition of loss function w.r.t task τ which aims to minimize the loss of θ after one iteration of gradient descent. And in every iteration, the algorithm needs to compute the gradient

$$\nabla_\theta F_\tau(\theta) = \nabla_\theta \mathcal{L}_{\tau,B}(U^1_{\tau,A}(\theta)) \tag{1}$$
$$= \nabla_\theta \mathcal{L}_{\tau,B}(\theta - \alpha\nabla_\theta \mathcal{L}_{\tau,A}(\theta)) \tag{2}$$

which involves the second derivative.

Obviously, we can update θ for more times to achieve lower loss as long as it does not overfit. For example, we replace $U^1_{\tau,A}(\theta)$ with $U^k_{\tau,A}(\theta)$ where $k > 1$ in the optimization goal. However, this change will cause higher derivative when performing gradient descent which is very hard to deal with. In order to avoid derivatives higher than first order, we should draw on the idea of first-order MAML(FOMAML) and treat those higher derivatives as constants.

Formally, our optimization goal is

$$\min_\theta \mathbb{E}_{\tau\sim p(\tau)}[F_\tau(\theta)] = \mathbb{E}_{\tau\sim p(\tau)}[\mathcal{L}_{\tau,B}(U^k_{\tau,A}(\theta))]$$

And we should compute

$$\nabla_\theta F_\tau(\theta) = \nabla_\theta \mathcal{L}_{\tau,B}(U^k_{\tau,A}(\theta)) \tag{3}$$
$$\approx \nabla_{\theta'} \mathcal{L}_{\tau,B}(\theta') \tag{4}$$

to perform gradient descent. In effect, our weight parameter θ are updated for k times on training set and then updated for once on the test set.

Since F_τ is non-convex with high probability, the algorithm may converge to some local optimal point which is not good enough. So we decide to run the original algorithm for several times from different initial points and find the result with lowest loss. We believe this strategy will improve the accuracy although it takes more time for training.

we are very careful to choose hyperparameters, almost the same with the original setting, and in tuning the hyperparameters we use the principle of single variable to observe the variations caused by the changing hyperparameters.

Our algorithm is shown in Algorithm 1 in detail.

For testing, we fine-tune our model with the test data. In other words, we conduct several model updates on the test set. Then we can test the accuracy of classification.

Algorithm 1: Improved MAML

Require: $p(\tau)$, N, k

for $try \leftarrow 1, \cdots, N$ **do**
  randomly initialize θ
  while not done **do**
    Sample task $\tau \sim p(\tau)$
    $A, B \leftarrow \tau$
    for $iteration \leftarrow 1, \cdots, k$ **do**
      Evaluate gradient $\nabla_\theta \mathcal{L}_{\tau,A}(\theta)$
      $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\tau,A}(\theta)$
    end for
    Evaluate gradient $\nabla_\theta \mathcal{L}_{\tau,B}(\theta)$
    $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\tau,B}(\theta)$
  end while
  Keep the result in an array
end for
Choose the best result from the array

4 Experimental Setting

We plan to evaluate the performance of our algorithm on Omniglot [3], miniImagenet [8] and tieredImageNet. And we will also evaluate the original MAML on these datasets as our baseline. In the end, we will compare the accuracy of our method and MAML in 5-way 1-shot, 5-way 5-shot, 20-way 1-shot and 20-way 5-shot classification experiments.

All of our experiment is conducted on GeForce RTX 2080 Ti with CPU being Intel(R) Xeon(R), using python 3.6 and tensorflow-gpu 1.14.

4.1 Omniglot

we have conducted 20way-1shot omniglot baseline using the command:

python main.py –datasource=omniglot –logdir=logs/omniglot20way1shot/ –metatrain_iterations=60000 –meta_batch_size=16 –update_batch_size=1 –num_classes=20 –update_lr=0.1 –stop_grad=False

Table 1: 20way-1shot omniglot baseline

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	4.9999%	90.5501%	90.6418%	90.7168%	90.7418%	90.7585%
confidence interval	±0.0000%	±90.5501%	±0.5137%	±0.5102%	±0.5081%	±0.5059%
update	update 6	update 7	update 8	update 9	update 10	
accuracy	90.7835%	90.8251%	90.8168%	90.8085%	90.8251%	
confidence interval	±0.5054%	±0.5052%	±0.5061%	±0.5069%	±0.5073%	

Table 2: 20way-1shot omniglot baseline

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	4.8417%	84.7667%	85.841703%	86.55836%	87.891763%	88.850087%
confidence interval	±0.00295	±0.00609	±0.00585	±0.00569	±0.00569	±0.0057
update	update 6	update 7	update 8	update 9	update 10	
accuracy	89.35011%	89.508444%	89.55009%	89.62509%	0.89650095%	
confidence interval	±0.00556	±0.00561	±0.00559	±0.00559	±0.00558	

4.2 miniImagenet

we have conducted 5way-1shot miniimagenet baseline using the command:

python main.py –datasource=miniimagenet –logdir=logs/miniimagenet5way1shot/ –metatrain_iterations=60000

–meta_batch_size=4 –update_batch_size=1 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –stop_grad=False

Table 3: 5-1shot miniimagenet baseline

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	0.480000	0.560000	0.623333	0.553333	0.630000	0.570000
update	update 6	update 7	update 8	update 9	update 10	
accuracy	0.530000	0.560000	0.566667	0.556667	0.536667	

we have conducted 5way-1shot miniimagenet using the command:

python main.py –datasource=miniimagenet –logdir=logs/miniimagenet5way1shot/ –metatrain_iterations=60000 –meta_batch_size=4 –update_batch_size=1 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –num_updates=3 –attempt_rounds=3

Table 4: 5-1shot miniimagenet

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	0.566667	0.486667	0.550000	0.403333	0.580000	0.556667
update	update 6	update 7	update 8	update 9	update 10	
accuracy	0.586667	0.656667	0.556667	0.566667	0.633333	

4.3 tiered Imagenet

we have conducted 5way-1shot tiered imagenet baseline using the command:

python main.py –datasource=tieredimagenet –logdir=logs/tieredimagenet5way1shot/ –metatrain_iterations=60000 –meta_batch_size=4 –update_batch_size=1 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –stop_grad=False

Table 5: 5way-1shot tiered imagenet baseline

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	19.19%	22.53%	24.29%	28.46%	29.79%	30.16%
confidence interval	±0.0126	±0.0103	±0.0123	±0.0141	±0.0146	±0.0149
update	update 6	update 7	update 8	update 9	update 10	
accuracy	30.23324%	30.466574%	30.433244%	30.39992%	30.33325%	
confidence interval	±0.0148	±0.0148	±0.0147	±0.0148	±0.0148	

we have conducted 5way-1shot tiered imagenet using the command:

python main.py –datasource=tieredimagenet –logdir=logs/tieredimagenet5way1shot/ –metatrain_iterations=60000 –meta_batch_size=4 –update_batch_size=1 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –num_updates=3 –attempt_rounds=3

Table 6: 5way-1shot tiered imagenet

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	19.999886%	36.3999%	41.599944%	45.233306%	45.56666%	45.566672%
confidence interval	±0.00000009	±0.0144	±0.0173	±0.0185	±0.0188	±0.0187
update	update 6	update 7	update 8	update 9	update 10	
accuracy	45.66667%	45.73334%	45.666662%	45.633325%	45.666662%	
confidence interval	±0.0187	±0.0188	±0.0187	±0.0187	±0.0187	

we have conducted 5way-5shot tiered imagenet baseline using the command:

python main.py –datasource=miniimagenet –logdir=logs/miniimagenet5way5shot/ –metatrain_iterations=60000 –meta_batch_size=4 –update_batch_size=5 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –stop_grad=False

Table 7: 5way-5shot tiered imagenet baseline

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	19.78%	58.96%	57.60%	59.76%	60.99%	63.30%
confidence interval	±0.0068	±0.0093	±0.0098	±0.0097	±0.0099	±0.0096
update	update 6	update 7	update 8	update 9	update 10	
accuracy	63.74%	63.84%	63.81%	63.80%	63.79%	
confidence interval	±0.0097	±0.0097	±0.00976	±0.00978	±0.00975	

we have conducted 5way-5shot tiered imagenet using the command:

python main.py –datasource=tieredimagenet –logdir=logs/tieredimagenet5way5shot/ –metatrain_iterations=60000 –meta_batch_size=4 –update_batch_size=5 –update_lr=0.01 –num_classes=5 –num_filters=32 –max_pool=True –num_updates=3 –attempt_rounds=3

Table 8: 5way-5shot tiered imagenet

update	update 0	update 1	update 2	update 3	update 4	update 5
accuracy	19.99%	49.15%	54.89%	60.74%	62.33%	63.34%
confidence interval	±0.000000091	±0.0087	±0.009997	±0.010313	±0.01039	±0.01025
update	update 6	update 7	update 8	update 9	update 10	
accuracy	63.46%	63.58%	63.59%	63.64%	63.71%	
confidence interval	±0.0103	±0.0103	±0.0102	±0.0102	±0.0103	

5 Conclusion

Our iMAML algorithm searches several optimal values from different initial points and only keep the best one to avoid the bad local optimal points. Our algorithm is tested on few-shot classification datasets, Omniglot, miniImagenet and tieredImagenet, and compared the accuracy of our algorithm with the baseline of MAML's in 5-way 1-shot, 5-way 5-shot, 20-way 1-shot and 20-way 5-shot classification experiments. The results show that our method improves the classification accuracy.

6 Acknowledgement

Thanks for this class statistical learning, teacher Newly and teaching assistant Cong, after a semester of statistical learning courses, We feel that we have a deeper understanding of various statistical learning methods.This project involves using deep learning framework to tackle particular problem, and this is a good exercise for converting what we have learned into practice. We have benefited a lot from the study of this course, thank the teacher and teaching assistance again.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017). arXiv: `1703.03400`. URL: `http://arxiv.org/abs/1703.03400`.

[2] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic Model-Agnostic Meta-Learning". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* Ed. by Samy Bengio et al. 2018, pp. 9537–9548. URL: `http://papers.nips.cc/paper/8161-probabilistic-model-agnostic-meta-learning`.

[3] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350 (2015), pp. 1332–1338.

[4] Xinzhe Li et al. "Learning to Self-Train for Semi-Supervised Few-Shot Classification". In: *Advances in Neural Information Processing Systems.* Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 10276–10286. URL: `https://proceedings.neurips.cc/paper/2019/file/bf25356fd2a6e038f1a3a59c26687e80-Paper.pdf`.

[5] Zhenguo Li et al. "Meta-SGD: Learning to Learn Quickly for Few Shot Learning". In: *CoRR* abs/1707.09835 (2017). arXiv: 1707.09835. URL: http://arxiv.org/abs/1707.09835.

[6] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *CoRR* abs/1803.02999 (2018). arXiv: 1803.02999. URL: http://arxiv.org/abs/1803.02999.

[7] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *In International Conference on Learning Representations (ICLR)*. 2017.

[8] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115 (2015), pp. 211–252.

[9] Andrei A. Rusu et al. "Meta-Learning with Latent Embedding Optimization". In: *CoRR* abs/1807.05960 (2018). arXiv: 1807.05960. URL: http://arxiv.org/abs/1807.05960.

[10] Adam Santoro et al. "One-shot Learning with Memory-Augmented Neural Networks". In: *CoRR* abs/1605.06065 (2016). arXiv: 1605.06065. URL: http://arxiv.org/abs/1605.06065.

[11] Qianru Sun et al. "Meta-Transfer Learning for Few-Shot Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[12] Flood Sung et al. "Learning to Compare: Relation Network for Few-Shot Learning". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 1199–1208. DOI: 10.1109/CVPR.2018.00131. URL: http://openaccess.thecvf.com/content%5C_cvpr%5C_2018/html/Sung%5C_Learning%5C_to%5C_Compare%5C_CVPR%5C_2018%5C_paper.html.

[13] Hung-Yu Tseng et al. "Cross-Domain Few-Shot Classification via Learned Feature-Wise Transformation". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SJl5Np4tPr.

[14] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 3630–3638. URL: http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.

[15] Ling Yang et al. "DPGN: Distribution Propagation Graph Network for Few-Shot Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 13387–13396. DOI: 10.1109/CVPR42600.2020.01340. URL: https://doi.org/10.1109/CVPR42600.2020.01340.

[16] Jaesik Yoon et al. "Bayesian Model-Agnostic Meta-Learning". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 7343–7353. URL: http://papers.nips.cc/paper/7963-bayesian-model-agnostic-meta-learning.