UDC 004.93

# FASTER OPTIMIZATION-BASED META-LEARNING ADAPTATION PHASE

**Khabarlak K. S.** – Post-graduate student of the Department of System Analysis and Control, Dnipro University of Technology, Dnipro, Ukraine.

## ABSTRACT

**Context.** Neural networks require a large amount of annotated data to learn. Meta-learning algorithms propose a way to decrease number of training samples to only a few. One of the most prominent optimization-based meta-learning algorithms is MAML. However, its adaptation to new tasks is quite slow. The object of study is the process of meta-learning and adaptation phase as defined by the MAML algorithm.

**Objective.** The goal of this work is creation of an approach, which should make it possible to: 1) increase the execution speed of MAML adaptation phase; 2) improve MAML accuracy in certain cases. The testing results will be shown on a publicly available few-shot learning dataset CIFAR-FS.

**Method.** In this work an improvement to MAML meta-learning algorithm is proposed. Meta-learning procedure is defined in terms of tasks. In case of image classification problem, each task is to try to learn to classify images of new classes given only a few training examples. MAML defines 2 stages for the learning procedure: 1) adaptation to the new task; 2) meta-weights update. The whole training procedure requires Hessian computation, which makes the method computationally expensive. After being trained, the network will typically be used for adaptation to new tasks and the subsequent prediction on them. Thus, improving adaptation time is an important problem, which we focus on in this work. We introduce Λ (lambda) pattern by which we restrict which weight we update in the network during the adaptation phase. This approach allows us to skip certain gradient computations. The pattern is selected given an allowed quality degradation threshold parameter. Among the pattern that fit the criteria, the fastest pattern is then selected. However, as it is discussed later, quality improvement is also possible is certain cases by a careful pattern selection.

**Results.** The MAML algorithm with Λ pattern adaptation has been implemented, trained and tested on the open CIFAR-FS dataset. This makes our results easily reproducible.

**Conclusions.** The experiments conducted have shown that via Λ adaptation pattern selection, it is possible to significantly improve the MAML method in the following areas: adaptation time has been decreased by a factor of 3 with minimal accuracy loss. Interestingly, accuracy for one-step adaptation has been substantially improved by using Λ patterns as well. Prospects for further research are to investigate a way of a more robust automatic pattern selection scheme.

**KEYWORDS:** few-shot learning, meta-learning, Model-Agnostic Meta-Learning, MAML, adaptation time, adaptation speed, optimization-based meta-learning.

## ABBREVIATIONS

MAML is Model-Agnostic Meta-Learning, a method of optimization-based few-shot learning;

ResNet is a Residual Network, a particular architecture of Convolutional Neural Networks;

NLP is Natural Language Processing.

## NOMENCLATURE

$N$ is a number of images per class that are given for the network training.

$K$ is a number of classes the network is trained to distinguish between.

$X$ is a network input, in our case images.

$\Phi(\theta, X)$ is a neural network.

$\theta$ is a matrix of network weights.

$B$ is a number of layers in the neural network.

$\rho(T)$ is a distribution of all tasks.

$T_\mathrm{i}$ is one of the tasks, consisting of Support Set $S_i$, Query Set $Q_i$.

$P$ is a number of adaptation steps.

$\theta_i^{(j)}$ is a matrix of adapted weights after $j$ iterations that correspond to $i$th task.

$\alpha$ is an adaptation step size, $\alpha > 0$.

$\beta$ is a learning rate, $\beta > 0$.

$\Lambda$ is an adaptation template, which controls which neural network layers should be updated during the adaptation procedure to the current task $T$.

## INTRODUCTION

The neural network accuracy for image classification has significantly improved thanks to deep convolutional neural networks. However, a very large number of images is required for such networks to train successfully. For instance, all of the ResNet [1] neural network configurations from ResNet-18 to ResNet-152 (18 and 152 layers deep correspondingly) are trained on the ImageNet dataset [2], which contains 1.281.167 images and 1.000 classes (about 1.200 samples per class). Obviously, for many of the practically significant tasks it is impossible to collect and label a dataset that large. Thus, learning deep convolutional networks from scratch might yield poor results. Because-of that, on the smaller datasets typically an approach called transfer learning is used instead. That is, an ImageNet pretrained network of a particular architecture is taken and then further finetuned on the target (smaller) dataset [1; 3; 4]. However, training on few examples per class is still a challenge. This contrasts to how we, humans, learn, when even a single example given to a child might be enough. Also, it is hard to estimate the

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

quality of a certain ImageNet pretrained network on the target dataset. Hence, we get a model selection problem: if the model $A$ is better than the model $B$ on ImageNet, will it be better on our small dataset? A promising approach to resolving both of these problems is to use meta-learning or its benchmark known as few-shot learning. Meta-learning trains the network on a set of different tasks, which are randomly sampled from the whole space of tasks. By learning the network in such a way, it is assumed that the network will learn features that are relevant to all of the tasks and not only to the single one, i.e., will learn more general features.

In this work we focus on one of the most prominent optimization-based meta-learning methods, called MAML [5]. This method has become a keystone, and as it will be shown in the literature overview section, many of the newer method base on its ideas. Training of the MAML method is split into the so-called adaptation and meta-gradient update phases.

**The subject of study** is the class of optimization-based meta-learning algorithms.

It has been shown that adaptation phase of the MAML is quite slow to perform [6], and in general, high neural network execution speed is a major problem for applications [7]. In this work we introduce gradient update patterns, i.e., a selective update of the neural network weights during the adaptation phase.

**The purpose of this work** is to show that by carefully selecting the newly-proposed gradient update pattern, it is possible to: 1) increase the execution speed of MAML adaptation phase; 2) significantly improve MAML performance in case, when only 1 adaptation phase is used. The testing results will be shown on a publicly-available few-shot learning dataset CIFAR-FS [8].

## 1 PROBLEM STATEMENT

The goal behind meta-learning is to train a neural network Φ(θ), that is capable of adapting to the new previously unknown tasks given a small number of examples. Meta-learning is also said to be learning to learn problem. The training procedure is defined using a concept of tasks, that are sampled from the whole task space $\rho(T)$ of the problem domain. The task is a tuple $T = \{S, Q\}$, consisting of the so-called Support Set $S = \{X_S, y_S\}$ and Query Set $Q = \{X_Q, y_Q\}$ [5; 9–11]. In literature, the Query Set is also sometimes referred to as Target Set. Support Set $\{X_S, y_S\}$ is used to adapt (or train) the network to the new task. The set $S$ is small. $X_S$ are the network inputs, $y_S$ – the expected predictions. The number of examples per class is denoted as $K$ and written as $K$-shot. $K$ is typically in range from 1 to 20, although no hard upper-bound is defined. $X_Q, y_Q$ are the query inputs and expected outputs correspondingly. Number of classes $N$ the network should distinguish between is denoted as $N$-way.

We have given the general training procedure, next we define it in more detail for image classification optimization-based meta-learning, which this paper is focused on. Optimization meta learning is defined in 2 steps: 1) adaptation step, which computes adaptation weights in a form of function θ'(θ), that minimize task-specific error $L(y_s, \Phi(\theta', X_S))$; 2) meta-gradient update, which updates meta-weights θ. The idea behind such training procedure is that by finding good weights θ, it will be possible to adapt to new previously unseen tasks with few training examples in the adaptation procedure. For classification, the loss function used is typically cross-entropy (1):

$$L(y, \Phi(\theta, X)) = -\sum_i y_i \log \Phi(\theta, X_i). \tag{1}$$

We define the algorithm-specific part in the Materials and Methods section. In this work we set a goal of improving adaptation step execution time and accuracy.

## 2 REVIEW OF THE LITERATURE

The meta-learning approaches are mainly divided into 3 broad categories [12]: metric-based, model-based and optimization-based. Representatives of each group differ in the neural network design and training procedure. In this work we focus on classification methods, yet applications exist in literally every field of machine learning [5; 13–15], such as NLP, Reinforcement Learning, Face Verification, etc.

Next, we describe each category of meta-learning methods. 1) In metric-based methods the goal is to define a neural network architecture that produces an embedding into a metric space and a similarity measure (metric), so that the distance between embeddings of the same class is smaller than that of different classes. Examples of such methods include Siamese Networks [16], Matching Networks [17], Prototypical Networks [9]. 2) In model-based methods the network architecture is designed, so that the model has explicit memory cells, which help the network to adapt quickly, for instance, Memory-Augmented Neural Networks [18]. 3) In optimization-based learning the network architecture is not changed, which means that conventional architectures for image classification can be used. One of the quintessential methods in this category is MAML [5], which defines the training procedure as a 2nd-order optimization problem. The method applicability has been shown in regression, classification and reinforcement learning. Two popular datasets were considered for image classification: Omniglot [19] and miniImageNet [10; 17], where MAML has beaten with a margin many of the previous methods. After MAML has been introduced, a lot of works have proposed its modifications. Reptile [20] has simplified MAML training scheme, MAML++ [11] has given practical recommendation on improving MAML training stability. In has been noted that while MAML++ has introduced more parameters to the network, total training time has decreased thanks to the performance optimizations proposed. Authors of Meta-SGD [21] note that by learning not only network weights, but also separate update coefficient for each of the weights, it is possible to achieve higher accuracies. However, the network training time and memory con-

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

sumption has significantly increased as twice the number of the parameters should be optimized.

In contrast to previous works, in this paper we focus on improving the network adaptation and not training time. We assume that after the initial training, the network can be adapted to multiple tasks in an online format. Thus, minimizing adaptation time is an important problem. The results obtained in the paper will be applicable to many of the optimization-based algorithms, including but not limited to the ones mentioned above.

## 3 MATERIALS AND METHODS

In this work we propose a modification to the MAML algorithm. As we have described in the problem statement section above, this class of algorithms is defined in terms of adaptation and meta-gradient update phases.

The algorithm starts by randomly sampling a training task $T_i \sim \rho(T)$. To sample a task $T_i$ means to 1) randomly select $N$ classes from all classes that are available in the dataset split (training, validation or test, based on which accuracy we want to compute); 2) randomly select $K$ images per each of $N$ classes for the Support Set and $K_Q$ images per each class $N$ for the Query Set. The first phase of the algorithm is adaptation, where MAML minimizes loss function (1) on the Support Set by performing several stochastic gradient descent steps. To do that the algorithm iteratively builds model weights $\theta_i^{(j)}(\theta)$ via formula (2), note that $\theta_i^{(0)} \equiv \theta$:

$$\theta_i^{(j)} = \theta_i^{(j-1)} - \alpha \nabla_{\theta_i} L\left(y_{S_i}, \Phi\left(\theta_i^{(j-1)}, X_{S_i}\right)\right) \quad (2)$$

Having iteratively built the task specific weights $\theta_i^{(j)}$, the algorithm updates the meta-weights $\theta$ using formula (3):

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{Q_i \in T_i} L\left(y_{Q_i}, \Phi\left(\theta_i^{(P)}, X_{Q_i}\right)\right) \quad (3)$$

In essence, in (3) the algorithm updates the meta-weights $\theta$ by averaging computed loss function (1) on the Query Set, for the neural networks $\Phi$ with weights $\theta_i^{(P)}$ on several tasks $T_i$, i.e., in this step the algorithm backpropagates through the losses of all the task-specific adaptations. Throughout the paper we use 4 tasks for the meta-update step. Note, that in (2) task-specific weights $\theta_i^{(j)}$ are computed on the Support Set, and in (3) Query Set is used for the loss computation. Also, in contrast to the conventional neural network training procedure the loss function is computed twice: first, to compute the adaptation weights $\theta_i^{(P)}$ in (2); second, to compute the resulting adaption loss in (3). Also, in (2) the gradient is taken by task-specific weights $\theta_i^{(j-1)}$ from previous step, and in (3) the gradient is taken by meta-weights $\theta$. Thus, as can be seen from formulas (2), (3), the method requires Hessian computation during the meta-gradient update, hence, this is a second-order optimization method. The whole training procedure can be seen in algorithm 1. A more detailed information can be found in the original paper [5].

Algorithm 1. MAML adaptation procedure

```
1:  Randomly sample task $T_i$ from task space $\rho(T)$
2:  For each task $T_i=\{S_i, Q_i\}$, where $S_i = \{X_S, y_S\}$, $Q_i = \{X_Q, y_Q\}$
3:      For iteration $j = \{1, \ldots, P\}$
4:          Adapt the network via formula (2) using $S_i$
5:      End for
6:  End for
7:  Update meta-weight $\theta$ via (3) using $Q_i$ and the task specific
    weights $\theta_i^{(P)}$
```

Next, we define our modified adaptation procedure. Given a convolutional neural network that has $B$ layers, we define an adaptation pattern (4), where $\Lambda_j$ is an indicative function as defined in (5), which indicates layers of the network that should be updated during backpropagation.

$$\Lambda = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_B\}, \quad (4)$$

$$\forall l : \Lambda_l = \begin{cases} 1, \textit{if layer l is updated}, \\ 0, \textit{otherwise}. \end{cases} \quad (5)$$

We say that pattern is full if $\forall l : \Lambda_l = 1$. In this case our adaptation phase will be equivalent to the one proposed in MAML. We consider all possible patterns $\Lambda$, except $\forall l : \Lambda_l = 0$, when no weights can be updated, thus, no adaptation is possible. We assume that updating only certain weights might be useful, because the neural networks tend to learn features that differ in complexity, the closer the layer is to the input the simple the features are [22]. Also, authors of Meta-SGD [21] have shown that by learning weight-specific learning rates the resulting quality was superior to the original MAML algorithm. However, Meta-SGD approach was much slower to train as both weights and learning rates have to be learned during the training procedure. Training time in our approach is intact. In contrast to previous works, we propose to update only certain weights, thus, essentially freezing some layers. This allows us to decrease gradient computations required during the adaptation phase as is shown on Fig. 1 for a convolutional network that contains 4 convolutional and a single fully-connected (linear) layer.

In Fig. 1 the backpropagation pass goes in the direction opposite to arrows (forward pass). The architecture is taken as an example and can be arbitrary in practice. For the example pattern $\Lambda = \{0,1,0,1,1\}$, we can see that for the Convolutional Block 4 and the Linear layers both the gradient is computed and the weights are updated. For Convolutional Block 3 gradients are computed, but weights are not updated as Convolutional Block 2 requires weight update. However, for Convolutional Block 1 no gradients computation or weight update are performed.

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

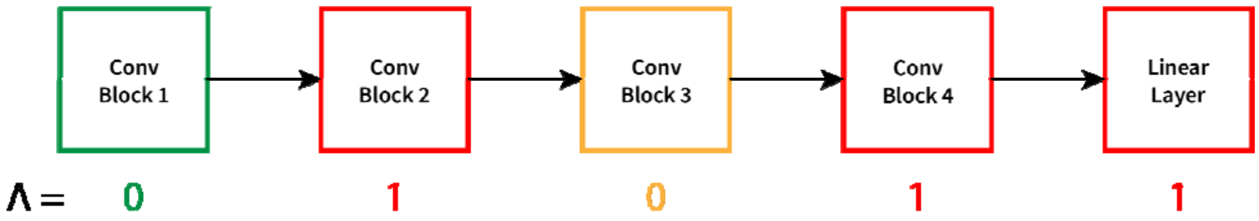


Figure 1 – Λ pattern backpropagation scheme. Backpropagation is performed in order reverse to the arrows. In red – gradients are computed, networks weights are updated; yellow – gradients are computed, no network weight update; green – both gradient computation and network weight update are skipped

Given the above-described Λ pattern description, the updated adaptation formula will look as follows (6):

$$\theta_i^{(j)} = \theta_i^{(j-1)} - \Lambda\alpha\nabla_{\theta_i} L\left(y_{S_i}, \Phi\left(\theta_i^{(j-1)}, X_{S_i}\right)\right) \quad (6)$$

## 4 EXPERIMENTS

To conduct the experiments, we have reimplemented the MAML algorithm. The following paragraph describes the details.

The authors of MAML have defined convolutional neural network architecture and have used it for miniImageNet experiments. This network is commonly referred to as “CNN4” in the later meta-learning literature. It has 4 convolutional blocks, followed by a linear layer. Each of the blocks has a convolutional layer with kernel size of 3 and padding of 1, followed by the Batch Normalization [23], ReLU activation and Max Pooling with kernel size of 2. Number of filters in the convolutional layers is a configurable parameter, the authors have used 32, which we follow. Number of outputs in the linear layer is defined by $K$ for $K$-way classification problem. Training is performed via Adam [24] gradient descent method as meta-optimizer with learning rate of $\beta = 10^{-3}$ and $\alpha = 0.01$ as the adaptation step size. Each model has been trained for 600 epochs. While the authors used meta-batch size of 2 for 5-shot and 4 for 2-shot experiment to reduce training memory consumption, we stick to 4 as it leads to slightly better performance on CIFAR-FS [8] dataset during our experiments. Also, the dataset memory footprint is small, so we don’t have to reduce memory consumption by using a smaller batch-size. Each epoch has 100 randomly sampled tasks. For the gradient update $N \cdot K$ samples are taken for $N$-shot $K$-way classification problem for training and 15 samples per class for evaluation, thus following [10].

In addition, we have modified the network adaptation procedure, so that it updates only weights defined by pattern Λ as defined in (4)–(6).

For the experiments we have used the novel CIFAR-FS [8] dataset. It has been constructed from a well-known classification dataset, called CIFAR-100 [25]. It has images of different kinds of mammals, reptiles, flowers, man-made things, etc. The images are in color and have a size 32x32. Originally, this dataset was not supposed to be used in a few-shot learning setting. In [8] it has been suggested to split 100 classes into train, validation and test sets. If it has been the non-few-shot neural network training, we would expect all of the 100 classes to be represented in each of the sets, only the images themselves would have been split. However, in few-shot learning case different disjoint classes are taken. Thus, 64 training, 16 validation and 20 test set classes have been selected. The exact classes that go into each split are important for testing the resulting accuracy and are defined in [8]. By using different classes for training and testing, the adaptation to the new classes can be better estimated. After such training the model is expected to quickly adapt to the new, unseen classes. We have taken the CIFAR-FS dataset for our experiments as it hasn’t been analyzed by the MAML authors and is also faster to compute than miniImageNet.

All of the training procedures and time measurements were done on our own MAML implementation and tested on NVIDIA GTX 1050Ti GPU.

## 5 RESULTS

Given the network configuration as described in the experiments section, we have implemented the MAML algorithm. CIFAR-FS accuracy and adaptation timings are presented in Table 1.

Table 1 – Accuracies and adaptation timings on CIFAR-FS dataset

| | 1-shot 2-way | 5-shot 2-way | 1-shot 5-way | 5-shot 5-way |
|---|---|---|---|---|
| Accuracy | 77.2% | 87.6% | 51.7% | 70.3% |
| Time | 38.43 ms | 40.70 ms | 41.67 ms | 45.35 ms |

In (6) we have proposed a modified adaptation scheme, where only a part of weights is updated during the adaptation procedure. To begin with, we consider only trivial patterns $\Lambda$, where only one network layer is updated during the adaptation procedure. We show the accuracy on the test set in Fig. 2, where in a and b we conduct the experiment for 1-shot 5-way and 5-shot 5-way configurations correspondingly. To see the impact of the number of adaptation steps, we also show the accuracies for $P = 10$ (default) and 1, 3, 5 adaptation steps. As it can be seen, the model accuracy differs significantly between the configurations. For 1-shot 5-way, learning one of the three first convolutional layers only has no effect, the accuracy remains on the level of random guessing (20%). However, training either convolutional layer 4 or the last linear layer improves the model accuracy. Note, that the number of parameters in layers differs. In Table 2, we

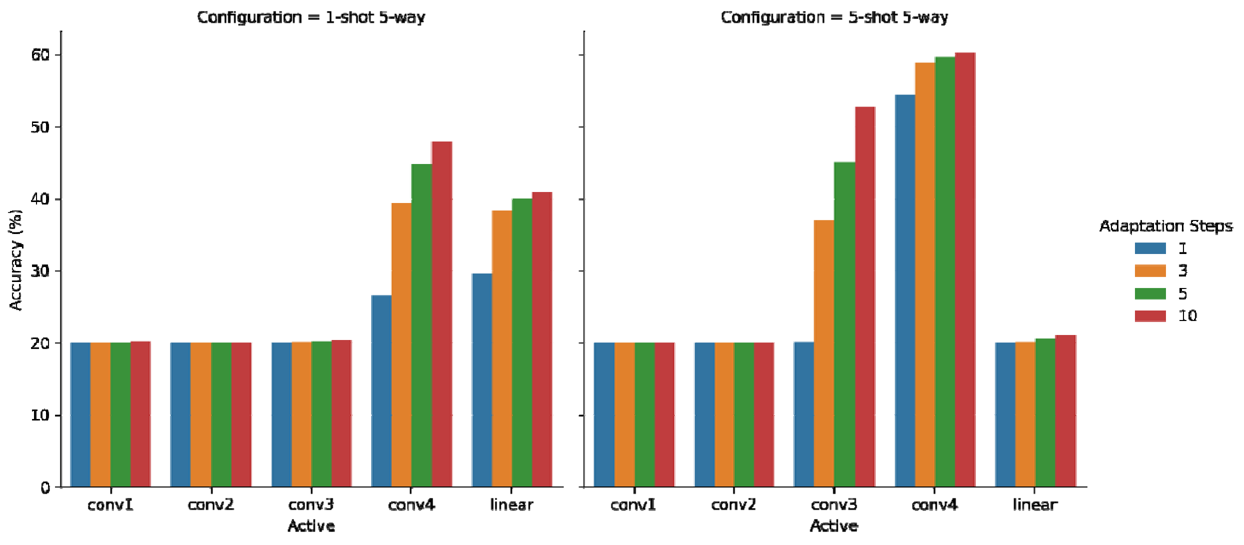
© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

show the number of parameters for each layer. Note that final layer has different number of parameters depending on *N* output classes. It can be seen that the first convolutional layer and the final linear (fully-connected) layers have fewer parameter than inner convolutional blocks. This can explain the fact that learning only the linear layer has worse performance. For 5-shot 5-way we see that only convolutional layers 3 and 4 have a positive impact on the performance if adapted alone. Interestingly, number of adaptation steps has a significant impact on the performance with only convolutional layer #3 enabled. As we will see later, such an impact is higher, than when the full network is updated during the adaptation.

In Fig. 3, a and b we depict a similar experiment for 1-shot 2-way and 5-shot 2-way configurations correspondingly. Note, that random guessing baseline for these configurations is now at 50%, so the lower bound for accuracy is now higher than in Fig. 2. Here we see an opposite trend, where updating the first layers also has a positive impact on the resulting accuracy. Contrasting to previous experiment, updating the Convolutional Block 4 only doesn't provide the best results in either case.

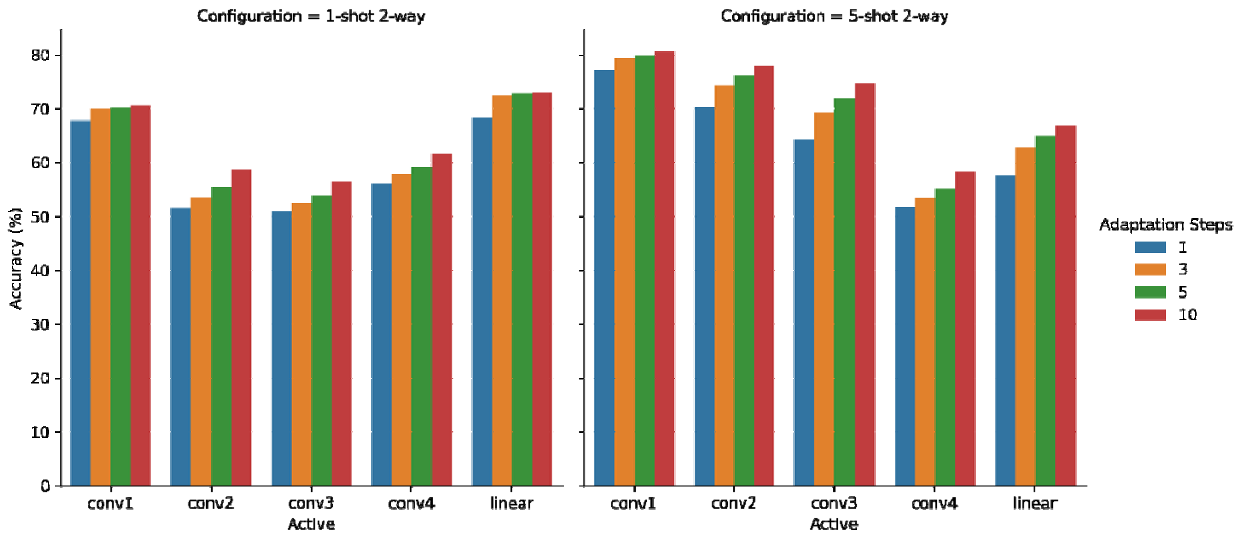
Table 2 – Number of parameters for each layer

| Layer Name | Number of Parameters |
|---|---|
| Conv Block 1 | 960 |
| Conv Block 2 | 9.312 |
| Conv Block 3 | 9.312 |
| Conv Block 4 | 9.312 |
| Linear | 1.602 (2-way)<br>4.005 (5-way) |
| Total | 30.498 (2-way)<br>32.901 (5-way) |



a b

Figure 2 – Adaptation accuracy for trivial Λ patterns, i.e., only a single layer is updated during adaptation: a is 1-shot 5-way, b is 5-shot 5-way



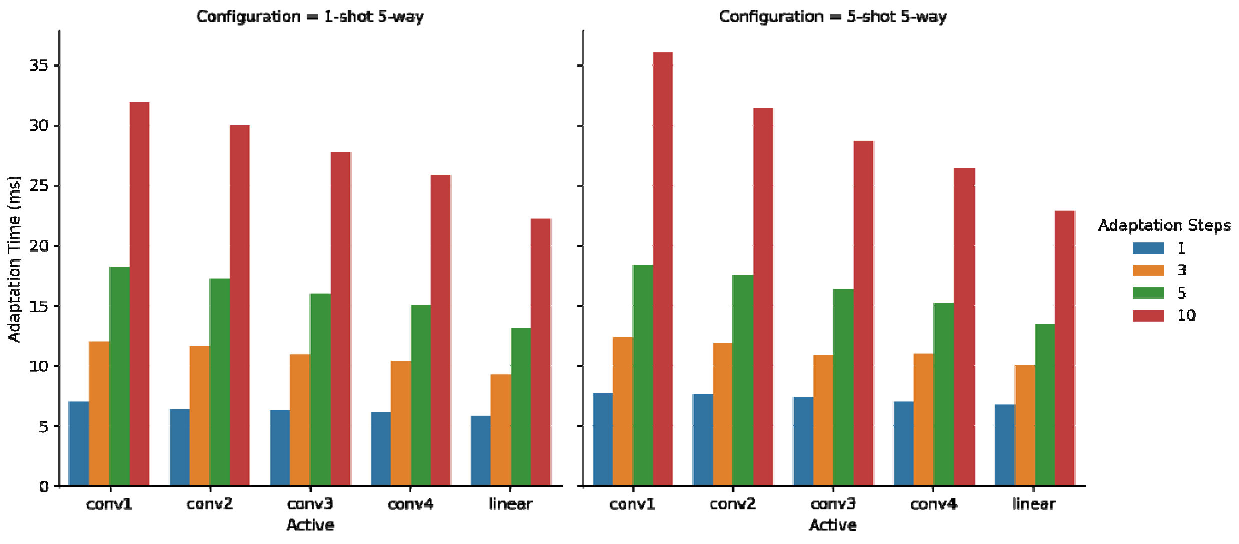
a b

Figure 3 – Adaptation accuracy for trivial Λ patterns, i.e., only a single layer is updated during adaptation: a is 1-shot 2-way, b is 5-shot 2-way

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

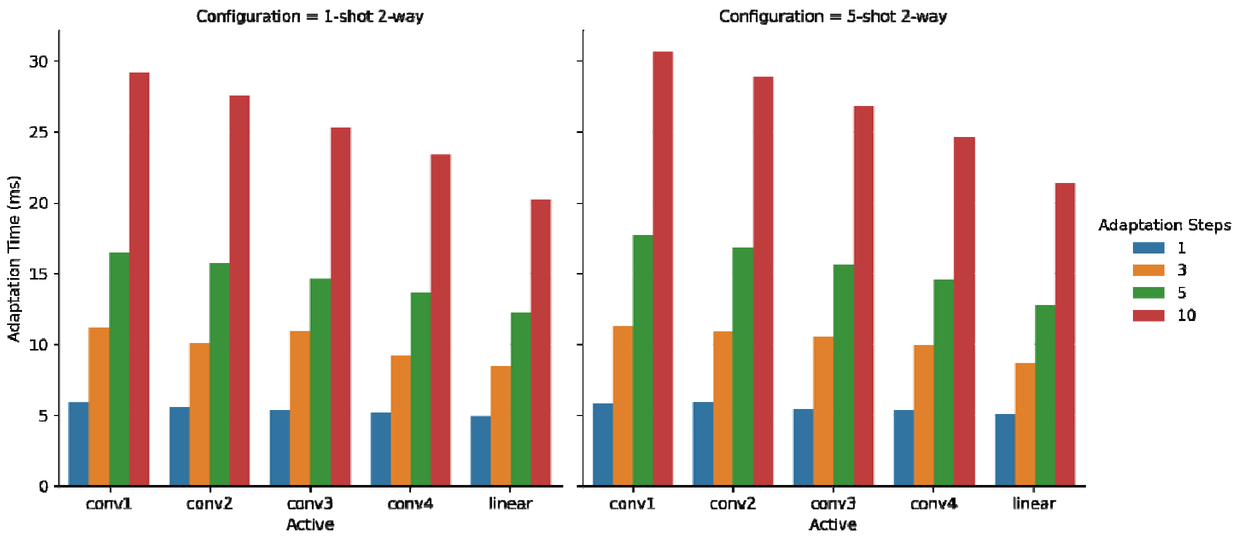
As one of the goals in our work is to improve the model adaptation speed, we have timed experiments for trivial patterns Λ. On Fig. 4 and 5 we show the model adaptation time corresponding to all of the four configurations depicted on Fig. 2 and 3. As we can see, in both cases we have a similar trend where the closer the layer we update to the end of the network, the smaller the adaption time is. This follows our previous idea that by skipping some gradient computations (as have been shown on Fig. 1), adaptation time can be reduced.

As can be seen from Fig. 4 and 5, number of adaptation steps has a significant impact on the adaptation speed. On Fig. 6 we show the model accuracy for each of the four scenarios and on Fig. 7 we depict the corresponding timings, both shown with respect to the number of the adaptation steps. As before, the experiments have been conducted for $P = 1, 3, 5$ and 10 adaptation steps. The results between those reference points have been linearly interpolated. The presented accuracies and timings are the average taken for all 31 possible patterns Λ. Note, that throughout the article we exclude pattern $\forall l : \Lambda_l = 0$, as no weights can be changed for such pattern, therefore no adaptation is possible. As can be seen, while the adaptation time grows linearly with the number of adaptation steps, the accuracy growth plateaus at around 5 adaptation steps. Actually, for the full pattern Λ increasing number of adaptation steps from 5 to 10 has less than 0.3% improvement in accuracy. In typical practical scenarios such an improvement is insignificant. Thus, we suggest that performing 10 adaptation steps is redundant.



a b

Figure 4 – Adaptation time for trivial Λ patterns:
a is 1-shot 5-way, b is 5-shot 5-way



a b

Figure 5 – Adaptation time for trivial Λ patterns:
a is 1-shot 2-way, b is 5-shot 2-way

DOI 10.15588/1607-3274-2022-1-10

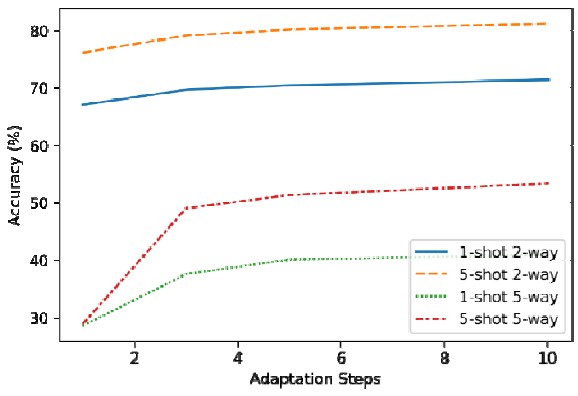


Figure 6 – Accuracy averaged for all patterns Λ for different *N*-shot *K*-way problems with respect to the number of adaptation steps *P*

Next, we try to search for such a pattern Λ and number of adaptation steps, so that the resulting accuracy drops no more than 0.07 times the full pattern accuracy. We see such a quality degradation threshold reasonable for practical applications. It should be noted that the approach we propose can be applied with an arbitrary quality degradation threshold. We show such patterns in table 3. Based on this table, we suggest using the $\Lambda^* = \{1,0,1,1,1\}$, which offers factor of 3.0 speed improvement with an insignificant quality loss. It can be seen that pattern $\Lambda = \{0,1,1,1,1\}$ also suits the specified criteria and also has a slightly higher (factor of 3.1) performance improvement, however, it has a significantly lower performance for both of the 2-way configurations, degrading on 2.5% and 3.2% relative to the best selected pattern $\Lambda^*$. We consider such a degradation not worth the speed up. The fact that enabling first CNN layer is significant for the 2-way learning accuracy, closely follows the presented above description of Fig. 3. Also, not to be mistaken, in Fig. 2–5, we had only one layer updated during the adaptation phase (thus $\Sigma_l \Lambda_l = 1$), however, the best selected pattern $\Lambda^*$ has all expect one layer updated.

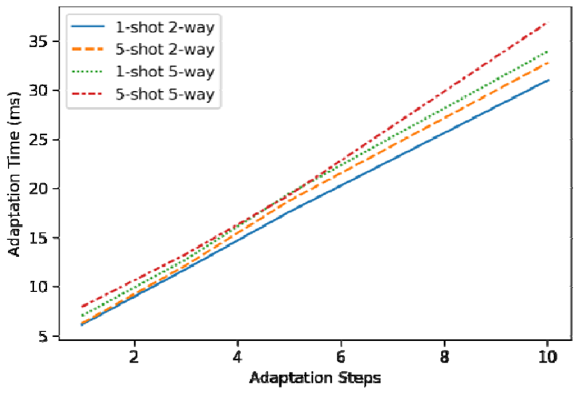


Figure 7 – Adaptation time averaged for all patterns Λ for different *N*-shot *K*-way problems with respect to the number of adaptation steps *P*

Finally, we pose a question, whether updating only part of weights in the neural network can improve the method performance. We have discovered, that in extreme case of learning with a single adaptation step ($P = 1$), we have significant improvement in 5-way adaptation performance by updating with a partial pattern Λ. The performance for the full pattern, as well as a partial, is shown in table 4.

Table 3 – Adaptation speedup depending on pattern Λ and the number of adaptation steps. Patterns with loss degradation of less than 7% (relative to full pattern Λ and 10 adaptation steps) are shown.

| Adaptation Steps | Pattern Λ | 1-shot 2-way (%) | 5-shot 2-way (%) | 1-shot 5-way (%) | 5-shot 5-way (%) | Mean Adaptation Time (ms) | Relative Speedup (times) |
|---|---|---|---|---|---|---|---|
| 3 | 0,1,1,1,1 | 74.7 | 83.2 | 49.3 | 69.7 | 13.3 | 3.1 |
| 3 | 1,0,1,1,1 | 76.6 | 85.9 | 49.3 | 69.8 | 13.9 | 3.0 |
| 3 | 1,1,1,1,1 | 76.6 | 87.2 | 49.3 | 70.0 | 15.0 | 2.8 |
| 5 | 0,1,1,1,1 | 75.2 | 83.9 | 51.5 | 69.9 | 20.0 | 2.1 |
| 5 | 1,0,1,1,1 | 76.9 | 86.2 | 51.4 | 70.1 | 21.1 | 2.0 |
| 5 | 1,1,1,1,1 | 77.0 | 87.4 | 51.6 | 70.2 | 22.6 | 1.8 |
| 10 | 0,1,1,1,1 | 75.4 | 84.6 | 51.7 | 70.1 | 36.1 | 1.2 |
| 10 | 1,0,1,1,1 | 77.1 | 86.6 | 51.7 | 70.1 | 38.6 | 1.1 |
| 10 | 1,1,1,1,1 | 77.2 | 87.6 | 51.7 | 70.3 | 41.5 | 1.0 |

Table 4 – Accuracy improvement for $P = 1$ gradient step adaptation with pattern selection

| Accuracy | 1-shot 2-way | 5-shot 2-way | 1-shot 5-way | 5-shot 5-way |
|---|---|---|---|---|
| $\Lambda = \{1,1,1,1,1\}$ | 74.3% | 86.0% | 36.8% | 20.4% |
| $\Lambda = \{1,1,0,1,1\}$ | 74.3% | 83.1% | 36.9% | 53.1% |

We have also performed a search of all cases, when our approach gives better results than the original with $P = 1$. The results are shown in Table 5.

Table 5 – Accuracy improvement for $P = 1$ gradient step adaptation with pattern selection if pattern is selected per configuration

| | 1-shot 2-way | 5-shot 2-way | 1-shot 5-way | 5-shot 5-way |
|---|---|---|---|---|
| Accuracy on $\Lambda = \{1,1,1,1,1\}$ | 74.3% | 86.0% | 36.8% | 20.4% |
| Accuracy on selected $\Lambda$ | 74.5% | 86.2% | 36.9% | 54.8% |
| Selected Pattern $\Lambda$ | 1,1,1,0,1 | 1,1,1,0,1 | 1,1,0,1,1 | 1,1,0,1,0 |

## 6 DISCUSSION

In [22] it has been shown that each trained neural network's convolutional layer has a different meaning. The first layer tends to learn simple features, like edges, lines or color gradients. The second layer increases its complexity and understands simple shapes, e.g., circles, corners or stripes, while the last layers learn high-level features, such as eyes, faces, text-like objects, etc. The exact features learned, obviously, depend on the training dataset, however, such logic is retained. In the few-shot learning classification scenario the tasks differ by the types of objects that the model has to classify (e.g., horse, vehicle, frog, etc.). As we have described in the experiments section, train and test sets have different disjoint classes presented. Thus, it might be reasonable to expect that only the last layers of the network should be changed to adapt to the new tasks and classes. This is exactly what we see in the case of 5-way classification as is shown on Fig. 2. However, such a statement contradicts to the experiment results from Fig. 3. By examining the original CIFAR-100 dataset, we can see that image labels (classes) form larger coarse groups. For instance, coarse class (or superclass) "aquatic mammals" contains "beaver", "dolphin", "otter", "seal", "whale". Other examples of superclasses include "fish", "large carnivores", "household electrical devices", etc. The training itself is performed on finer classes. From the examples we have picked, it becomes obvious that instances of different classes have a significant variation in color. Images of aquatic mammals and fish typically contain blue and gray colors, while large carnivores might have more yellow and green. Thus, in case of 2-way classification it is more probable that both classes will be picked from a single or several similar superclasses than in case of 5-way classification. Consequently, we suggest that updating the first layer of a neural network in a 2-way few-shot learning scenario adjusts the feature distribution to the one expected by the following neural network layers. We see this as an analogy of how a human eye works: it adjusts the amount of light coming to the retina by expanding or contracting the pupil, so that it becomes easier to see the details.

From Table 3 we see that keeping the inner layers stale is the most fruitful way to improve the performance, with little to no quality loss. A substantial increase in adaptation speed has been achieved with a target quality loss set to 7% relative to the original pattern $\Lambda = \{1,1,1,1,1\}$ and $P = 10$ adaptation steps. The actual quality loss turns out to be even smaller as we have skipped slightly faster, but worse pattern $\Lambda = \{0,1,1,1,1\}$. Thereby, with the best $\Lambda^* = \{1,0,1,1,1\}$ and $P = 3$ adaptation steps, we achieve a factor of 3.0 speed improvement. Our quality losses are the following: 1-shot 2-way is 0.78%, 5-shot 2-way is 1.97% 1-shot 5-way is 4.86% and 5-shot 5-way is 0.71%. Even smaller quality losses can be achieved by consulting table 3. Note, that these are relative quality losses. If the losses are computed in absolute terms, they become even more negligible. Thus, we state that have achieved a significant adaptation time reduction with small-enough quality loss.

We also discuss a way to improve algorithm quality by selecting a pattern $\Lambda$. In an extreme case of single adaptation step, avoiding to update the inner layer has helped to improve the overall model quality as is shown in table 4. We have also been able to find such a pattern for each of the few-shot learning configurations such that it improves the model performance for $P = 1$ adaptation step in table 5. It is curious that no such behavior is observed in cases when $P > 1$. To the best of our knowledge such behavior has not been previously observed and should be further investigated.

## CONCLUSIONS

MAML is an optimization-based few-shot learning method that is able to learn an arbitrary neural network by using only a few samples per class. Many algorithms follow the learning scheme proposed in MAML. In this work we solve the problems of 1) long adaptation time, and 2) poor performance in cases when a single adaptation step is used.

**The scientifical novelty** of obtained results is that the method of reducing number of gradient computations during MAML adaptation phase has been introduced via the newly proposed $\Lambda$ patterns. By selecting an appropriate adaptation pattern, we have significantly improved the method in the following areas: 1) long MAML adaptation time has been decreased by the factor 3 with minimal accuracy loss; 2) accuracy for cases when only a single adaptation step is used has been substantially improved.

**The practical significance** of obtained results is that an improvement of adaptation time of the widespread MAML algorithm will enable applicability of the algorithm on less powerful devices and will in general decrease the time needed for the algorithm to adapt to new tasks.

**Prospects for further research** are to investigate a way of a more robust automatic pattern selection scheme for an arbitrary training dataset and network configuration.

## ACKNOWLEDGEMENTS

The author expresses gratitude to Larysa Koriashkina, PhD, Associate Professor of the Department of System Analysis and Control, Dnipro University of Technology for support and a fruitful paper discussion. The work is supported by the state budget scientific research project of Dnipro University of Technology "Development of new

mobile information technologies for person identification and object classification in the surrounding environment" (state registration number 0121U109787).

## REFERENCES

1. He K., Zhang X. , Ren S. et al. Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR 2016. Las Vegas, NV, USA, June 27–30, 2016. IEEE Computer Society, 2016. pp. 770–778. DOI: 10.1109/CVPR.2016.90.
2. Deng J., Dong W., Socher R. et al. ImageNet: A large-scale hierarchical image database, *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20–25 June 2009*. Miami, Florida, USA, IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.
3. Huang G., Liu Z., Maaten L. et al. Densely Connected Convolutional Networks, 2*017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Honolulu, HI, USA, July 21–26, 2017, IEEE Computer Society, 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.
4. Zagoruyko S., Komodakis N. Wide Residual Networks, *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2016, pp. 87.1–87.12. DOI: 10.5244/C.30.87.
5. Finn C., Abbeel P., Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, Proceedings of Machine Learning Research*. PMLR, 2017, Vol. 70, pp. 1126–1135.
6. Rajeswaran A., Finn C., Kakade S. et al. Meta-Learning with Implicit Gradients, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019*. Vancouver, BC, Canada, 2019, pp. 113–124.
7. Khabarlak K., Koriashkina L. Fast Facial Landmark Detection and Applications: A Survey [Electronic resource], *arXiv:2101.10808 [cs]*, 2021. Access mode: https://arxiv.org/abs/2101.10808
8. [Bertinetto L., Henriques J., Torr P. et al. Meta-learning with differentiable closed-form solvers [Electronic resource], *7th International Conference on Learning Representations, ICLR 2019*. New Orleans, LA, USA, May 6–9, 2019. Access mode: https://openreview.net/forum?id=HyxnZh0ct7.
9. Snell J., Swersky K., Zemel R.S. // Prototypical Networks for Few-shot Learning, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 201*7, December 4–9, 2017. Long Beach, CA, USA, 2017, pp. 4077–4087.
10. Ravi S., Larochelle H. Optimization as a Model for Few-Shot Learning [Electronic resource], *5th International Conference on Learning Representations, ICLR 2017*. Toulon, France, April 24–26, 2017, Conference Track Proceedings. Access mode: https://openreview.net/forum?id=rJY0-Kcll.
11. Antoniou A., Edwards H., Storkey A. J. How to train your MAML [Electronic resource], *7th International Conference on Learning Representations, ICLR 2019*. New Orleans, LA, USA, May 6–9, 2019. Access mode: https://openreview.net/forum?id=HJGven05Y7.
12. Weng L. Meta-Learning: Learning to Learn Fast [Electronic resource]. Access mode: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html.
13. Yin W. Meta-learning for Few-shot Natural Language Processing: A Survey [Electronic resource], *CoRR,* 2020, Vol. abs/2007.09604. Access mode: https://arxiv.org/abs/2007.09604.
14. Wang Y., Yao Q. , Kwok J. et al. Generalizing from a Few Examples: A Survey on Few-shot Learning, *ACM Comput. Surv*, 2020, Vol. 53, No. 3, pp. 63:1–63:34. DOI: 10.1145/3386252.
15. Guo Y., Zhang L. One-shot Face Recognition by Promoting Underrepresented Classes [Electronic resource], *CoRR,* 2017, Vol. abs/1707.05574. Access mode: http://arxiv.org/abs/1707.05574.
16. Koch G., Zemel R., Salakhutdinov R. Siamese neural networks for one-shot image recognition / G. Koch, // *ICML deep learning workshop*. Lille, 2015, Vol. 2.
17. Vinyals O., Blundell C., Lillicrap T. et al. Matching Networks for One Shot Learning, *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.
18. Santoro A., Bartunov S., Botvinick M. et al. Meta-Learning with Memory-Augmented Neural Networks, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19–24, 2016, JMLR Workshop and Conference Proceedings. JMLR.org*, 2016, Vol. 48, pp. 1842–1850.
19. Lake B. M., Salakhutdinov R., Tenenbaum J. B. Human-level concept learning through probabilistic program induction, *Science*, 2015, Vol. 350, No. 6266, pp. 1332–1338. DOI: 10.1126/science.aab3050.
20. Nichol A., Achiam J., Schulman J. On First-Order Meta-Learning Algorithms [Electronic resource], *CoRR,* 2018, Vol. abs/1803.02999. Access mode: http://arxiv.org/abs/1803.02999.
21. Li Z., Zhou F., Chen F. et al. Meta-SGD: Learning to Learn Quickly for Few Shot Learning [Electronic resource], *CoRR*. 2017, Vol. abs/1707.09835. Access mode: http://arxiv.org/abs/1707.09835.
22. Zeiler M.D., Fergus R. Visualizing and Understanding Convolutional Networks, *Computer Vision, ECCV 2014, 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I, Lecture Notes in Computer Science*. Springer, 2014, Vol. 8689, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.
23. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015: JMLR Workshop and Conference Proceedings. – JMLR.org,* 2015*,* Vol. 37, pp. 448–456.
24. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization, *3rd International Conference on Learning Representations, ICLR 2015*. San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015.
25. Krizhevsky A. Learning multiple layers of features from tiny images [Electronic resource], *University of Toronto*, 2009, Access mode: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Received 22.11.2021.
Accepted 14.01.2022.

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

УДК 004.93

## ПРИСКОРЕННЯ ФАЗИ АДАПТАЦІЇ ОПТИМІЗАЦІЙНОГО МЕТА-НАВЧАННЯ

**Хабарлак К. С.** – аспірант кафедри Системного аналізу та управління Національного технічного університету «Дніпровська політехніка», Дніпро, Україна.

### АНОТАЦІЯ

**Актуальність.** Нейронні мережі потребують багато розмічених даних для навчання. Алгоритми мета-навчання пропонують спосіб навчатися лише за декількома прикладами. Один з найзначніших алгоритмів оптимізаційного мета-навчання – це MAML. Однак, його процедура адаптації до нових задач є досить повільною. Об’єктом дослідження є процес мета-навчання та фаза адаптації в тому вигляді, як її визначено в алгоритмі MAML.

**Мета.** Метою даної роботи є створення підходу, що дозволить: 1) зменшити час виконання адаптації алгоритму MAML; 2) покращити якість алгоритму в ряді випадків. Показати результати тестування на публічно доступному наборі даних для мета-навчання CIFAR-FS.

**Метод.** В даній роботі запропоновано покращення алгоритму мета-навчання MAML. Процедура мета-навчання визначається через так звані «задачі». В разі класифікації зображень кожна задача є спробою навчитися класифікувати зображення нових класів лише за декількома навчальними прикладами. В алгоритмі MAML визначено 2 кроки процедури навчання: 1) адаптація до нової задачі; 2) оновлення мета-параметрів мережі. Вся тренувальна процедура потребує обчислення гесіану, що робить метод обчислювально складним. Після навчання мережа, зазвичай, буде використовуватися для адаптації до нових задач та наступної класифікації на них. Таким чином, покращення часу адаптації мережі є важливою проблемою. Саме на цій проблемі ми фокусуємося в даній роботі. Нами запропоновано шаблон Λ (лямбда) за допомогою якого ми обмежуємо, які параметри мережі слід оновлювати під час кроку адаптації. Даний підхід дозволяє не обчислювати градієнти для обраних параметрів та таким чином зменшити кількість необхідних обчислень. Шаблон обирається в межах параметру дозволеного зменшення якості мережі. Серед шаблонів, що відповідають заданому критерію, обирається найшвидший. Однак, як буде показано далі, в деяких випадках також можливе підвищення якості за допомогою правильно обраного шаблону адаптації.

**Результати.** Було реалізовано, навчено та перевірено якість роботи алгоритму MAML із шаблоном адаптації Λ на відкритому наборі даних CIFAR-FS, що робить отримані результати легко відтворюваними.

**Висновки.** Проведені експерименти показують, що із вибором шаблону Λ можливе значне покращення методу MAML в наступних областях: час адаптації було зменшено в 3 рази за мінімальних втрат якості. Цікаво, що для однокрокової адаптації якість значно виросла за умови використання запропонованого шаблону. Перспективи подальших досліджень можуть полягати в розробці більш робастного методу автоматичного вибору шаблонів.

**КЛЮЧОВІ СЛОВА:** пристрілкове навчання, мета-навчання, Model-Agnostic Meta-Learning, MAML, час адаптації, швидкість адаптації, оптимізаційне мета-навчання.

УДК 004.93

## УСКОРЕНИЕ ФАЗЫ АДАПТАЦИИ ОПТИМИЗАЦИОННОГО МЕТА-ОБУЧЕНИЯ

**Хабарлак К. С.** – аспирант кафедры Системного анализа и управления Национального технического университета «Днепровская политехника», Днепр, Украина.

### АННОТАЦИЯ

**Актуальность.** Нейронные сети требуют большого количества размеченных данных для обучения. Алгоритмы мета-обучения предлагают способ обучаться лишь по нескольким примерам. Одним из наиболее выдающихся алгоритмов оптимизационного мета-обучения является MAML. Однако, его процедура адаптации к новым задачам достаточно медленная. Объектом исследования является процесс мета-обучения и фаза адаптации в виде, как она определена в алгоритме MAML.

**Цель.** Цель данной работы – создание подхода, которых позволит: 1) уменьшить время выполнения адаптации алгоритма MAML; 2) улучшить качество алгоритма в ряде случаев. Показать результаты тестирования на открытом наборе данных для мета-обучения CIFAR-FS.

**Метод.** В данной работе предложено улучшение алгоритма мета-обучения MAML. Процедура мета-обучения определяется через так называемые «задачи». В случае классификации изображений каждая задача является попыткой научиться классифицировать изображения новых классов по нескольким обучающим примерам. В алгоритме MAML определено 2 шага в процедуре обучения: 1) адаптация к новой задаче; 2) обновления мета-параметров сети. Вся процедура обучения требует вычисление гессиана, что делает метод вычислительно сложным. После обучения сеть, как правило, будет использоваться для адаптации к новым задач и последующей классификации на них. Таким образом, улучшение времени адаптации сети является важной проблемой. Именно на этой проблеме мы и фокусируемся в данной работе. Нами предложено шаблон Λ (лямбда), с помощью которого мы ограничиваем, какие параметры сети следует обновлять во время шага адаптации. Данный подход позволяет не вычислять градиенты для выбранных параметров и таким образом уменьшить количество необходимых вычислений. Шаблон выбирается в рамках значения параметра разрешенного падения качества сети. Среди шаблонов, которые соответствуют заданному критерию, выбирается наиболее быстрый. Однако, как будет показано дальше, в некоторых случаях также возможно повышение качества с помощью правильно выбранного шаблона адаптации.

**Результаты.** Было реализовано, обучено и проверено качество работы алгоритма MAML с шаблоном адаптации Λ на открытом наборе данных CIFAR-FS, что делает полученные результаты легко воспроизводимыми.

**Выводы.** Проведенные эксперименты показывают, что с выбором шаблона Λ возможно значительное улучшение метода MAML в следующих областях: время адаптации было уменьшено в 3 раза при минимальных потерях в качестве. Интересно и то, что для одношаговой адаптации качество значительно выросло при условии использования выбранного шабло-

© Khabarlak K. S., 2022
DOI 10.15588/1607-3274-2022-1-10

на. Перспективы дальнейших исследований могут заключаться в разработке более робастного метода автоматического выбора шаблонов.

**КЛЮЧЕВЫЕ СЛОВА:** пристрелочное обучение, мета-обучение, Model-Agnostic Meta-Learning, MAML, время адаптации, скорость адаптации, оптимизационное мета-обучение.

## ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Deep Residual Learning for Image Recognition / [K. He, X. Zhang, S. Ren et al.] // 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016. – IEEE Computer Society, 2016. – P. 770–778. DOI: 10.1109/CVPR.2016.90.
2. ImageNet: A large-scale hierarchical image database / [J. Deng, W. Dong, R. Socher et al.] // 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20–25 June 2009, Miami, Florida, USA. – IEEE Computer Society, 2009. – P. 248–255. DOI: 10.1109/CVPR.2009.5206848.
3. Densely Connected Convolutional Networks / [G. Huang, Z. Liu, L. Maaten et al.] // 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017. – IEEE Computer Society, 2017. – P. 2261–2269. DOI: 10.1109/CVPR.2017.243.
4. Zagoruyko S. Wide Residual Networks / S. Zagoruyko, N. Komodakis // Proceedings of the British Machine Vision Conference (BMVC). – BMVA Press, 2016. – P. 87.1–87.12. DOI: 10.5244/C.30.87.
5. Finn C. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks / C. Finn, P. Abbeel, S. Levine // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017: Proceedings of Machine Learning Research. – PMLR, 2017. – Vol. 70. – P. 1126–1135.
6. Meta-Learning with Implicit Gradients / [A. Rajeswaran, C. Finn, S. Kakade et al.] // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada. – 2019. – P. 113–124.
7. Khabarlak K. Fast Facial Landmark Detection and Applications: A Survey [Electronic resource] / K. Khabarlak, L. Koriashkina // arXiv:2101.10808 [cs]. – 2021. – Access mode: https://arxiv.org/abs/2101.10808
8. Meta-learning with differentiable closed-form solvers [Electronic resource] / [L. Bertinetto, J. Henriques, P. Torr et al.] // 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. Access mode: https://openreview.net/forum?id=HyxnZh0ct7.
9. Snell J. Prototypical Networks for Few-shot Learning / J. Snell, K. Swersky, R.S. Zemel // Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA. – 2017. – P. 4077–4087.
10. Ravi S. Optimization as a Model for Few-Shot Learning [Electronic resource] / S. Ravi, H. Larochelle // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. – Access mode: https://openreview.net/forum?id=rJY0-Kcll.
11. Antoniou A. How to train your MAML [Electronic resource] / A. Antoniou, H. Edwards, A. J. Storkey // 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. Access mode: https://openreview.net/forum?id=HJGven05Y7.
12. Weng L. Meta-Learning: Learning to Learn Fast [Electronic resource] / L. Weng. – Access mode: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html.
13. Yin W. Meta-learning for Few-shot Natural Language Processing: A Survey [Electronic resource] / W. Yin // CoRR. – 2020. – Vol. abs/2007.09604. – Access mode: https://arxiv.org/abs/2007.09604.
14. Generalizing from a Few Examples: A Survey on Few-shot Learning / [Y. Wang, Q. Yao, J. Kwok et al.] // ACM Comput. Surv. – 2020. – Vol. 53, № 3. – P. 63:1–63:34. DOI: 10.1145/3386252.
15. Guo Y. One-shot Face Recognition by Promoting Underrepresented Classes [Electronic resource] / Y. Guo, L. Zhang // CoRR. – 2017. – Vol. abs/1707.05574. – Access mode: http://arxiv.org/abs/1707.05574.
16. Koch G. Siamese neural networks for one-shot image recognition / G. Koch, R. Zemel, R. Salakhutdinov // ICML deep learning workshop. – Lille, 2015. – Vol. 2.
17. Matching Networks for One Shot Learning / [O. Vinyals, C. Blundell, T. Lillicrap et al.] // Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5–10, 2016, Barcelona, Spain. – 2016. – P. 3630–3638.
18. Meta-Learning with Memory-Augmented Neural Networks / [A. Santoro, S. Bartunov, M. Botvinick et al.] // Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19–24, 2016 : JMLR Workshop and Conference Proceedings. – JMLR.org, 2016. – Vol. 48. – P. 1842–1850.
19. Lake B.M. Human-level concept learning through probabilistic program induction / B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum // Science. – 2015. – Vol. 350. – № 6266. – P. 1332–1338. – DOI: 10.1126/science.aab3050.
20. Nichol A. On First-Order Meta-Learning Algorithms [Electronic resource] / A. Nichol, J. Achiam, J. Schulman // CoRR. – 2018. – Vol. abs/1803.02999. – Access mode: http://arxiv.org/abs/1803.02999.
21. Meta-SGD: Learning to Learn Quickly for Few Shot Learning [Electronic resource] / [Z. Li, F. Zhou, F. Chen et al.] // CoRR. – 2017. – Vol. abs/1707.09835. – Access mode: http://arxiv.org/abs/1707.09835.
22. Zeiler M.D. Visualizing and Understanding Convolutional Networks / M.D. Zeiler, R. Fergus // Computer Vision – ECCV 2014 – 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I : Lecture Notes in Computer Science. – Springer, 2014. – Vol. 8689. – P. 818–833. – DOI: 10.1007/978-3-319-10590-1_53.
23. Ioffe S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / S. Ioffe, C. Szegedy // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015: JMLR Workshop and Conference Proceedings. – JMLR.org, 2015. – Vol. 37. – P. 448–456.
24. Kingma D.P. Adam: A Method for Stochastic Optimization / D.P. Kingma, J. Ba // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. – 2015.
25. Krizhevsky A. Learning multiple layers of features from tiny images [Electronic resource] / A. Krizhevsky. – University of Toronto, 2009. – Access mode: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf