Contextual Gradient Scaling for Few-Shot Learning

Sanghyuk Lee  Seunghyun Lee  Byung Cheol Song
Department of Electrical and Computer Engineering, Inha University
{sanghyuk.lee625, lsh910703}@gmail.com, bcsong@inha.ac.kr

Abstract

Model-agnostic meta-learning (MAML) is a well-known optimization-based meta-learning algorithm that works well in various computer vision tasks, e.g., few-shot classification. MAML is to learn an initialization so that a model can adapt to a new task in a few steps. However, since the gradient norm of a classifier (head) is much bigger than those of backbone layers, the model focuses on learning the decision boundary of the classifier with similar representations. Furthermore, gradient norms of high-level layers are small than those of the other layers. So, the backbone of MAML usually learns task-generic features, which results in deteriorated adaptation performance in the inner-loop. To resolve or mitigate this problem, we propose contextual gradient scaling (CxGrad), which scales gradient norms of the backbone to facilitate learning task-specific knowledge in the inner-loop. Since the scaling factors are generated from task-conditioned parameters, gradient norms of the backbone can be scaled in a task-wise fashion. Experimental results show that CxGrad effectively encourages the backbone to learn task-specific knowledge in the inner-loop and improves the performance of MAML up to a significant margin in both same- and cross-domain few-shot classification.

1. Introduction

Deep learning has enabled rapid development of various computer vision tasks such as classification [13][20][40], object detection [11][22][29], and segmentation [12][23][51]. However, the success of many deep learning-based computer vision algorithms highly counts on large datasets. Unfortunately, there is no guarantee that we can always collect large-scale data due to privacy issues or the low frequency of occurrence of the target event.

The critical difference between machines and humans in the learning process is whether or not to use prior knowledge. Machines are not good at leveraging prior knowledge (learned in the past) on current learning tasks, while humans can learn new tasks quickly by utilizing prior knowledge.

To bridge this gap, few-shot learning has been developed [45]. Especially, meta-learning draws a lot of attention. Meta-learning learns prior knowledge, also known as meta-knowledge [14], through various tasks. That is, transferring prior knowledge to a new task enables more efficient learning of the task. As a famous meta-learning, MAML [10] learns an initialization of a model as meta-knowledge. Thus, MAML makes the adaptation to a new task complete in only a few steps.

However, MAML does not achieve effective adaptation due to the following two problems related to gradient norms. The first problem is that as meta-training progresses, the gradient norm of the classifier increases while those of the backbone decrease. Note that a gradient norm implies to what extent a certain layer has to be updated. As the gra-
gradient norm of the classifier becomes larger than the backbone, the representation reuse and the decision boundary-oriented update tend to dominate further and further. As a result, since good representations are unavailable and the adaptation highly depends on the decision boundary during learning, a new task cannot be solved properly. In short, this problem makes effective adaptation difficult. The second problem is that gradient norms of high-level layers of the backbone are much smaller than those of the low- and mid-level layers. It is well known that high-level layers are good at learning discriminative features [47]. Thus, this problem indicates that the backbone focuses on extracting generic and low-level features rather than discriminative and high-level features from a given task.

Due to the problems mentioned above, MAML is tempted to learn task-generic representations and its adaptation strongly relies on the decision boundary-oriented update. So it is likely to perform unsatisfactory adaptation in both meta-training and meta-testing. This phenomenon becomes severer when dataset domains in meta-training and meta-testing are different from each other (e.g., cross-domain). Meanwhile, note that our argument is closely related to what is said in [26]. It insists that an appropriate representation change is required for MAML. Actually, BOIL [26] froze the classifier during adaptation to change representations as much as possible. However, BOIL still has a limit to leverage the representation change because it does not explicitly handle gradient norms.

To solve the problems mentioned above, this paper proposes Contextual Gradient Scaling (CxGrad). CxGrad learns the task-embedding vectors by adopting context parameters [48] in adaptation. Specifically, a sub-network takes context parameters as input and generates scaling factors to scale gradient norms of the backbone in a task-wise manner. Since the sub-network consists of differentiable layers and allows end-to-end learning, CxGrad can find the local optimal scaling factors to increase the proportion of the backbone in meta-training. Figure 1 compares how MAML and CxGrad perform adaptation in the inner-loop, respectively. We can observe that CxGrad facilitates backbone learning to get a more optimized representation than MAML. Note that the dispersed samples make clusters after adaptation. Therefore, CxGrad makes the backbone learn task-specific features, which results in the performance improvement of MAML in both same- and cross-domain few-shot classification.

The contribution points of this paper are summarized as follows:

- We propose a meta-learning algorithm that contextually scales gradient norms. The proposed scaling method decreases gradient norms of the classifier and increases those of the backbone for effective adaptation.
- Furthermore, the proposed method learns more task-specific features than MAML by increasing gradient norms of high-level layers of the backbone.
- As a result, the proposed method provides state-of-the-art or very competitive performance in both same- and cross-domain few-shot classification.

2. Related Work

The goal of meta-learning, also known as learning to learn [41], is to utilize meta-knowledge shared between tasks by transferring it to a new task for more efficient adaptation [6, 7, 35, 36, 41]. In general, meta-learning can be classified into three categories: metric-based, model-based, and optimization-based algorithms. Metric-based algorithms prefer adaptation through an embedding function that maps each input into an embedding space [17, 38, 39, 43]. The embedding function learns to make similar classes mapped close to each other and different classes mapped far from each other. Specifically, it embeds the support set and the query set, and then compares their similarities. Model-based algorithms encode the adaptation process into the feed-forward path. They parameterize a task by encoding the training set [29], or use a separate buffer [33]. Optimization-based algorithms literally optimize the adaptation process [1, 10, 28].

The most representative optimization-based method is MAML [10]. Since MAML can be applied to various computer vision fields based on optimization, it has recently received considerable attention, along with few-shot classification. MAML learns an initialization as meta-knowledge that can be adapted to a new task in a small number of steps. Still, MAML suffers from a few drawbacks, such as requiring high computation cost due to bi-level optimization and lower performance than other meta-learning methods.

This study pursues task-dependent meta-learning. Zintgraf [43] proposes context parameters, i.e., a low-dimensional task representation, which embed task information in a vector form. Context parameters are additionally fed into a given model to generate factors for modulating the intermediate feature maps. ModGrad [37] adopts context parameters of [43] to solve the noisy gradient problem in the low-data regime by element-wise gradient modulation. [5] observes that the direction of updating meta-knowledge between tasks causes conflicts and proposes a method called L2F to attenuate the influence of meta-knowledge in a task-dependent manner. In detail, L2F generates scaling factors and then scales the model parameters with them.

Also, this study is closely related to algorithms using auxiliary task-dependent information during backpropagation. For example, Meta-SGD [21] chooses learning rates and update directions as meta-knowledge and achieves
better generalization ability than MAML. However, since each parameter is updated with its own learning rate, the amount of parameters to be learned becomes doubled. MAML++ [2] focuses on the instability issue that MAML suffers in meta-training and then presents a solution using multi-step loss optimization, cosine annealing of learning rate [24], and so on. Also, since MAML++ learns the learning rate for each layer and each step, it can reduce the cost of parameter size, resulting in overcoming disadvantages of Meta-SGD.

Besides, we try to change the representation for each task. Recently, two studies [26, 27] made contradictory claims about the necessity of the inner-loop in MAML. [27] showed that MAML could be sufficiently learned without inner-loop by reusing features. However, [27] didn’t provide any experimental result regarding cross-domain. On the other hand, [26] induced representation change in the inner-loop by freezing the classifier and showed that the inner-loop plays an important role in solving cross-domain few-shot classification.

The proposed method can be differentiated from the methods mentioned above in some respects. First, in order to induce task-specific feature learning, L2F [5] attenuates conflicts among tasks, but the proposed method scales gradient norms of the backbone by using the property of batch normalization (BN) [15]. Second, while CAVIA [48] and ModGrad [37] update context parameters in a single step, the proposed method makes the context parameters depend on all steps by accumulating the information obtained from the meta-learner at each step. The third one is the difference in the representation change perspective. Although BOIL [26] emphasizes the importance of the representation change for the first time, its representation change is very limited only by freezing the classifier. On the contrary, assuming that a major factor interfering with representation change results from gradient norms, the proposed method generates task-wise scaling factors in an end-to-end manner and then scales gradient norms of the backbone. Thus, more effective representation change can be derived.

3. Proposed Method

3.1. Preliminaries

Before explaining the details of the proposed method, we describe overviews of meta-learning and MAML in this section. Suppose there exists a distribution over tasks $p(T)$. The goal of meta-learning is to learn transferable meta-knowledge from $p(T)$. For this, we compose a mini-batch $\{T_i\}_{i=1}^{B}$ with multiple tasks sampled from $p(T)$ and $B$ is the batch size. Here, $i$ is the index of a task. Each $T_i$ consists of a task-specific loss function $L_{T_i}$, a support set $S_i$, and a query set $Q_i$:

$$S_i = \{(x_j, y_j)\}_{j=1}^{S_i}, \quad Q_i = \{(x_j, y_j)\}_{j=1}^{Q_i}$$

where $\cdot$ is the cardinality of a set, $x_j \in \mathcal{X}$ is a sample of the input space $\mathcal{X}$, and $y_j \in \mathcal{Y}$ is the corresponding label. $\mathcal{Y}$ stands for the label space. Meta-learning performs adaptation to $T_i$ through $S_i$, and learns meta-knowledge through evaluation using $Q_i$. In few-shot classification, $S_i$ and $Q_i$ share the same $N$ classes, but the samples of the classes are different from each other. If a class of $S_i$ is composed of $K$ samples, the corresponding $T_i$ is called $N$-way $K$-shot task and $|S_i|$ becomes $NK$. MAML defines an initialization $\theta$ as meta-knowledge and learns the initialization. Based on $\theta$, MAML can complete adaptation to a new task $T_i$ in only a small number of steps. Given $\{T_i\}_{i=1}^{B}$, the following processes are performed for each $T_i$. First, a model is optimized on $S_i$ in the inner-loop starting from $\theta$ to get a task-specific parameter $\theta_i$. Then, the adaptation proceeds according to the following equation:

$$\theta_i = \theta - \alpha \nabla_{\theta} L_{T_i}(\theta; S_i)$$

where $\alpha$ is the learning rate in the inner-loop. Next, based on $\theta_i$, $L_{T_i}(\theta_i; Q_i)$, i.e., loss for $Q_i$ is calculated. Finally, if all the processes are completed for every task in the mini-batch, $\theta$ is updated by Eq. (2).

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \sum_{i=1}^{B} L_{T_i}(\theta_i; Q_i)$$

where $\eta$ is the learning rate in the outer-loop. In other words, how well the meta-knowledge is learned is evaluated and $\theta$ is updated based on the evaluation. Therefore, the entire procedure results in learning well-generalized $\theta$.

3.2. Contextual Gradient Scaling

Gradient norm is one of the important indicators of how well a model is learning. So, its analysis is worthwhile. Let’s take a look at the gradient norm of each layer in the inner-loop. In Figure 2a we can find that not only the gradient norm of the classifier is much larger than that of each backbone layer, but also gradient norms of high-level layers of the backbone are smaller than those of the other layers. Due to this phenomenon, MAML tends to concentrate on decision boundary-oriented update and low-/mid-level features. Thus, MAML does not achieve effective adaptation in the inner-loop. There were a few naive approaches to solve this problem, e.g., freezing the classifier [26] or clipping the gradients of the classifier. However, such approaches do not affect gradient norms of the backbone explicitly.

Thus, this paper proposes an explicit solution, CxGrad, that contextually scales gradient norms. CxGrad utilizes a
Figure 2: Gradient norms of MAML and CxGrad during adaptation in meta-training. Gradient norms are averaged over inner-loop steps and tasks of the batch. In (a) MAML, the gradient norm of the classifier is bigger than those of backbone layers. Furthermore, gradient norms of low- and mid-levels are bigger than those of high-level layers. Considering the fact that high-level layers learn more discriminative features, MAML reuses similar representation from meta-knowledge and heavily depends on updating a decision boundary. However, in (b) CxGrad, gradient norms of high-level layers are much bigger than the others. Also, the norm gets smaller near the front side of the backbone. As a result, CxGrad focuses on changing representation and the backbone learns to extract task-specific features.

Before explaining the scaling process in detail, we need to briefly review the property of BN. While BN is scale-invariant in forward propagation, it is not in backward propagation. Let’s take a look at this property mathematically. Following the notations of [9], let the parameter vector of a convolution layer be $v$, the scaled version of $v$ be $\hat{v} = av$, a vector of the input feature map be $x$, and the covariance matrix be $\Sigma$. Here, $a$ is a positive real number. Also, the output feature map is $z = v^\top x$, and the scaled feature map is $\hat{z} = \hat{v}^\top x$. Then, the outputs of BN layers before and after scaling the parameter vector by a scaling factor $a$ during forward propagation have the following relationship.

$$
\text{BN}(\hat{v}^\top x) = \frac{\hat{v}^\top (x - E[x])}{\sqrt{\hat{v}^\top \Sigma \hat{v}}} = \frac{av^\top (x - E[x])}{\sqrt{a^2 v^\top \Sigma v}} = \frac{v^\top (x - E[x])}{\sqrt{v^\top \Sigma v}} = \text{BN}(v^\top x)
$$

However, in case of backward propagation, this scale-invariance does not hold. Instead, the gradient norm is scaled in proportion to the reciprocal of $a$.

$$
\frac{\partial \text{BN}(\hat{z})}{\partial \hat{v}} = \frac{1}{a} \frac{\partial \text{BN}(z)}{\partial v}
$$

Note that since the BN property also holds even in Layer normalization [3] and Group normalization [46], the scaling process can be applied to any architectures containing the other normalization layers as well as BN layers.

Based on the BN property, gradient norms of the backbone are properly scaled using a scaling factor set. Let the scaling factor set of the backbone be $G = \{\gamma_i(\ell)\}_{\ell=1}^L$. Here, $\ell$ denotes an index of the backbone layer and $L$ denotes the number of backbone layers. Also, let the parameters of the backbone and classifier constituting a model be $w_i, \psi_i$, respectively. Then, CxGrad generates $G$ from $g_\phi$ which is a sub-network parameterized with $\phi$. If a scaling factor $\gamma_i(\ell)$ is always a positive number less than 1, the norm of the $\ell$-th backbone layer parameter $\|w_{i(\ell)}\|$ can increase without any upper bound because each gradient norm is scaled by the reciprocal of the scaling factor according to Eq. (5). In order to avoid this, we $\ell_2$-normalize $w_{i(\ell)}$ to make $\|w_{i(\ell)}\|=1$ before scaling. Finally, the entire scaling procedure can be expressed as follows:

$$
\begin{align*}
\gamma_i(\ell) &\leftarrow \{\gamma_i(\ell) w_{i(\ell)}/\|w_{i(\ell)}\|\}_{\ell=1}^L \\
\end{align*}
$$

Owing to this inherent nature of BN, during forward propagation, the model outputs do not change even if the scaling process is applied. In other words, the loss doesn’t change. However, during backward propagation, gradient norms are
the context parameters generate like the other layers in the increasing direction. Besides, since softplus can be differentiable and both $g_\phi$ and $\nu_i$ are included in the computation graph. Finally, $G$ is generated with $\nu_i$ as an input to $g_\phi$.

\[
\nu_i \leftarrow \nu_i - \beta \nabla_{\nu_i} \mathcal{L}(\theta_i; S_i)
\]

We omit the task notation $T_i$ from the task-specific loss function $\mathcal{L}_{T_i}$ because we only focus on classification tasks in this paper. Learning via Eq. (7) is plausible because $g_\phi$ consists of differentiable layers and both $g_\phi$ and $\nu_i$ are included in the computation graph. Finally, $G$ is generated with $\nu_i$ as an input to $g_\phi$.

\[
G = \{\gamma_i \ell \}_{\ell=1}^T = g_\phi(\nu_i)
\]

Some may ask how about scaling gradient norms when updating $\nu_i$. However, to update $\nu_i$, forward and backward propagation must be performed twice each. The first forward and backward propagation indicate the process of including $\nu_i$ in the computation graph (Eq. (6)) and the process of updating $\nu_i$ (Eq. (7)). Finally, the gradients calculated by the second backward propagation are scaled, and then $\theta_i$ is updated. On the other hand, the above process can be simplified by the BN property. Suppose we update $\nu_i$ before the first inner-loop step. Then, gradient norms can be scaled through only one pair of forward and backward propagation during the subsequent adaptation process. Furthermore, $\nabla_{\nu_i} \mathcal{L}_{T_i}(\theta_i; S_i)$ can be computed simultaneously with $\nabla_{\theta_i} \mathcal{L}_{T_i}(\theta_i; S_i)$ without any extra process. For details, please refer to Algorithm 1 and Algorithm 2.

4. Experiments

Various experiments show how efficient the proposed method is in few-shot classification. First, the proposed method is compared with several state-of-the-art (SOTA) algorithms on benchmark datasets, and then its quantitative evaluation in a cross-domain environment is given. In addition, we compare feature similarity at each backbone layer before and after adaptation, and visualize the effect of the proposed method using t-SNE [42]. Finally, various ablation studies are provided. In all the experiments below, four convolutional blocks, proposed by [43], are used as the backbone architecture. For more implementation details, please refer to the supplementary material. The codes are available at [https://github.com/shlee625/CxGrad](https://github.com/shlee625/CxGrad).

4.1. Few-Shot Classification

Two famous benchmark datasets, i.e., miniImageNet [43] and tieredImageNet [30], are used for few-shot classification. MiniImageNet composed of a total of 60,000 images was obtained by extracting 100 classes from
Table 1: Test accuracy on 5-way miniImageNet and tieredImageNet classification. CxGrad outperforms the others in 5-shot on both datasets and in 1-shot on tieredImageNet. CxGrad shows comparable result even in 1-shot on miniImageNet.

| Method        | miniImageNet 1-shot | miniImageNet 5-shot | tieredImageNet 1-shot | tieredImageNet 5-shot |
|---------------|---------------------|---------------------|-----------------------|-----------------------|
| MatchingNet   | 43.44 ± 0.77%       | 55.31 ± 0.73%       | –                     | –                     |
| RelationNet   | 50.44 ± 0.82%       | 65.32 ± 0.70%       | –                     | –                     |
| ProtoNet      | 49.42 ± 0.78%       | 68.20 ± 0.66%       | –                     | –                     |
| MAML          | 48.70 ± 1.84%       | 63.11 ± 0.92%       | 49.06 ± 0.50%         | 67.48 ± 0.47%         |
| CAVIA (512)   | 51.82 ± 0.65%       | 65.85 ± 0.55%       | –                     | –                     |
| BOIL          | 49.61 ± 0.16%       | 66.45 ± 0.37%       | 48.58 ± 0.27%         | 69.37 ± 0.12%         |
| MAML++        | 52.15 ± 0.26%       | 68.32 ± 0.44%       | –                     | –                     |
| ALFA          | 50.58 ± 0.51%       | 69.12 ± 0.47%       | 53.16 ± 0.49%         | 70.54 ± 0.46%         |
| ModGrad       | 53.20 ± 0.86%       | 69.17 ± 0.69%       | –                     | –                     |
| L2F           | 52.10 ± 0.50%       | 69.38 ± 0.46%       | 54.40 ± 0.50%         | 73.34 ± 0.44%         |
| CxGrad (Ours) | 51.80 ± 0.46%       | 69.82 ± 0.42%       | 55.55 ± 0.46%         | 73.55 ± 0.41%         |

Table 2: Test accuracy on 5-way 5-shot cross-domain few-shot classification on two meta-test datasets.

| Method         | miniImageNet → CUB | miniImageNet → CIFAR-FS |
|----------------|--------------------|-------------------------|
| MAML           | 52.70 ± 0.32%      | 55.82 ± 0.50%           |
| ALFA           | 58.35 ± 0.25%      | 59.76 ± 0.49%           |
| BOIL           | 60.84 ± 0.45%      | 63.28 ± 0.45%           |
| L2F            | 60.89 ± 0.22%      | 63.73 ± 0.48%           |
| CxGrad (Ours)  | 63.92 ± 0.44%      | 64.85 ± 0.44%           |

ImageNet [32]. Each class contains 600 images of 84×84. Here, 100 classes are again divided into three folds, i.e., 64, 16, 20 classes which are used for meta-training, meta-validation, and meta-testing, respectively. The classes between folds do not overlap each other. TieredImageNet is a larger dataset than miniImageNet. Similar to miniImageNet, tieredImageNet is created by extracting 608 classes from ImageNet, and consists of a total of 779,165 images. It has the same spatial resolution as miniImageNet, but its classes are grouped into 34 upper categories unlike miniImageNet. Then, 20, 6, and 8 categories are used for meta-training, meta-validation, and meta-testing, respectively, and these categories do not overlap each other, either. Therefore, the datasets are suitable for evaluating the generalization ability of each algorithm.

Table 1 compares the proposed method with several meta-learning algorithms in 5-way 1-shot and 5-shot classification on miniImageNet and tieredImageNet. We can find that the proposed method noticeably outperforms MAML by 3.1% to 6.49% in all configurations. In particular, in the case of 5-shot on miniImageNet, it is noteworthy that the proposed method shows 0.53% better than the SOTA algorithm. On the other hand, in the case of 1-shot on miniImageNet, the proposed method shows slightly lower than the SOTA algorithm. We think that since 1-shot has fewer samples per class than 5-shot, context parameters for embedding the tasks do not obtain sufficient information during meta-training. However, this is not the case with 1-shot on tieredImageNet. This is because despite the same 1-shot, sufficient meta-knowledge is learned thanks to the larger amount of iterations and datasets.

4.2. Cross-Domain Few-Shot Classification

To further analyze the effectiveness of our adaptation strategy, we examined few-shot classification in cross-domain environments. Unlike Section 4.1 two datasets not used in meta-training were employed here for the tasks in meta-testing. Thus, it is necessary to learn a more optimized representation in the inner-loop for a task on the new dataset. In other words, the cross-domain experiment can better examine the efficacy of representation change. For this experiment, 5-way 5-shot was employed in the same architecture as Section 4.1 After meta-knowledge is learned on miniImageNet, a meta-test was conducted by transferring the meta knowledge to the tasks on a different domain dataset.

The datasets for the meta-test are CUB-200-2011 [44] and CIFAR-FS [8]. CUB has 200 fine-grained classes and consists of a total of 11,788 images. CUB is further divided into 100 meta-train classes, 50 meta-validation classes, and 50 meta-test classes. CIFAR-FS is a dataset based on CIFAR100 [19]. It has 100 classes, and each class consists of 600 images, for a total of 60,000 images. CIFAR-FS is again divided into 64 meta-train classes, 16 meta-validation classes, and 20 meta-test classes. Each image is resized to 84×84.

Table 2 compares CxGrad with several optimization-based algorithms. Note that the result of BOIL [26] is re-
produced and the others are from [4] as they are. As argued in [26], if meta-training and meta-testing employ datasets in different domains respectively, representation change becomes further important as domain-specific knowledge is not meaningful any more. This is because domain-specific meta-knowledge does not cover the task distributions of the other domains. Here, domain-specific part in meta-knowledge is learned by samples from $p(T)$ of source domain, i.e., the domain of the meta-training dataset. What can fill this knowledge gap between the domains is the representation change. CxGrad can alleviate the knowledge gap between domains as much as possible because it focuses on learning task-specific features from the target domain tasks. Therefore, CxGrad overwhelms the previous algorithms on CUB and CIFAR-FS. In particular, MAML’s performance are improved with a large margin of at least 9%.

4.3. Representation Change

In ANIL [27] and BOIL [26], the layer-wise feature similarity before and after adaptation was measured and compared in terms of CKA [18]. CKA is one of the metrics to evaluate feature similarity. The larger the CKA value, the greater the similarity. By monitoring the CKA values, we can judge whether to reuse the existing representation or to use the representation optimized for a given task. We analyze MAML, BOIL, and the proposed method in the 5-way 5-shot on miniImageNet. In Figure 3, we plot the average and 95% confidence interval of the CKA value among multiple tasks in meta-test phase. Note that we reproduced other algorithms and measured the CKA values.

Looking at Figure 3 we can find that all three algorithms have CKA values close to 1 at the low- and mid-level layers. This indicates that they all learn task-generic low-level features as claimed in [47]. On the other hand, at the high-level layers, MAML still has CKA values near 1, but BOIL and CxGrad have low CKA values. MAML remains the representation at the high-level layers, but BOIL and CxGrad utilize a more discriminative representation than MAML. Furthermore, CxGrad has a lower CKA value than BOIL. That is, CxGrad learns more task-specific features than BOIL as well as MAML by scaling gradient norms in the backbone.

4.4. Visualization of Embeddings

Figure 4 visualizes how CxGrad embeds the tasks via t-SNE [42], i.e., a nonlinear dimensionality reduction tool that maps high-dimensional data to low-dimensional space. First, we can observe that MAML does not perform effective representation change in both miniImageNet and CUB. Only a few classes that are properly clustered before adaptation make clusters after adaptation, while the other classes don’t. On the other hand, CxGrad shows that samples dispersed before adaptation form a cluster after adaptation. It means that CxGrad indeed facilitates effective representation change in the inner-loop. Therefore, CxGrad can realize effective adaptation in the cross-domain datasets as well as the same domain datasets.

4.5. Ablation Study

Update scheme for context parameters A question may arise as to whether just updating context parameters for each step, i.e., $\nu_t$ is sufficient. We argue that it is useful for better performance to update context parameters through whole steps of the inner-loop. To prove this, the following ablation study compares two update schemes in 5-way 5-shot on miniImageNet: task-wise and step-wise. The task-wise method means updating the context parameters over all steps of the inner-loop for $T_i$. On the other hand, the step-wise method means updating the context parameters for each separate step. In this case, we have to reset $\nu_t$ before moving on to the next step. Table 3 shows that task-wise update has better performance than step-wise update. This means that it is more advantageous to accumulate all knowledge of the parameters updated at each step.

Effect of the number of steps Now let’s examine the effect of the number of inner-loop steps in adaptation. This experiment is also performed in 5-way 5-shot on miniImageNet. Table 4 shows that although five steps shows the highest performance, the difference from a single step showing the lowest performance amounts to only 1%.

| Update scheme for $\nu_t$ | Accuracy       |
|--------------------------|----------------|
| step-wise                | 68.83 ± 0.43%  |
| task-wise                | 69.82 ± 0.42%  |

Table 3: Effect of context parameter update scheme. The accuracy results from 5-way 5-shot on miniImageNet.
Therefore, the proposed method is relatively robust to the number of inner-loop steps.

**Inner-loop learning rate** The proposed method has another learning rate ($\beta$) related to context parameters. To examine the performance change according to the learning rate, we compare the models trained with three different learning rates. For convenience, the difference between the learning rates was fixed 10 times. This experiment was performed in 5-way 1-shot and 5-shot on miniImageNet. In Table 5, both cases show better performance when $\beta$ is 1.0. However, note that even if the learning rates differ by 100 times, the difference in performance does not exceed 1%. Therefore, we can claim that the proposed method is meaningfully robust to hyperparameters.

### 5. Conclusion

This paper argues that the ineffective adaptation of MAML results from the two reasons: 1) the gradients are concentrated on the classifier rather than the backbone and 2) the gradients are not concentrated on the high-level layer inside of the backbone. As a result, the performance of MAML is hindered in meta-testing. In particular, the performance degradation is more pronounced in a cross-domain environment. In order to solve this problem, we scale the gradient norms of the backbone by using the property of BN. Here, the scaling factors are generated from the context parameters which embed the task information. As a result, the gradients are scaled in a task-wise manner. Experimental results show that the proposed method accomplishes more effective adaptation. In particular, the knowledge gap caused by domain change in a cross-domain environment is solved by changing the representation through gradient scaling.

### Acknowledgments

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2021-0-02052) supervised by the IITP(Institute for Information & Communications Technology Promotion) and IITP grant funded by MSIT [2020-0-01389, Artificial Intelligence Convergence Research Center(Inha University)].
[39] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In CVPR, 2018.

[40] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICML, 2019.

[41] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In Learning to learn, pages 3–17. Springer, 1998.

[42] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[43] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In NIPS, 2016.

[44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[45] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys (CSUR), 53(3):1–34, 2020.

[46] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018.

[47] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.

[48] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In ICML, 2019.
of the intermediate features in g

The dimension of the context parameters and the dimension of a ReLU activation function, and a 2 × 2 max-pooling layer. The dimension of the context parameters and the dimension of the intermediate features in gφ are both 100. The dimension of the outputs of gφ is same as the number of convolutional layers in the backbone. In Q1, each class contains 15 samples. None of the data-augmentation methods are used in training. The batch size B is 4 for 1-shot and 2 for 5-shot. We optimize the model in the inner-loop for 5 steps and set the learning rates α, β, and η to be 0.01, 1.0, 0.001, respectively. Adam [16] is used as the meta-optimizer in outer-loop. Our model is trained for 50,000 iterations on miniImageNet and 125,000 iterations on tieredImageNet. An ensemble of models, whose ranks are top 5 in terms of meta-validation accuracy, is evaluated on 600 tasks from meta-test set. We run 3 independent runs with 3 different seeds and report the average results.

A. Experiment Settings

Our implementation and experiment settings are based on those of ALFA [4]. In all the experiments, the network architecture is 4 convolutional blocks [43] followed by a fully connected layer. Each block consists of a convolutional layer with 3 × 3 kernel, a batch normalization layer, a ReLU activation function, and a 2 × 2 max-pooling layer. The dimension of the context parameters and the dimension of the intermediate features in gφ are both 100. The dimension of the outputs of gφ is same as the number of convolutional layers in the backbone. In Q1, each class contains 15 samples. None of the data-augmentation methods are used in training. The batch size B is 4 for 1-shot and 2 for 5-shot. We optimize the model in the inner-loop for 5 steps and set the learning rates α, β, and η to be 0.01, 1.0, 0.001, respectively. Adam [16] is used as the meta-optimizer in outer-loop. Our model is trained for 50,000 iterations on miniImageNet and 125,000 iterations on tieredImageNet. An ensemble of models, whose ranks are top 5 in terms of meta-validation accuracy, is evaluated on 600 tasks from meta-test set. We run 3 independent runs with 3 different seeds and report the average results.

B. Experimental Results on Bigger Backbone

In this section, we provide additional experimental results on a deeper backbone, especially ResNet-12 [13]. ResNet-12 consists of 4 residual blocks. Each residual block is composed of three convolutional blocks, each of which consists of a convolutional layer, a BN layer, and a ReLU activation function. A pointwise convolutional block is positioned at the skip connection for matching the number of channels between the residual inputs and the outputs. A 2 × 2 max-pooling layer is at the end of each residual block. The number of channels begins with 64 and gets doubled by each residual block. Finally, we aggregate the spatial dimension of the final representation by a global average pooling layer and pass it to the classifier. For ResNet-12, we apply the scaling process for all the residual blocks. More specifically, in each residual block, we only scale the weights for the three convolutional blocks, not for the pointwise con-

| Method         | miniImageNet |         |         |
|----------------|--------------|---------|---------|
|               | 1-shot       | 5-shot  |         |
| MAML [10]     | 58.37 ± 0.49%| 69.76 ± 0.46%|         |
| BOIL [26]     | -            | 71.30 ± 0.28%|         |
| L2F [5]       | 59.71 ± 0.49%| 77.04 ± 0.42%|         |
| ALFA [4]      | 59.74 ± 0.49%| 77.96 ± 0.41%|         |
| CxGrad (Ours) | 60.19 ± 0.45%| 75.17 ± 0.40%|         |

Table 6: Test accuracy on 5-way miniImageNet classification.

| Method | tieredImageNet |         |         |
|--------|----------------|---------|---------|
|        | 1-shot         | 5-shot  |         |
| MAML [10] | 58.58 ± 0.49%| 71.24 ± 0.43%|         |
| L2F [5]   | 64.04 ± 0.48%| 81.13 ± 0.39%|         |
| ALFA [4]  | 64.62 ± 0.49%| 82.48 ± 0.38%|         |
| CxGrad (Ours) | 65.47 ± 0.44%| 82.52 ± 0.35%|         |

Table 7: Test accuracy on 5-way tieredImageNet classification.

volutional block. Table 6 and Table 7 provide 5-way few-shot classification performance using ResNet-12 on miniImageNet and tieredImageNet, respectively.

C. Optimization Landscape

Ioffe [15] argued that batch normalization helps the training by reducing internal covariate shift (ICS). However, Santurkar [34] found that the true reason is that batch normalization smooths the optimization landscape in training. In other words, batch normalization improves the Lipschitzness of both the loss and the gradients of a model. To prove this argument, he measured the variation in loss, gradient predictiveness, and “effective” β smoothness in the vicinity of a certain point on the optimization landscape. For more details, please refer to Section 3 in [34]. Based on this study, Baik [5] analyzed the optimization landscape of MAML and his method L2F. Likewise, in this section, we also analyze how our method affects the optimization landscape in the inner-loop following [5, 34]. In Figure 5, we plot these three measurements and explain the meaning of each one.

In order to analyze the optimization landscape in the inner-loop, we have to observe the loss and the gradients in the vicinity of the model parameters adapted to Ti. To accomplish this, we perform adaptation with a new learning rate set in [0, 4] × α. We set α = 0.01 as before. Let the new learning rate set Λ = {x| x = 0.5αi, i ∈ [8]}, j-th learning rate of Λ be αj, and θi, s be the parameters of the adapted model to Ti at s-th step in the inner-loop. Then, we can compute several model parameters around θi, s as below:

\[ \theta_{i,s}^α = \theta_{i,s} - \alpha_j \nabla_{\theta_{i,s}} \mathcal{L}(\theta_{i,s}, S_i) \]  

(9)

Next, let B denote the batch size and S denote the number of inner-loop steps. We plot the variation in the loss and the gradient predictiveness computed by Eq. (10) and Eq. (11) in Figure 5a and Figure 5b, respectively.

\[ \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \mathcal{L}(\theta_{i,s}^α, S_i) \]  

(10)
Figure 5: Optimization landscape during adaptation. Like other experiments, we plot these results from 5-way 5-shot mini-ImageNet few-shot classification. In order to analyze the landscape, we update the model parameter from a certain inner-loop step with various range of learning rates. We refer to these updated parameters as points. With these, we can analyze the local landscape from the certain step. (a) We measure the variation in loss calculated at the points. (b) We measure the variation in $\ell_2$ distance between the gradients at the certain step and at each point. (c) “effective” $\beta$ smoothness refers to the maximum $\ell_2$ difference of (b) over distance as we move to the corresponding point from the certain step.

$$
\frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \| h(\theta_{i,s}^{\alpha_j}) - h(\theta_i) \| \quad (11)
$$

where $h(\theta_i) = \nabla_{\theta_i} \mathcal{L}(\theta_i; S_i)$. In Figure 5c we calculate the values by Eq. (12) and plot them. Here, $\bar{\alpha}_j^* = \arg\max_{\alpha_j \in A} \| h(\theta_{i,s}^{\bar{\alpha}_j}) - h(\theta_i) \|$.  

$$
\frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \frac{\| h(\theta_{i,s}^{\alpha_j}) - h(\theta_i) \|}{\| \bar{\alpha}_j^* h(\theta_i) \|} \quad (12)
$$

Looking at Figure 5a, we can observe the loss landscape in the inner-loop as the training proceeds. In the case of MAML, the variation in loss becomes larger from approximately 5,000 iterations. It means that the loss landscape gets sharper in the vicinity of the points on it. On the contrary, in the case of CxGrad, the variation in loss is drastically reduced, implying that CxGrad efficiently and effectively smooths the loss landscape. In Figure 5b gradient predictiveness means how far the gradients around a certain point are from the gradient at the point. The farther the distance, the lower the stability. At the beginning of training, CxGrad is more unstable than MAML in terms of gradients because the sub-network $g_{\phi}$ doesn’t learn sufficient knowledge from tasks to scale the gradients of the backbone in a task-wise manner. Nevertheless, CxGrad retains enough stability in no time. Lastly, in Figure 5c CxGrad shows better Lipschitzness than MAML. It means that CxGrad also smooths the gradients besides the loss. As a result, CxGrad improves both the convergence speed and the performance of the model.