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, 19, 38], object detection [11, 21, 28], and segmentation [12, 22, 30]. 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.

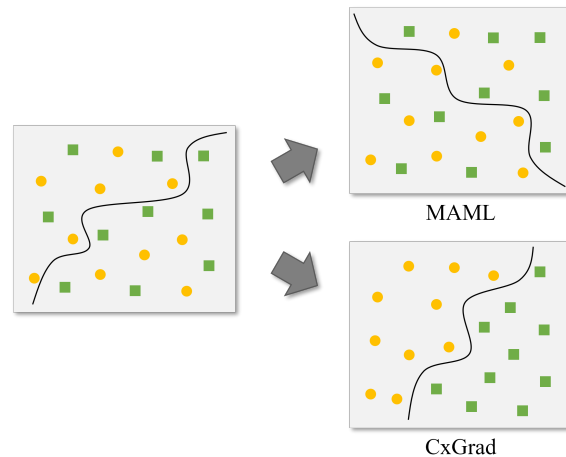To bridge this gap, few-shot learning has been devel-

Figure 1: An overview of the adaptation scheme of MAML and CxGrad. Shapes represent samples belonging to different classes, and lines represent decision boundaries of the classifier. MAML focuses on updating the decision boundary to adapt to a given task. However, CxGrad mainly updates the feature extractor rather than the decision boundary. As a result, although training tasks and target tasks are far different, we can get optimized representation for the target tasks. Furthermore, CxGrad updates the decision boundary based on the changed representation.

oped [43]. 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-

dient 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 [45]. 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 adaption 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 [25]. [25] insists that an appropriate representation change is required for MAML. Actually, BOIL [25] 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 [46] 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 [39], is to utilize meta-knowledge shared between tasks by transferring it to a new task for more efficient adaptation [6, 7, 33, 34, 39]. 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 [16, 36, 37, 41]. 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 [24], or use a separate buffer [32]. Optimization-based algorithms literally optimize the adaptation process [1, 10, 27].

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 [46] 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 [35] adopts context parameters of [46] 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 [20] 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 [23], 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 the disadvantage of Meta-SGD.

Besides, we try to change the representation for each task. Recently, two studies [25, 26] made contradictory claims about the necessity of the inner-loop in MAML. [26] showed that MAML could be sufficiently learned without inner-loop by reusing features. However, [26] didn't provide any experimental result regarding cross-domain. On the other hand, [25] 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 [46] and ModGrad [35] 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 [25] 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 $p(\mathcal{T})$. The goal of meta-learning is to learn transferable meta-knowledge from $p(\mathcal{T})$. For this, we compose a mini-batch $\{\mathcal{T}_i\}_{i=1}^B$ with multiple tasks sampled from $p(\mathcal{T})$ and B is the batch size. Here, i is the index of a task. Each $\mathcal{T}_i$ consists of a task-specific loss function $\mathcal{L}_{\mathcal{T}_i}$, a support set S_i, and a query set Q_i:

$$S_i = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{|S_i|}, \quad Q_i = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{|Q_i|} \quad (1)$$

where $|\cdot|$ is the cardinality of a set, $\boldsymbol{x}_j \in \mathcal{X}$ 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 $\mathcal{T}_i$ 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 $\mathcal{T}_i$ is called N-way K-shot task and $|S_i|$ becomes NK.

MAML defines an initialization $\boldsymbol{\theta}$ as meta-knowledge and learns the initialization. Based on $\boldsymbol{\theta}$, MAML can complete adaptation to a new task $\mathcal{T}_i$ in only a small number of steps. Given $\{\mathcal{T}_i\}_{i=1}^B$, the following processes are performed for each $\mathcal{T}_i$. First, a model is optimized on S_i in the inner-loop starting from $\boldsymbol{\theta}$ to get a task-specific parameter $\boldsymbol{\theta}_i$. Then, the adaptation proceeds according to the following equation:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}; S_i) \quad (2)$$

where α is the learning rate in the inner-loop. Next, based on $\boldsymbol{\theta}_i$, $\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i; Q_i)$, i.e., loss for Q_i is calculated. Finally, if all the processes are completed for every task in the mini-batch, $\boldsymbol{\theta}$ is updated by Eq. (3).

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \sum_{i=1}^B \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i; Q_i) \quad (3)$$

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

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 [25] 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
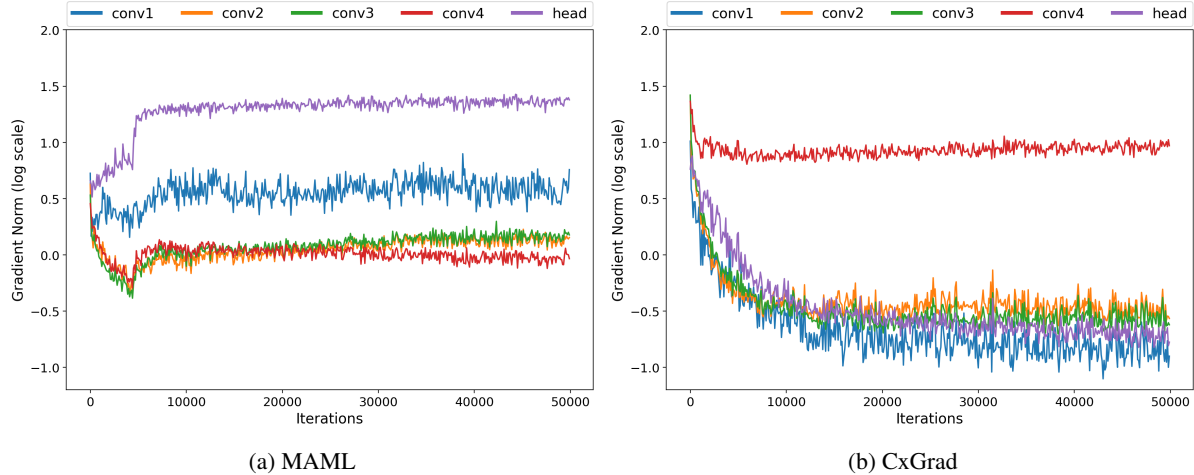
(a) MAML

(b) CxGrad

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.

sub-network g_ϕ and context parameters $\boldsymbol{\nu}_i$. g_ϕ plays an important role in generating scaling factors and $\boldsymbol{\nu}_i$ can be interpreted as embeddings of tasks. First, g_ϕ produces task-specific scaling factors using $\boldsymbol{\nu}_i$. Then, in order to simplify the scaling procedure in the inner-loop, we use the BN property.

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 $\boldsymbol{v}$, the scaled version of $\boldsymbol{v}$ be $\hat{\boldsymbol{v}} = a\boldsymbol{v}$, a vector of the input feature map be $\boldsymbol{x}$, and the covariance matrix be $\boldsymbol{\Sigma}$. Here, a is a positive real number. Also, the output feature map is $\boldsymbol{z} = \boldsymbol{v}^\intercal \boldsymbol{x}$, and the scaled feature map is $\hat{\boldsymbol{z}} = \hat{\boldsymbol{v}}^\intercal \boldsymbol{x}$. 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.

$$\mathrm{BN}(\hat{\boldsymbol{v}}^\intercal \boldsymbol{x}) = \frac{\hat{\boldsymbol{v}}^\intercal(\boldsymbol{x} - \mathrm{E}[\boldsymbol{x}])}{\sqrt{\hat{\boldsymbol{v}}^\intercal \boldsymbol{\Sigma} \hat{\boldsymbol{v}}}} = \frac{a\boldsymbol{v}^\intercal(\boldsymbol{x} - \mathrm{E}[\boldsymbol{x}])}{\sqrt{a^2 \boldsymbol{v}^\intercal \boldsymbol{\Sigma} \boldsymbol{v}}}$$
$$= \frac{\boldsymbol{v}^\intercal(\boldsymbol{x} - \mathrm{E}[\boldsymbol{x}])}{\sqrt{\boldsymbol{v}^\intercal \boldsymbol{\Sigma} \boldsymbol{v}}} = \mathrm{BN}(\boldsymbol{v}^\intercal \boldsymbol{x}) \tag{4}$$

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.

$$\frac{\partial \mathrm{BN}(\hat{\boldsymbol{z}})}{\partial \hat{\boldsymbol{v}}} = \frac{1}{a} \frac{\partial \mathrm{BN}(\boldsymbol{z})}{\partial \boldsymbol{v}} \tag{5}$$

Note that since the BN property also holds even in Layer normalization [3] and Group normalization [44], 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 $\mathcal{G} = \{\gamma_i^{(\ell)}\}_{\ell=1}^{L}$. Here, ℓ 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 $\boldsymbol{w}_i, \boldsymbol{\psi}_i$, respectively. Then, CxGrad generates $\mathcal{G}$ from g_ϕ which is a sub-network parameterized with ϕ. If a scaling factor $\gamma_i^{(\ell)}$ is always a positive number less than 1, the norm of the ℓ-th backbone layer parameter $\|\boldsymbol{w}_i^{(\ell)}\|$ 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 ℓ_2-normalize $\boldsymbol{w}_i^{(\ell)}$ to make $\|\boldsymbol{w}_i^{(\ell)}\| = 1$ before scaling. Finally, the entire scaling procedure can be expressed as follows:

$$\boldsymbol{w}_i \leftarrow \{\gamma_i^{(\ell)} \boldsymbol{w}_i^{(\ell)}/\|\boldsymbol{w}_i^{(\ell)}\|\}_{\ell=1}^{L} \tag{6}$$

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

Algorithm 1 Contextual Gradient Scaling (CxGrad)

Require: α, β, η: Learning rates
Require: $p(\mathcal{T})$: Distribution over tasks

1: Randomly initialize $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$
2: **while** not done **do**
3:     Sample a mini-batch of tasks $\{\mathcal{T}_i\}_{i=1}^{B}$ from $p(\mathcal{T})$
4:     **for** $i \in [B]$ **do**
5:         $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}$
6:         Reset context parameters: $\boldsymbol{\nu}_i = \mathbf{0}$
7:         Sample support and query sets (S_i, Q_i) from $\mathcal{T}_i$
8:         $\boldsymbol{\theta}_i' = \texttt{IGS}(\boldsymbol{\theta}_i, \boldsymbol{\nu}_i)$
9:         Compute $\nabla_{\boldsymbol{\nu}_i} \mathcal{L}(\boldsymbol{\theta}_i'; S_i)$ using support sets.
10:       Update context parameters:
        $\boldsymbol{\nu}_i \leftarrow \boldsymbol{\nu}_i - \beta \nabla_{\boldsymbol{\nu}_i} \mathcal{L}(\boldsymbol{\theta}_i'; S_i)$
11:       **for** the number of inner-loop steps **do**
12:           $\boldsymbol{\theta}_i \leftarrow \texttt{IGS}(\boldsymbol{\theta}_i, \boldsymbol{\nu}_i)$
13:           Compute $\nabla_{(\boldsymbol{\theta}_i, \boldsymbol{\nu}_i)} \mathcal{L}(\boldsymbol{\theta}_i; S_i)$ using support sets.
14:           Update both network and context parameters:
          $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \alpha \nabla_{\boldsymbol{\theta}_i} \mathcal{L}(\boldsymbol{\theta}_i; S_i)$
          $\boldsymbol{\nu}_i \leftarrow \boldsymbol{\nu}_i - \beta \nabla_{\boldsymbol{\nu}_i} \mathcal{L}(\boldsymbol{\theta}_i; S_i)$
15:       **end for**
16:       Compute $\mathcal{L}(\boldsymbol{\theta}_i; Q_i)$ using query sets.
17:     **end for**
18:     Update the model parameters:
        $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\phi}) - \eta \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \sum_{i=1}^{B} \mathcal{L}(\boldsymbol{\theta}_i; Q_i)$
19: **end while**

Algorithm 2 Implicit Gradient Scaling (IGS)

Input: Model parameters $\boldsymbol{\theta}_i = \{\boldsymbol{\omega}_i, \boldsymbol{\psi}_i\}$
Input: Context parameters $\boldsymbol{\nu}_i$

1: Generate a set of scaling factors:
    $\{\gamma_i^{(\ell)}\}_{\ell=1}^{L} = g_\phi(\boldsymbol{\nu}_i)$
2: Scales the parameters of the backbone:
    $\boldsymbol{\omega}_i \leftarrow \left\{ \gamma_i^{(\ell)} \boldsymbol{\omega}_i^{(\ell)} / \|\boldsymbol{\omega}_i^{(\ell)}\| \right\}_{\ell=1}^{L}$
3: **Return** $\boldsymbol{\theta}_i$

---

scaled by the reciprocal of $\gamma_i^{(\ell)}$. Putting these two cases together, the scaling factor for the ℓ-th layer can be defined by $\|\boldsymbol{w}_i^{(\ell)}\| / \gamma_i^{(\ell)}$.

Next, let's look into the generation process of $\mathcal{G}$. g_ϕ that generates $\mathcal{G}$ is a two-layer MLP with ReLU and softplus. ReLU is located between the layers, and softplus is located at the end of the sub-network. The reason why we adopt softplus is that it also can output a positive number greater than 1. So the gradient norms also can be scaled in a decreasing direction. Besides, since softplus can be differentiated like the other layers in g_ϕ, the model can learn ϕ and the context parameters $\boldsymbol{\nu}_i$, which are the input to g_ϕ, in an end-to-end manner. Meanwhile, the context parameters $\boldsymbol{\nu}_i$ serve as a task embedding. If the learning rate for $\boldsymbol{\nu}_i$ is β, $\boldsymbol{\nu}_i$ is updated as follows:

$$\boldsymbol{\nu}_i \leftarrow \boldsymbol{\nu}_i - \beta \nabla_{\boldsymbol{\nu}_i} \mathcal{L}(\boldsymbol{\theta}_i; S_i) \tag{7}$$

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

$$\mathcal{G} = \{\gamma_i^{(\ell)}\}_{\ell=1}^{L} = g_\phi(\boldsymbol{\nu}_i) \tag{8}$$

$\boldsymbol{\nu}_i$ should only embed the representation of the i-th task. So, before adapting to the next task $\mathcal{T}_{i+1}$, $\boldsymbol{\nu}_i$ must be reset to a zero vector as in [35, 46].

Some may ask how about scaling gradient norms when updating $\boldsymbol{w}_i$. However, to update $\boldsymbol{\nu}_i$, forward and backward propagation must be performed twice each. The first forward and backward propagation indicate the process of including $\boldsymbol{\nu}_i$ in the computation graph (Eq. (6)) and the process of updating $\boldsymbol{\nu}_i$ (Eq. (7)), respectively. Then, $\mathcal{G}$ is generated with the updated $\boldsymbol{\nu}_i$ used in the second forward propagation. Finally, the gradients calculated by the second backward propagation are scaled, and then $\boldsymbol{\theta}_i$ is updated. On the other hand, the above process can be simplified by the BN property. Suppose we update $\boldsymbol{\nu}_i$ 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, $\nabla_{\boldsymbol{\nu}_i} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i; S_i)$ can be computed simultaneously with $\nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i; S_i)$ 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 [40]. Finally, various ablation studies are provided. In all the experiments below, four convolutional blocks, proposed by [41], 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.

4.1. Few-Shot Classification

Two famous benchmark datasets, i.e., miniImageNet [41] and tieredImageNet [29], are used for few-shot classification. MiniImageNet composed of a total of 60,000 images was obtained by extracting 100 classes from

Method	miniImageNet		tieredImageNet	
	1-shot	5-shot	1-shot	5-shot
MatchingNet [41]	$43.44 \pm 0.77\%$	$55.31 \pm 0.73\%$	–	–
RelationNet [37]	$50.44 \pm 0.82\%$	$65.32 \pm 0.70\%$	–	–
ProtoNet [36]	$49.42 \pm 0.78\%$	$68.20 \pm 0.66\%$	–	–
MAML [10]	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$	$49.06 \pm 0.50\%$	$67.48 \pm 0.47\%$
CAVIA (512) [46]	$51.82 \pm 0.65\%$	$65.85 \pm 0.55\%$	–	–
BOIL [25]	$49.61 \pm 0.16\%$	$66.45 \pm 0.37\%$	$48.58 \pm 0.27\%$	$69.37 \pm 0.12\%$
MAML++ [2]	$52.15 \pm 0.26\%$	$68.32 \pm 0.44\%$	–	–
ALFA [4]	$50.58 \pm 0.51\%$	$69.12 \pm 0.47\%$	$53.16 \pm 0.49\%$	$70.54 \pm 0.46\%$
ModGrad [35]	$\mathbf{53.20 \pm 0.86}\%$	$69.17 \pm 0.69\%$	–	–
L2F [5]	$52.10 \pm 0.50\%$	$69.38 \pm 0.46\%$	$54.40 \pm 0.50\%$	$73.34 \pm 0.44\%$
CxGrad (Ours)	$51.80 \pm 0.46\%$	$\mathbf{69.82 \pm 0.42}\%$	$\mathbf{55.55 \pm 0.46}\%$	$\mathbf{73.55 \pm 0.41}\%$

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.

ImageNet [31]. Each class contains 600 images of 84×84. Here, 100 classes are again divided into three folds, i.e., 64, 16, and 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-

Method	miniImageNet	
	$\rightarrow$ CUB	$\rightarrow$ CIFAR-FS
MAML	$52.70 \pm 0.32\%$	$55.82 \pm 0.50\%$
ALFA	$58.35 \pm 0.25\%$	$59.76 \pm 0.49\%$
BOIL	$60.84 \pm 0.45\%$	$63.28 \pm 0.45\%$
L2F	$60.89 \pm 0.22\%$	$63.73 \pm 0.48\%$
CxGrad (Ours)	$\mathbf{63.92 \pm 0.44}\%$	$\mathbf{64.85 \pm 0.44}\%$

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

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 [42] 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 [18]. 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 [25] is re-
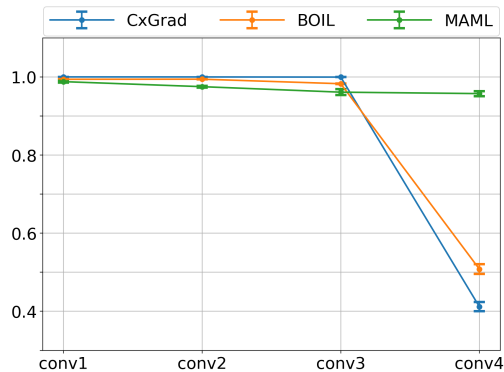
Figure 3: CKA values of three optimization-based meta-learning algorithms. The x-axis means each layer in the backbone and the y-axis means the CKA value.

Update scheme for ν_i	Accuracy
step-wise	$68.83 \pm 0.43\%$
task-wise	$\mathbf{69.82 \pm 0.42}\%$

Table 3: Effect of context parameter update scheme. The accuracy results from 5-way 5-shot on miniImageNet.

produced and the others are from [4] as they are. As argued in [25], 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 $p(\mathcal{T})$ 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 [26] and BOIL [25], the layer-wise feature similarity before and after adaptation was measured and compared in terms of CKA [17]. 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 [45]. 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 [40], 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, ν_i 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 $\mathcal{T}_i$. On the other hand, the step-wise method means updating the context parameters for each separate step. In this case, we have to reset ν_i 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%.

	Before Adaptation	After Adaptation	Before Adaptation	After Adaptation

(a) MAML   (b) CxGrad

Figure 4: Visualization for latent feature representation using t-SNE. The first row is visualizations for meta-test on miniImageNet and the second row is on CUB. Each shape represents each class and the latent features are from the output of the last convolutional block of the backbone.

Step	1	2	3	4	5
Accuracy	69.12	69.33	69.16	69.53	**69.82**

Table 4: Ablation study on the number of inner-loop steps.

	Context parameter lr (β)	Accuracy
1-shot	0.01	51.56
	0.1	51.29
	1.0	**51.80**
5-shot	0.01	69.28
	0.1	69.34
	1.0	**69.82**

Table 5: Ablation study on the learning rate for the context parameters.

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 (β) 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 β 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 mean-

ingfully 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

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *ICLR*, 2019.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. In *NeurIPS*, 2020.

[5] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *CVPR*, 2020.

[6] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, volume 2, 1992.

[7] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[8] Luca Bertinetto, João F. Henriques, Philip H.S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2018.

[9] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. In *NIPS*, 2017.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[16] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015.

[17] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

[18] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2016.

[24] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017.

[25] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. In *ICLR*, 2021.

[26] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *ICLR*, 2020.

[27] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[29] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[32] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[33] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[34] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[35] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. On modulating the gradient for meta-learning. In *ECCV*, 2020.

[36] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[37] 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.

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

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

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

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

[42] 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.

[43] 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.

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

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

[46] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *ICML*, 2019.