DropGrad: Gradient Dropout Regularization for Meta-Learning

Anonymous authors
Paper under double-blind review

Abstract

With the growing attention on learning-to-learn new tasks using only a few examples, meta-learning has been widely used in numerous problems such as few-shot classification, reinforcement learning, and domain generalization. However, meta-learning models are prone to overfitting when there are no sufficient training tasks for the meta-learners to generalize. Although existing approaches such as Dropout are widely used to address the overfitting problem, these methods are typically designed for regularizing models of a single task in supervised training. In this paper, we introduce a simple yet effective method to alleviate the risk of overfitting for gradient-based meta-learning. Specifically, during the gradient-based adaptation stage, we randomly drop the gradient in the inner-loop optimization of each parameter in deep neural networks, such that the augmented gradients improve generalization to new tasks. We present a general form of the proposed gradient dropout regularization and show that this term can be sampled from either the Bernoulli or Gaussian distribution. To validate the proposed method, we conduct extensive experiments and analysis on numerous tasks, demonstrating that the gradient dropout regularization mitigates the overfitting problem and improves the performance upon various gradient-based meta-learning frameworks.

1 Introduction

In recent years, significant progress has been made in meta-learning, which is also known as *learning to learn*. One common setting is that, given only a few training examples, meta-learning aims to learn new *tasks* rapidly by leveraging the past experience acquired from the known tasks. It is a vital machine learning problem due to the potential for reducing the amount of data and time for adapting an existing system. Numerous recent methods successfully demonstrate how to adopt meta-learning algorithms to solve various learning problems, such as few-shot classification (Finn et al., 2017; Santoro et al., 2016; Snell et al., 2017), reinforcement learning (Gupta et al., 2018; Rakelly et al., 2019), and domain generalization (Balaji et al., 2018; Li et al., 2018).

Despite the demonstrated success, meta-learning frameworks are prone to overfitting (Kim et al., 2018) when there do not exist sufficient training tasks for the meta-learners to generalize. For instance, few-shot classification on the mini-ImageNet (Vinyals et al., 2016) dataset contains only 64 training categories. Since the training tasks can be only sampled from this small set of classes, meta-learning models may overfit and fail to generalize to new testing tasks.

Significant efforts have been made to address the overfitting issue in the supervised learning framework, where the model is developed to learn a *single* task (e.g., recognize the same set of categories in both training and testing phase). The Dropout (Srivastava et al., 2014) method randomly drops (zeros) intermediate activations in deep neural networks during the training stage. Relaxing the limitation of binary dropout, the Gaussian dropout (Wang & Manning, 2013) scheme augments activations with noise sampled from a Gaussian distribution. Numerous methods (Ghiasi et al., 2018; Larsson et al., 2017; Tompson et al., 2015; Wan et al., 2013; Zoph et al., 2018) further improve the Dropout method by injecting structural noise or scheduling the dropout process to facilitate the training procedure. Nevertheless, these methods are developed to regularize the models to learn a single task, which may not be effective for meta-learning frameworks.

In this paper, we address the overfitting issue (Kim et al., 2018) in gradient-based meta-learning. As shown in Figure 1(a), given a new task, the meta-learning framework aims to adapt model parameters θ to be θ' via the gradients computed according to the few examples (support data $\mathcal{X}^s$). This gradient-based adaptation process is also known as the *inner-loop* optimization. To alleviate the overfitting issue, one straightforward approach is to apply the existing dropout method to the model weights directly. However, there are two sets of model parameters θ and θ' in the inner-loop optimization. As such, during the meta-training stage, applying normal dropout would cause inconsistent randomness, i.e., dropped neurons, between these two sets of model parameters. To tackle this issue, we propose a dropout method on the gradients in the inner-loop optimization, denoted as *DropGrad*, to regularize the training procedure. This approach naturally bridges θ and θ', and thereby involves only one randomness for the dropout regularization. We also note that our method is model-agnostic and generalized to various gradient-based meta-learning frameworks such as (Antoniou et al., 2019; Finn et al., 2017; Li et al., 2017). In addition, we demonstrate that the proposed dropout term can be formulated in a general form, where either the binary or Gaussian distribution can be utilized to sample the noise, as demonstrated in Figure 1(b).

To evaluate the proposed DropGrad method, we conduct experiments on numerous learning tasks, including few-shot classification on the mini-ImageNet (Vinyals et al., 2016), reinforcement learning (Finn et al., 2017), and online object tracking (Park & Berg, 2018), showing that DropGrad can be applied to and improve different tasks. In addition, we present comprehensive analysis by using various meta-learning frameworks, adopting different dropout probabilities, and explaining which layers to apply gradient dropout. To further demonstrate the generalization ability of DropGrad, we perform a challenging cross-domain few-shot classification task, in which the meta-training and meta-testing sets are from two different distributions, i.e., mini-ImageNet and CUB (Welinder et al., 2010). We show that with the proposed method, the performance is significantly improved under the cross-domain setting.

In this paper, we make the following contributions:

- We propose a simple yet effective gradient dropout approach to improve the generalization ability of gradient-based meta-learning frameworks.
- We present a general form for gradient dropout and show that both binary and Gaussian sampling schemes mitigate the overfitting issue.
- We demonstrate the effectiveness and generalizability of the proposed method via extensive experiments on numerous tasks.

2 Related Work

Meta-Learning. Meta-learning aims to adapt the past knowledge learned from previous tasks to new tasks with few training instances. Most meta-learning algorithms can be categorized into three groups: 1) Memory-based approaches (Rezende et al., 2016; Santoro et al., 2016) utilize recurrent networks to process few training examples of new tasks sequentially; 2) Metric-based frameworks (Oreshkin et al., 2018; Snell et al., 2017; Sung et al., 2018; Vinyals et al., 2016) make predictions by referring to the features encoded from the input data and training instances in a generic metric space; 3) Gradient-based methods (Antoniou et al., 2019; Finn et al., 2017; 2018; Kim et al., 2018; Li et al., 2017; Rusu et al., 2019; Ravi & Beatson, 2019) learn to optimize the model via gradient descent with few examples, which is the focus of this work. In the third group, MAML (Finn et al., 2017) learns model initialization (i.e., initial parameters) that is amenable to fast fine-tuning with few instances. In addition to model initialization, MetaSGD (Li et al., 2017) learns a set of learning rates for different model parameters. Furthermore, MAML++ (Antoniou et al., 2019) makes several improvements based on MAML to facilitate the training process with additional performance gain. However, these methods are still prone to overfitting as the dataset for the training tasks is insufficient for the model to adapt well. Recently, Kim et al. (Kim et al., 2018) and Rusu et al. (Rusu et al., 2019) address this issue via the Bayesian approach and latent embeddings. Nevertheless, these methods employ additional parameters or networks which entail significant computational overhead and may not be applicable to arbitrary frameworks. In contrast, the proposed gradient dropout regularization does not impose any overhead and thus can be readily integrated into the gradient-based models mentioned above.

Dropout Regularization. Built upon the Dropout (Srivastava et al., 2014) method, various schemes (Ghiasi et al., 2018; Goodfellow et al., 2013; Larsson et al., 2017; Tompson et al., 2015;

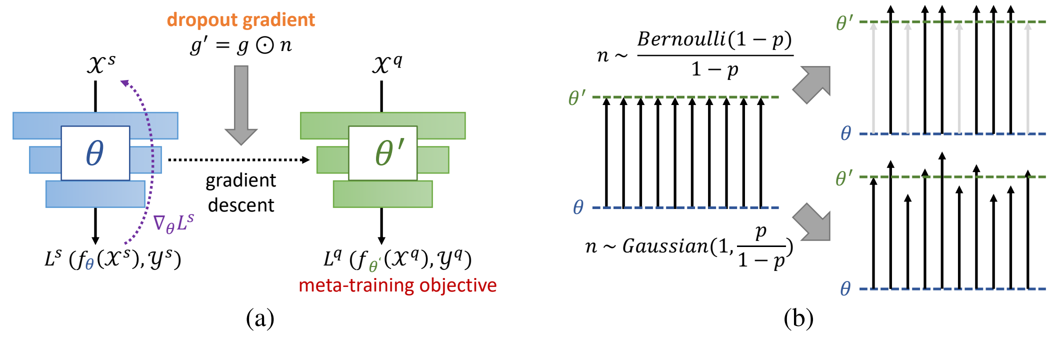
dropout gradient
$g' = g \odot n$
$\mathcal{X}^s$
θ
$\nabla_\theta L^s$
$L^s(f_\theta(\mathcal{X}^s), \mathcal{Y}^s)$
gradient descent
$\mathcal{X}^q$
θ'
$L^q(f_{\theta'}(\mathcal{X}^q), \mathcal{Y}^q)$
meta-training objective
(a)
$n \sim \frac{Bernoulli(1-p)}{1-p}$
$n \sim Gaussian(1, \frac{p}{1-p})$
θ'
θ
(b)

Figure 1: **Illustration of the proposed method.** (a) The proposed DropGrad method imposes a noise term n to augment the gradient in the inner-loop optimization during the meta-training stage. (b) The DropGrad method samples the noise term n from either the Bernoulli or Gaussian distribution, in which the Gaussian distribution provides a better way to account for uncertainty.

Wan et al., 2013) have been proposed to regularize the training process of deep neural networks for supervised learning. The core idea is to inject noise into intermediate activations when training deep neural networks. Several recent studies improve the regularization on convolutional neural networks by making the injected structural noise. For instance, the SpatialDropout (Tompson et al., 2015) method drops the entire channel from an activation map, the DropPath (Larsson et al., 2017; Zoph et al., 2018) scheme chooses to discard an entire layer, and the DropBlock (Ghiasi et al., 2018) algorithm zeros multiple continuous regions in an activation map. Nevertheless, these approaches are designed for deep neural networks that aim to learn a *single* task, e.g., learning to recognize a fixed set of categories. In contrast, our algorithm aims to regularize the gradient-based meta-learning frameworks that suffer from the overfitting issue on the *task*-level, e.g., introducing new tasks.

3 Gradient Dropout Regularization

Before introducing details of our proposed dropout regularization on gradients, we first review the gradient-based meta-learning framework.

3.1 Preliminaries for Meta-Learning

In meta-learning, multiple tasks $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ are divided into meta-training $\mathcal{T}^{\text{train}}$, meta-validation $\mathcal{T}^{\text{val}}$, and meta-testing $\mathcal{T}^{\text{test}}$ sets. Each task T_i consists of a support set $D^s = (\mathcal{X}^s, \mathcal{Y}^s)$ and a query set $D^q = (\mathcal{X}^q, \mathcal{Y}^q)$, where $\mathcal{X}$ and $\mathcal{Y}$ are a set of input data and the corresponding ground-truth. The support set D^s represents the set of few labeled data for learning, while the query set D^q indicates the set of data to be predicted.

Given a novel task and a parametric model f_θ, the objective of a gradient-based approach during the meta-training stage is to minimize the prediction loss L^q on the query set D^q according to the signals provided from the support set D^s, and thus the model f_θ can be adapted. Figure 1(a) show an overview of the MAML (Finn et al., 2017) method, which offers a general formulation of gradient-based frameworks. For each iteration of the meta-training phase, we first randomly sample a task $T = \{D^s, D^q\}$ from the meta-training set $\mathcal{T}^{\text{train}}$. We then adapt the initial parameters θ to be task-specific parameters θ' via gradient descent:

$$\theta' = \theta - \alpha \odot g, \tag{1}$$

where α is the learning rate for gradient-based adaptation and $\odot$ is the operation of element-wise product, i.e., Hadamard product. The term g in equation 1 is the set of gradients computed according to the objectives of model f_θ on the support set $D^s = (\mathcal{X}^s, \mathcal{Y}^s)$:

$$g = \bigtriangledown_\theta L^s(f_\theta(\mathcal{X}^s), \mathcal{Y}^s). \tag{2}$$

We call the step of equation 1 as the inner-loop optimization and typically, we can do multiple gradient steps for equation 1, e.g., smaller than 10 in general. After the gradient-based adaptation, the initial parameters θ are optimized according to the loss functions of the adapted model $f_{\theta'}$ on the query set $D^q = (\mathcal{X}^q, \mathcal{Y}^q)$:

$$\theta = \theta - \eta \bigtriangledown_\theta L^q(f_{\theta'}(\mathcal{X}^q), \mathcal{Y}^q), \tag{3}$$

where η is the learning rate for meta-training. During the meta-testing stage, the model f_θ is adapted according to the support set D^s and the prediction on query data $\mathcal{X}^q$ is made without accessing the ground-truth $\mathcal{Y}^q$ in the query set. We note that several methods are built upon the above formulation introduced in MAML. For example, the learning rate α for gradient-adaptation is viewed as the optimization objective (Antoniou et al., 2019; Li et al., 2017), and the initial parameters θ are not generic but conditional on the support set D^s (Rusu et al., 2019).

3.2 Gradient Dropout

The main idea is to impose uncertainty to the core objective during the meta-training step, i.e., the gradient g in the inner-loop optimization, such that θ' receives gradients with noises to improve the generalization of gradient-based models. As described in Section 3.1, adapting the model θ to θ' involves the gradient update in the inner-loop optimization described in equation 2. Based on this observation, we propose to randomly drop the gradient in equation 2, i.e., g, during the inner-loop optimization, as illustrated in Figure 1. Specifically, we augment the gradient g as follows:

$$g' = g \odot n, \tag{4}$$

where n is a noise regularization term sampled from a pre-defined distribution. With the formulation of equation 4, in the following we introduce two noise regularization strategies via sampling from different distributions, i.e., Bernoulli and Gaussian distributions.

Binary DropGrad. We randomly zero the gradient with the probability p, in which the process can be formulated as:

$$g' = g \odot n_b, \quad n_b \sim \frac{Bernoulli(1-p)}{1-p}, \tag{5}$$

where the denominator $1-p$ is the normalization factor. Note that, different from the Dropout (Srivastava et al., 2014) method which randomly drops the intermediate activations in a supervised learning network under a single task setting, we perform the dropout on the gradient level.

Gaussian DropGrad. One limitation of the Binary DropGrad scheme is that the noise term n_b is only applied in a binary form, which is either 0 or $1-p$. To address this disadvantage and better provide a better regularization with uncertainty, we extend the Bernoulli distribution to the Gaussian formulation. Since the expectation and variance of the noise term n_b in the Binary DropGrad method are respectively $\mathrm{E}(n_b) = 1$ and $\sigma^2(n_b) = \frac{p}{1-p}$, we can augment the gradient g with noise sampled from the Gaussian distribution:

$$g' = g \odot n_g, \quad n_g \sim Gaussian(1, \frac{p}{1-p}). \tag{6}$$

As a result, two noise terms n_b and n_g are statistically comparable with the same dropout probability p. In Figure 1(b), we illustrate the difference between the Binary DropGrad and Gaussian DropGrad approaches. We also show the process of applying the proposed regularization using the MAML (Finn et al., 2017) method in Algorithm 1, while similar procedures can be applied to other gradient-based meta-learning frameworks, such as MetaSGD (Li et al., 2017) and MAML++ (Antoniou et al., 2019).

4 Experimental Results

In this section, we evaluate the effectiveness of the proposed DropGrad method by conducting extensive experiments on three learning problems: few-shot classification, reinforcement learning, and online object tracking. In addition, for the few-shot classification experiments, we analyze the effect of using binary and Gaussian noise, which layers to apply DropGrad, and performance in the cross-domain setting. The source code and trained models will be made available to the public.

Algorithm 1: Applying DropGrad on MAML (Finn et al., 2017)

```
1 Require: a set of training tasks $\mathcal{T}^{\text{train}}$, adaptation learning rate $\alpha$, meta-learning rate $\eta$
2 randomly initialize $\theta$
3 while training do
4     randomly sample a task $T = \{D^s(\mathcal{X}^s, \mathcal{Y}^s), D^q(\mathcal{X}^q, \mathcal{Y}^q)\}$ from $\mathcal{T}^{\text{train}}$
5     $g = \bigtriangledown_\theta L^s(f_\theta(\mathcal{X}^s), \mathcal{Y}^s)$
6     compute $g'$ according to equation 5 or equation 6      // Apply DropGrad
7     $\theta' = \theta - \alpha \times g'$
8     $\theta = \theta - \eta \bigtriangledown_\theta L^q(f_{\theta'}(\mathcal{X}^q), \mathcal{Y}^q)$
9 end
```

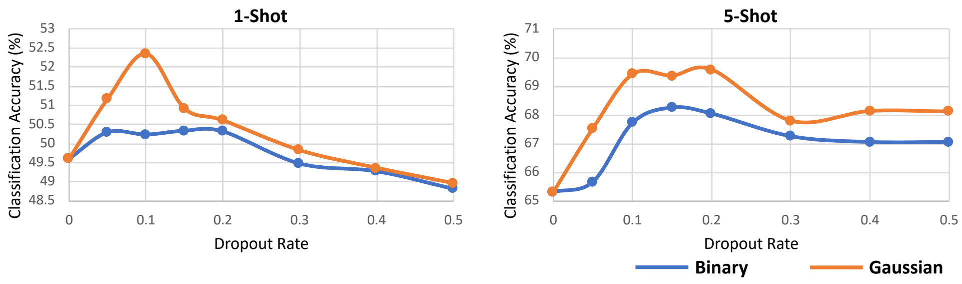

Figure 2: **Comparison between the proposed Binary and Gaussian DropGrad methods.** We compare the 1-shot (*left*) and 5-shot (*right*) performance of MAML (Finn et al., 2017) trained with two different forms of DropGrad under various dropout rates on mini-ImageNet. The proposed DropGrad method is particularly effective with the dropout rate in $[0.1, 0.2]$. Moreover, the Gaussian DropGrad method consistently obtains better results compared to the Binary DropGrad scheme.

4.1 Few-Shot Classification

Few-shot classification aims to recognize a set of new categories, e.g., five categories (5-way classification), with few, e.g., one (1-shot) or five (5-shot), example images from each category. In this setting, the support set D^s contains the few images $\mathcal{X}^s$ of the new categories and the corresponding categorical annotation $\mathcal{Y}^s$. We conduct experiments on the mini-ImageNet dataset (Vinyals et al., 2016), which is widely used for evaluating few-shot classification approaches. As a subset of the ImageNet (Deng et al., 2009), the mini-ImageNet dataset contains 100 categories and 600 images for each category. We use the 5-way evaluation protocol in (Ravi & Larochelle, 2017) and split the dataset into 64 training, 16 validating, and 20 testing categories.

Implementation Details. We apply the proposed DropGrad regularization method to train the following gradient-based meta-learning frameworks: MAML (Finn et al., 2017), MetaSGD (Li et al., 2017), and MAML++ (Antoniou et al., 2019). We use the implementation from Chen et al. (Chen et al., 2019) for MAML and use our own implementation for MetaSGD.[1] We use the ResNet-18 (He et al., 2016) model as the backbone network for both MAML and MetaSGD. As for MAML++, we use the original source code.[2]

Similar to recent studies, e.g., (Rusu et al., 2019), we also pre-train the feature extractor of ResNet-18 by minimizing the classification loss on the 64 training categories from the mini-ImageNet dataset for the MetaSGD method, which is denoted by MetaSGD*. For all the experiments, we use the default hyper-parameter settings provided by the original implementation and select the model according to the validation performance for evaluation.

Comparison between Binary and Gaussian DropGrad. We first evaluate how the proposed Binary and Gaussian DropGrad methods perform on the MAML framework with different values of the dropout probability p. Figure 2 shows that both methods are effective especially when the dropout

[1]https://github.com/wyharveychen/CloserLookFewShot
[2]https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch

Table 1: **Few-shot classification results on mini-ImageNet.** The Gaussian DropGrad method improves the performance of gradient-based models on 1-shot and 5-shot classification tasks.

Model	1-shot	5-shot
MAML (Finn et al., 2017)	$49.61 \pm 0.92\%$	$65.72 \pm 0.77\%$
MAML w/ Gaussian DropGrad ($p = 0.1$)	$\mathbf{52.35 \pm 0.86\%}$	$\mathbf{69.42 \pm 0.73\%}$
MetaSGD (Li et al., 2017)	$51.51 \pm 0.87\%$	$69.67 \pm 0.75\%$
MetaSGD w/ Gaussian DropGrad ($p = 0.1$)	$\mathbf{53.38 \pm 0.93\%}$	$\mathbf{71.14 \pm 0.72\%}$
MetaSGD*	$60.44 \pm 0.87\%$	$72.55 \pm 0.54\%$
MetaSGD* w/ Gaussian DropGrad ($p = 0.1$)	$\mathbf{61.69 \pm 0.84\%}$	$\mathbf{73.33 \pm 0.57\%}$
MAML++ (Antoniou et al., 2019)	$50.21 \pm 0.50\%$	$68.66 \pm 0.46\%$
MAML++ w/ Gaussian DropGrad ($p = 0.2$)	$\mathbf{51.13 \pm 0.50\%}$	$\mathbf{69.80 \pm 0.46\%}$

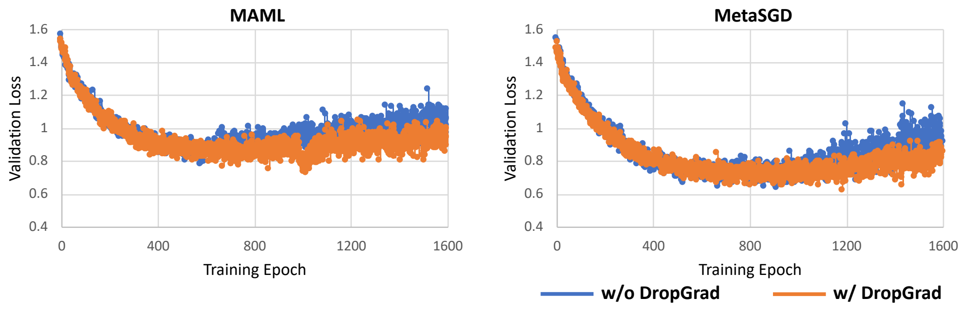

Figure 3: **Validation loss over training epochs.** We show the validation curves of the MAML (*left*) and MetaSGD (*right*) frameworks trained on the 5-shot mini-ImageNet dataset. The curves validate that the proposed DropGrad method alleviates the overfitting problem.

rate is in the range of $[0.1, 0.2]$, while setting the dropout rate to 0 is to turn the proposed DropGrad method off. Since the problem of learning from only one instance (1-shot) is more complicated, the overfitting effect is less severe compared to the 5-shot setting. As a result, applying the DropGrad method with a dropout rate larger than 0.3 degrades the performance. Moreover, the Gaussian DropGrad method consistently outperforms the binary case on both 1-shot and 5-shot tasks, due to a better regularization term n_g with uncertainty. We then apply the Gaussian DropGrad method with the dropout rate of 0.1 or 0.2 in the following experiments.

Comparison with existing dropout methods. To show that the proposed DropGrad method is effective for gradient-based meta-learning frameworks, we compare it with two existing dropout schemes applied on the network activations in both f_θ and f'_θ. We choose the Dropout (Srivastava et al., 2014) and SpatialDropout (Tompson et al., 2015) methods, since the former is a commonly-used approach while the latter is shown to be effective for applying to 2D convolutional maps. The performance of MAML on 5-shot classification on the mini-ImageNet dataset is: *DropGrad* $69.42 \pm 0.73\%$, *SpatialDropout* $68.09 \pm 0.56\%$, and *Vanilla Dropout* $67.44 \pm 0.57\%$. This demonstrates the benefit of using the proposed DropGrad method, which effectively tackles the issue of inconsistent randomness between two different models f_θ and f'_θ in the inner-loop optimization of gradient-based meta-learning frameworks.

Overall performance on the mini-ImageNet dataset. Table 1 shows the results of applying the proposed Gaussian DropGrad method to different frameworks. The results validate that the proposed regularization scheme consistently improves the performance of various gradient-based meta-learning approaches. In addition, we present the curve of validation loss over training episodes from MAML and MetaSGD on the 5-shot classification task in Figure 3. We observe that the overfitting problem is more severe in training the MetaSGD method since it consists of more parameters to be

Table 2: **Performance of applying DropGrad to different layers.** We conduct experiments on the 5-shot classification task using MAML on mini-ImageNet. It is more helpful in improving the performance by dropping the gradients closer to the output layers (e.g., FC and Block4+FC).

Origin	FC	Block4 + FC	Full	Block1 + Conv	Conv
$65.72 \pm 0.77\%$	$68.93 \pm 0.55\%$	$69.02 \pm 0.57\%$	$69.42 \pm 0.73\%$	$64.96 \pm 0.80\%$	$65.53 \pm 0.75\%$

Table 3: **Cross-domain performance for few-shot classification.** We use the mini-ImageNet and CUB datasets for the meta-training and meta-testing steps, respectively. The improvement of applying the proposed DropGrad method is more significant in the cross-domain cases than the intra-domain ones.

Model	1-Shot	5-Shot
MAML (Finn et al., 2017)	$31.52 \pm 0.52\%$	$45.56 \pm 0.51\%$
MAML w/ DropGrad ($p = 0.1$)	$\mathbf{33.20 \pm 0.67\%}$	$\mathbf{51.05 \pm 0.56\%}$
MetaSGD (Li et al., 2017)	$34.52 \pm 0.63\%$	$49.22 \pm 0.58\%$
MetaSGD w/ DropGrad ($p = 0.1$)	$\mathbf{36.77 \pm 0.72\%}$	$\mathbf{55.13 \pm 0.72\%}$
MetaSGD*	$43.98 \pm 0.77\%$	$57.95 \pm 0.81\%$
MetaSGD* w/ Gaussian DropGrad ($p = 0.1$)	$\mathbf{45.33 \pm 0.81\%}$	$\mathbf{59.94 \pm 0.82\%}$
MAML++ (Antoniou et al., 2019)	$40.73 \pm 0.49\%$	$60.57 \pm 0.49\%$
MAML++ w/ DropGrad ($p = 0.2$)	$\mathbf{44.27 \pm 0.50\%}$	$\mathbf{63.79 \pm 0.48\%}$

optimized. The DropGrad regularization method mitigates the overfitting issue and facilitates the training procedure.

Layers to apply DropGrad. We study which layers in the network to apply the DropGrad regularization in this experiment. The backbone ResNet-18 model contains a convolutional layer (Conv) followed by 4 residual blocks (Block1, Block2, Block3, Block4) and a fully-connected layer (FC) as the classifier. We perform the Gaussian DropGrad method on different parts of the ResNet-18 model for MAML on the 5-shot classification task. The results are presented in Table 2. We find that it is more critical to drop the gradients closer to the output layers (e.g., FC and Block4+FC). Applying the DropGrad method to the input side (e.g., Block1+Conv and Conv), however, may even negatively affect the training and degrade the performance. This can be explained by the fact that features closer to the output side are more abstract and thus tend to overfit. As using the DropGrad regularization term only increases a negligible overhead, we use the *Full* model, where our method is applied to all the layers in the experiments unless otherwise mentioned.

4.2 Cross-Domain Few-Shot Classification

To further evaluate how the proposed DropGrad method improves the generalization ability of gradient-based meta-learning models, we conduct a cross-domain experiment, in which the meta-testing set is from an *unseen* domain. We use the cross-domain scenario introduced by Chen et al. (Chen et al., 2019), where the meta-training step is performed on the mini-ImageNet dataset while the meta-testing evaluation is conducted on the CUB dataset (Hilliard et al., 2018). Note that, different from Chen et al. (Chen et al., 2019) who select the model according to the validation performance on the CUB dataset, we pick the model via the validation performance on the mini-ImageNet dataset for evaluation. The reason is that we target at analyzing the generalization ability to the unseen domain, and thus we do not utilize any information provided from the CUB dataset.

Table 3 shows the results using the Gaussian DropGrad method. Since the domain shift in the cross-domain scenario is larger than that in the intra-domain case (i.e., both training and testing tasks are sampled from the mini-ImageNet dataset), the performance gains of applying the proposed DropGrad method reported in Table 3 are more significant than those in Table 1. The results demonstrate that the DropGrad scheme is able to effectively regularize the gradients and transfer them for learning new tasks in an unseen domain.

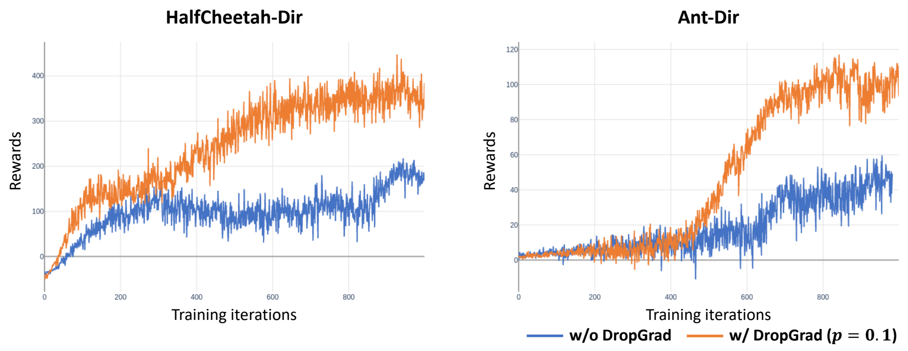

Figure 4: **Few-shot reinforcement learning results.** Two settings, HalfCheetah-Dir (*left*) and Ant-Dir (*right*), are considered in our experiments using the MAML-TPRO framework. We show the reward curves after the model is updated with the few trajectories and both rewards converge favorably against the original training.

Table 4: **Precision and success rate on the OTB2015 dataset.** The DropGrad method can be applied to visual tracking and improve the tracking performance.

Model	Precision	Success rate
MetaCREST (Park & Berg, 2018)	0.7994	0.6029
MetaCREST w/ DropGrad ($p = 0.2$)	**0.8172**	**0.6145**
MetaSDNet (Park & Berg, 2018)	0.8673	0.6434
MetaSDNet w/ DropGrad ($p = 0.2$)	**0.8746**	**0.6520**

4.3 Reinforcement Learning

We adopt the few-shot reinforcement learning (RL) setting as in (Finn et al., 2017), which aims to make the system adapt to new experiences and learn the corresponding policy quickly with limited prior experience (i.e., trajectories). In this setting, the support set D^s contains few trajectories and the corresponding rewards, while the query set D^q is formed by a set of new trajectories sampled from the running policy. We conduct the experiment with the locomotion tasks simulated by the MuJoCo (Todorov et al., 2012) simulator. Two environments are considered in this experiment: HalfCheetah robot and Ant robot with forward/backward movement, i.e., HalfCheetah-Dir and Ant-Dir.

Implementation Details. We adopt the MAML-TPRO (Finn et al., 2017) framework as the baseline method. Since the rewards are usually not differentiable, policy gradients are calculated for adapting the RL models to new experiences in both inner- and outer-loop optimization. For applying the proposed DropGrad scheme in the RL framework, we augment the policy gradients calculated according to rewards in the support set during the inner-loop optimization. We use a public Pytorch implementation with the default hyper-parameter setting in the experiments.[3]

Reinforcement Learning Results. In Figure 4, we present the rewards after the model is optimized with the few trajectories, i.e., in each iteration we perform one-step policy gradient update for the inner-loop optimization. In both environments, the training process with the proposed DropGrad regularization method converges to favorable rewards compared to the original training without the proposed regularization. This improvement could be attributed by the uncertainty on gradients that provide a better exploration of the policy.

[3]https://github.com/tristandeleu/pytorch-maml-rl

4.4 Online Object Tracking

Visual object tracking targets at localizing one particular object in a video sequence given the bounding box annotation in the first frame. To adapt the model to the subsequent frames, one approach is to apply online adaptation during tracking. The Meta-Tracker (Park & Berg, 2018) method uses meta-learning to improve two state-of-the-art online trackers, including the correlation-based CREST (Song et al., 2017) and the detection-based MDNet (Nam & Han, 2016), which are denoted as MetaCREST and MetaSDNet. Based on the error signals from future frames, the Meta-Tracker updates the model during offline meta-training, and obtains a robust initial network that generalizes well over future frames. We apply the proposed DropGrad method to train the MetaCREST and MetaSDNet models with evaluation on the OTB2015 (Wu et al., 2015) dataset.

Implementation Details. We train the models using the original source code.[4] For meta-training, we use a subset of a large-scale video detection dataset (Russakovsky et al., 2015), and the 58 sequences from the VOT2013 (Kristan et al., 2013), VOT2014 (Kristan et al., 2014) and VOT2015 (Kristan et al., 2015) datasets, excluding the sequences in the OTB2015 database, based on the same settings in the Meta-Tracker (Park & Berg, 2018). We use the default hyper-parameter settings and evaluate the performance with the models at the last training iteration.

Object Tracking Results. The results of online object tracking on the OTB2015 dataset are presented in Table 4. The one-pass evaluation (OPE) protocol without restarts at failures is used in the experiments. We measure the precision and success rate based on the center location error and the bounding-box overlap ratio, respectively. The precision is calculated with a threshold 20, and the success rate is the averaged value with the threshold ranging from 0 to 1 with a step of 0.05. We show that applying the proposed DropGrad method consistently improves the performance in precision and success rate on both MetaCREST and MetaSDNet trackers.

5 Conclusions

In this work, we propose a simple yet effective gradient dropout approach for regularizing the training of gradient-based meta-learning frameworks. The core idea is to impose uncertainty by augmenting the gradient in the adaptation step during meta-training. We propose two forms of noise regularization terms, including Bernoulli and Gaussian distributions, and demonstrate that the proposed DropGrad improves the model performance in three learning tasks. In addition, extensive analysis and studies are provided to further understand the benefit of our method. One study on cross-domain few-shot classification is also conducted to show that the DropGrad method is able to mitigate the overfitting issue under a larger domain gap.

References

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *ICLR*, 2019. 2, 4, 5, 6, 7

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018. 1

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019. 5, 7, 12

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018. 2

[4]https://github.com/silverbottlep/meta_trackers

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, 2018. 1, 2, 3

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, 2013. 2

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *NeurIPS*, 2018. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018. 7

Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018. 1, 2

D Kinga and J Ba Adam. Adam: A method for stochastic optimization. In *ICLR*, 2015. 12

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. 13

Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Fatih Porikli, Luka Čehovin Zajc, Georg Nebehay, Gustavo Fernandez, and Tomas Vojir et al. The visual object tracking vot2013 challenge results. In *ICCV Workshop*, 2013. 9

Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin Zajc, Georg Nebehay, Tomas Vojir, Gustavo Fernandez, and Alan Lukežič et al. The visual object tracking vot2014 challenge results. In *ECCV Workshop*, 2014. 9

Matej Kristan, Jiri Matas, Aleš Leonardis, Michael Felsberg, Luka Čehovin Zajc, Gustavo Fernandez, Tomas Vojir, Gustav Häger, and Georg Nebehay et al. The visual object tracking vot2015 challenge results. In *ICCV Workshop*, 2015. 9

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 1, 2, 3

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018. 1

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. 2, 4, 5, 6, 7, 12

Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 9

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018. 2

Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, 2018. 2, 8, 9, 12

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019. 1

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *ICLR*, 2019. 2

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 5, 12

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. *JMLR*, 48, 2016. 2

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. *IJCV*, 2015. 9

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. 2, 4, 5

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 1, 2

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017. 1, 2

Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson W. H. Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *ICCV*, 2017. 9

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 1, 2, 4, 6

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 2

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, 2012. 8

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. 1, 2, 3, 6

Hung-Yu Tseng, Shalini De Mello, Jonathan Tremblay, Sifei Liu, Stan Birchfield, Ming-Hsuan Yang, and Jan Kautz. Few-shot viewpoint estimation. In *BMVC*, 2019. 12, 13

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016. 1, 2, 5

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 1, 3

Sida Wang and Christopher Manning. Fast dropout training. In *ICML*, 2013. 1

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 2

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 2015. 9

Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *ECCV*, 2016. 12

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 3

A Appendix

In this appendix, we first supplement the implementation details. We then present additional experimental results of few-shot viewpoint estimation, few-shot classification, and object online tracking. Finally, we compare the proposed DropGrad regularization algorithm with the simulated annealing methods.

A.1 Supplementary Implementation Details

Few-shot classification. We use the implementation from Chen et al. (2019) to train and evaluate MAML (Finn et al., 2017) on few-shot classification tasks.[5] In the meanwhile, we modify the same implementation for MetaSGD (Li et al., 2017) by ourselves. We verify our implementation by evaluating the MetaSGD model using Conv4, which is the same backbone network adopted in the original paper. The 5-way 5-shot classification results on the mini-ImageNet dataset (Ravi & Larochelle, 2017) reported by our implementation and the original paper are $65.31 \pm 0.66\%$ and $64.03 \pm 0.94\%$, respectively.

To train both the MAML and MetaSGD models, we keep the default settings in the original implementation by Chen et al. (2019). We apply the Adam (Kinga & Adam, 2015) optimizer with the learning rate of 0.001. The mini-batch size is set to be 4. We train the model with 400 epochs and do not apply the learning rate decay strategy.

Online object tracking. We conduct experiments of online object tracking based on the PyTorch implementation by Park & Berg (2018). For MetaSDNet, the first three Convolutional layers of VGG-16 are used as the feature extractor. During meta-training, the last three fully-connected layers are randomly initialized. We only update the last three fully-connected layers in the first 5,000 iterations, and then train the entire network for the remaining iterations. We adopt Adam optimizer (Kinga & Adam, 2015) with an initial learning rate of 10^{-4}, and decrease the learning rate to 10^{-5} after $10,000$ iterations. In total, we train the network for $15,000$ iterations. For MetaCREST, we use Adam optimizer with a learning rate of 10^{-6}, and train the model for $10,000$ iterations.

A.2 Few-Shot Viewpoint Estimation

Viewpoint estimation aims to estimate the viewpoint (i.e., 3D rotation), denoted as $R \in \mathrm{SO}(3)$, between the camera and the object of a specific category in the image. Given a few examples (i.e., 10 images in this work) of a novel category with viewpoint annotations, few-shot viewpoint estimation attempts to predict the viewpoint of arbitrary objects of the same category. In this problem, the support set D^s contains few images $\mathbf{x}^s$ of a new class and the corresponding viewpoint annotations $\mathbf{y}^s$. We conduct experiments on the ObjectNet3D dataset (Xiang et al., 2016), a viewpoint estimation benchmark dataset which contains 100 categories. Using the same evaluation protocol in (Tseng et al., 2019), we extract 76 and 17 categories for training and testing, respectively.

Implementation Details. We apply the proposed DropGrad on the MetaView (Tseng et al., 2019) method, which is a meta-Siamese viewpoint estimator that applies gradient-based adaptation for novel categories. We obtain the source code from the authors, and keep all the default setting for training. Since there is no validation set available, we pick the model trained in the last epoch for evaluation.

Viewpoint Estimation Results. We show the viewpoint estimation results in Table 5. The evaluation metrics include Acc30 and MedErr which represent the percentage of viewpoints with rotation error under 30° and the median rotation error, respectively. The overall performance is improved by applying the proposed DropGrad method to the MetaView model during training.

A.3 Few-Shot Classification

In all experiments shown in Section 4, we use the default hyper-parameter values from the original implementation of the adopted methods. In this experiment, we explore the hyper-parameter choices

[5] https://github.com/wyharveychen/CloserLookFewShot

Table 5: **Viewpoint estimation results.** The DropGrad method can be applied to few-shot viewpoint estimation frameworks to mitigate the overfitting problem.

Model	Acc30 (↑)	MedErr (↓)
MetaView (Tseng et al., 2019)	$45.00 \pm 0.45\%$	$33.60 \pm 0.94°$
MetaView w/ DropGrad ($p = 0.1$)	$\mathbf{46.16 \pm 0.55\%}$	$\mathbf{33.10 \pm 0.82°}$

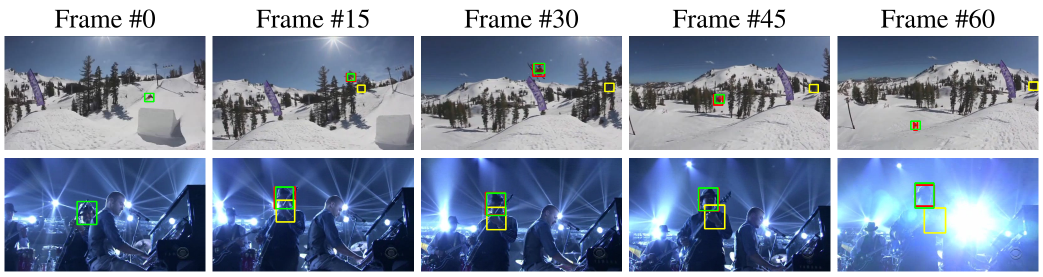

Figure 5: **Qualitative results of object online tracking on the OTB2015 dataset.** Top and bottom rows show the sample results of MetaCREST and MetaSDNet, respectively. Red boxes are the ground truth, yellow boxes represent the original results, and green boxes stand for the results where the DropGrad method is applied.

for MAML (Finn et al., 2017). Specifically, we conduct an ablation study on the learning rate α and the number of inner-loop optimizations n_{inner} in MAML. As shown in Table 6, the proposed DropGrad method improves the performance consistently under different sets of hyper-parameters.

Table 6: 5**-shot classification results of MAML under various hyper-parameter settings.** We study the learning rate α and number of iterations n_{inner}in the inner-loop optimization of MAML using mini-ImageNet dataset.

α, n_{inner}	$0.01, 5$ (original)	$0.1, 5$	$0.001, 5$	$0.01, 3$	$0.01, 7$
MAML (Finn et al., 2017)	$65.72 \pm 0.77\%$	$65.98 \pm 0.79\%$	$58.55 \pm 0.80\%$	$64.84 \pm 0.80\%$	$68.11 \pm 0.74\%$
MAML w/ DropGrad ($p = 0.1$)	$\mathbf{69.42 \pm 0.73\%}$	$\mathbf{67.78 \pm 0.73\%}$	$\mathbf{64.05 \pm 0.79\%}$	$\mathbf{65.42 \pm 0.80\%}$	$\mathbf{69.65 \pm 0.70\%}$

A.4 Object Online Tracking

We present sample results of object online tracking in Figure 5. We apply the proposed DropGrad method on the MetaCREST and MetaSDNet methods and evaluate these models on the OTB2015 dataset. Compared with the original MetaCREST and MetaSDNet, models trained with the DropGrad method track objects more accurately.

A.5 Comparison to Simulated Annealing

The proposed DropGrad algorithm is also related to simulated annealing (SA) (Kirkpatrick et al., 1983). While conceptually similar to a certain extent, the goals and formulations are significantly different. SA modulates gradients by exploring uncertain solutions with the goal to escape from the local minimum during the training stage. On the other hand, our DropGrad method, drops the *inner* gradient to introduce uncertainty in the forward pass of the gradient-based meta-learning framework.