A Nested Bi-level Optimization Framework for Robust Few Shot Learning

Krishnateja Killamsetty*
The University of Texas at Dallas
krishnateja.killamsetty@utdallas.edu

Changbin Li*
The University of Texas at Dallas
changbin.li@utdallas.edu

Chen Zhao, Feng Chen, Rishabh Iyer
The University of Texas at Dallas
{chen.zhao, feng.chen, rishabh.iyer}@utdallas.edu

Abstract

Model-Agnostic Meta-Learning (MAML), a popular gradient-based meta-learning framework, assumes that the contribution of each task or instance to the meta-learner is equal. Hence, it fails to address the domain shift between base and novel classes in few-shot learning. In this work, we propose a novel robust meta-learning algorithm, NESTEDMAML, which learns to assign weights to training tasks or instances. We consider weights as hyper-parameters and iteratively optimize them using a small set of validation tasks set in a nested bi-level optimization approach (in contrast to the standard bi-level optimization in MAML). We then apply NESTEDMAML in the meta-training stage, which involves (1) several tasks sampled from a distribution different from the meta-test task distribution, or (2) some data samples with noisy labels. Extensive experiments on synthetic and real-world datasets demonstrate that NESTEDMAML efficiently mitigates the effects of "unwanted" tasks or instances, leading to significant improvement over the state-of-the-art robust meta-learning methods.

1 Introduction

Meta-learning [26, 17, 25, 30, 7] can achieve quick adaption for UNSEEN tasks by identifying common structures among various SEEN tasks, enabling faster learning of a new task with as little data as possible. However, existing meta-learning techniques (*e.g.,* MAML [7]) often fail to generalize well when the test tasks belong to a different distribution from the training tasks distribution [4]. For example, MAML assumes equal weights to all samples and tasks during meta-training. This task homogeneity assumption of MAML often limits its ability to work in real-world applications.

We motivate the importance of robust meta-learning when meta-training tasks have OOD tasks using the following examples. For example, consider the task of detecting vehicles at night under different weather conditions. In this case, the meta-test tasks only consist of images of vehicles at night. Since the procurement vehicles driving data at night, covering all critical scenarios is difficult, we need a model that can quickly adapt to rare driving conditions. Hence, we consider meta-training tasks to consist of images of the vehicles in multiple lighting scenarios. In this case, some of the tasks in meta-training may degrade the meta-test performance. So, it is vital to have a meta-learning model that is robust to OODs.

Another example is rare lung cancer detection from medical x-ray images. Since the procurement of rare lung cancer images is both problematic and expensive, it is beneficial to use prior knowledge

*Equal contribution.

5th Workshop on Meta-Learning at NeurIPS 2021, held virtually.

of cancer images. Specifically, the meta-test tasks contain images of rare lung cancer, whereas meta-training tasks consist of general cancer x-ray images. In the examples given above, meta-test tasks belong to specialized slices where the data availability is meager compared to meta-training tasks. The meta-training task distribution is biased compared to that in the meta-test. To keep the whole meta-training tasks for generalization and reduce the adverse impact of the biased distribution in meta-training, we propose a novel robust few-shot learning algorithm in the presence of outliers in meta-training time, which is similar to the corruptions in training time in the traditional robust learning [27]. This is different from the existing robust few-shot learning papers [35, 15, 10] which consider the corruption only happens in meta-test time.

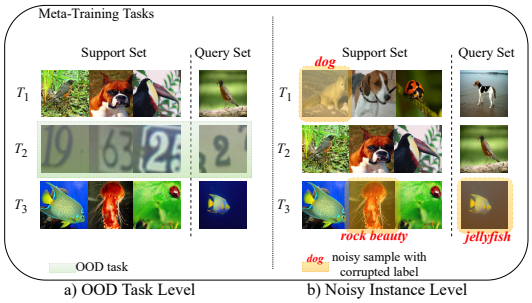


Figure 1: We consider corrupted training set for few-shot learning: a) OOD task level and b) noisy instance level. T_2 in a) is an OOD task that is sampled from a different distribution. b) contains some noisy samples which are mislabeled. For example, the actual label of the first sample in T_1 should be "*Arctic fox*" which is labeled as "*dog*"; The labels of two noisy samples in T_3 are flipped wrongly. The first one should be "*rock beauty*", and the other one should be "*jellyfish*".

To simulate the corruptions in meta-training, two levels of outliers (Figure 1) are considered: *a*) Out-Of-Distribution (OOD) task, where the meta-training has tasks that are out of distribution to the meta-test tasks (i.e., the meta-test dataset is a specialized slice of meta-train) and *b*) noisy instance level, where some labels might be noisy (due to human labeling errors or inherent ambiguity of certain classification problems) for meta-training samples.

A natural way of dealing with corrupted data in meta-training is by assigning weights to either tasks or individual instances. For example, assigning zero weight to OOD or noisy tasks/instances in the meta-train set improves the meta-learning algorithm's performance. Inspired by [23], in this work, we propose an end-to-end robust meta-learning framework called NESTEDMAML that can achieve the reweighting schema along with learning good model initialization parameters in the few-shot learning scenario.

NESTEDMAML considers the weights as hyper-parameters and uses a small set of meta-validation tasks representing the meta-test tasks to find the optimal hyper-parameters by minimizing the meta-loss on the validation tasks in a **nested bi-level** manner. An overview of NESTEDMAML is given in Figure 2. In practice, the size of the meta-validation tasks set required by NESTEDMAML is tiny compared to the meta-training dataset. Hence, creating a small and clean meta-validation set is neither expensive nor unrealistic, even for rare specialized use cases of a real-life scenario. A similar strategy has been applied in [23, 28, 13]. However, they focus on traditional supervised learning, and we generalize this to task- and instance-level in a meta-learning setting. Since NESTEDMAML uses an online framework to perform a joint optimization of the weight hyper-parameters and model parameters for the weighted MAML model, the computational time of ours is comparable to MAML.

Contributions of our work are summarized as follows: 1) We study the general form of the task and instance weighted meta-learning, where we learn the optimal weights and model initialization parameters by optimizing a *nested bi-level* objective function. To the best of our knowledge, ours is the first work that studies the *nested bi-level* optimization problem, which comes naturally in such a new setting. 2) We introduce a novel algorithmic framework NESTEDMAML that uses a small set of validation tasks to enable robust meta-learning. We solve the *nested bi-level* optimization problem efficiently through a series of practical approximations and provide a theoretical convergence analysis for NESTEDMAML. In particular, we show that NESTEDMAML converges in $\mathcal{O}(1/\epsilon^2)$ iterations under reasonable assumptions and contrast this with existing bounds of MAML. 3) We provide comprehensive synthetic and real-world data experiments demonstrating that NESTEDMAML achieves state-of-the-art results in two scenarios (OOD tasks and noisy instance labels).

2 Related Work

There are several lines of meta-learning algorithms: nearest neighbors-based methods [30], recurrent network-based methods [22], and gradient-based methods. As the representative of gradient-based

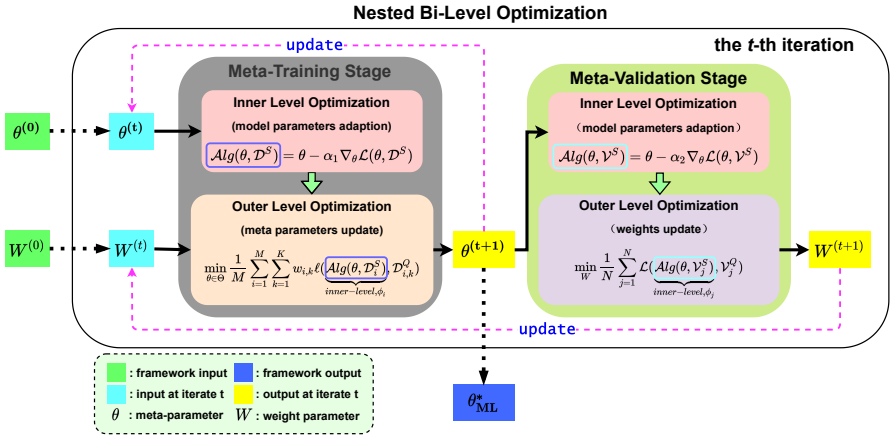


Figure 2: Overview of our NESTEDMAML framework that solves a *nested bi-level* optimization problem. (a) In the meta-training stage, model parameters ϕ_i of each task are adapted from meta-parameter $\boldsymbol{\theta}$ through the inner level optimization; (b) In the outer-level of the meta-training stage, we update the meta-parameters using the weights W from the previous iterate; (c) Weights are further updated in the meta-validation stage using the gradient of the meta-losses with respect to current W.

meta-learning algorithms, MAML [7] and its variants [8, 19, 24, 21, 2, 20, 36, 37] learn a shared initialization of model parameters across a variety of tasks during the meta-training phase that can adapt to new tasks using a few gradient steps. Cai et al. [3] proposes a simple weighted meta-learning approach only for the basis regression problem that selects weights by minimizing a data-dependent bound involving an empirical integral probability metric between the weighted sources and target risks. However, this approach cannot be easily extended to complex scenarios with arbitrary loss functions.

There are few meta-learning papers discussing learning with OOD tasks. Jeong and Kim [12] propose an OOD detection framework in meta-learning through generating fake samples which resemble in-distribution samples and combine them with real samples. However, they assume the outlier instances exist in the query set, which is different from ours. The most relevant field is from the perspective of task heterogeneity [31, 29, 34]. Vuorio et al. [31] proposed MMAML to deal with multimodal task distribution with disjoint and far apart modes and generates a set of separate meta-learned prior parameters to deal with each mode of a multimodal distribution. If we view that all the OOD tasks belong to a single mode, this is relevant to our setting. To tackle the distribution drift from meta-training to meta-test, B-TAML [14] learn to relocate the initial parameters to a new start point based on the arriving unseen tasks in the meta-test. The setting considered in our work and B-TAML work can be viewed as similar if we assume some of the datasets considered in the multi-dataset classification setting of B-TAML as OOD datasets.

To tackle samples with corrupted labels, some researches [16, 11, 32] introduce noise-robust models. Ren et al. [23] and Shu et al. [28] propose a noisy data filtering strategy using an instance reweighting strategy where the weights are learned automatically. However, the effect of noisy labels on few-shot learning requires more attention. Although Yin et al. [35], Lu et al. [15], Goldblum et al. [10] proposes robust meta-learning or few-shot learning, they assume a presence of outliers containing in meta-test, which is different from ours.

3 Preliminaries

3.1 Notations

In the setting of meta-learning for few-shot learning, there is a set of meta-training tasks $\{\mathcal{T}_i\}_{i=1}^M$ sampled from the probability distribution $p_{tr}(\mathcal{T})$. Each few-shot learning task $\mathcal{T}_i$ has an associated dataset $\mathcal{D}_i$ containing two disjoint sets $\{\mathcal{D}_i^S, \mathcal{D}_i^Q\}$, where the superscripts S and Q denote support set and query set respectively. The query sets take the form $\mathcal{D}_i^Q = \{\boldsymbol{x}_i^k, y_i^k\}_{k=1}^K$ and similarly for $\mathcal{D}_i^S$. Meta-validation tasks are denoted in a similar manner: $\{\mathcal{T}_j^{\mathcal{V}} = \{\mathcal{V}_j^S, \mathcal{V}_j^Q\}\}_{j=1}^N$ Let the loss function

be denoted as $\mathcal{L}(\boldsymbol{\phi}, \mathcal{D})$ with $\boldsymbol{\phi}$ denoting model parameters and $\mathcal{D}$ denoting the dataset, and $\ell(\boldsymbol{\theta}, d)$ with model parameters $\boldsymbol{\theta}$ on the data-point d. For example, $\mathcal{L}(\phi, \mathcal{D}_i^Q)$ denotes the loss of the i^{th} training task query set $\mathcal{D}_i^Q$ for given model parameters $\phi \in \boldsymbol{\Phi} \equiv \mathbb{R}^d$, where $\boldsymbol{\phi} := \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}^S)$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta} \equiv \mathbb{R}^d$ is the meta-parameter. $\mathcal{A}lg(\cdot)$ corresponds to a learning algorithm.

For notation convenience, we write $\mathcal{L}_i(\phi) := \mathcal{L}(\phi, \mathcal{D}_i^Q)$; $\mathcal{L}_{V_j}(\phi) := \mathcal{L}(\phi, \mathcal{V}_j^Q)$; $\widehat{\mathcal{L}}_{V_j}(\boldsymbol{\phi}) := \mathcal{L}(\boldsymbol{\phi}, \mathcal{V}_j^S)$. We denote scalars by lower case italic letters, vectors by lower case boldface letters, and matrices by capital italic letters throughout the paper. A table of notations with corresponding explanations is given in Appendix A.

3.2 Model-Agnostic Meta-Learning

The goal of MAML [7] is to obtain the optimal initial parameters that minimize the meta-training objective:

$$\overbrace{\boldsymbol{\theta}^*_{ML} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{F}(\boldsymbol{\theta})}^{outer-level} \qquad \text{where, } \mathcal{F}(\boldsymbol{\theta}) = \frac{1}{M} \sum\nolimits_{i=1}^{M} \mathcal{L}(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)}^{inner-level}, \mathcal{D}_i^Q) \tag{1}$$

This is a bi-level optimization problem, where we construe that $\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)$ explicitly or implicitly optimizes the inner-level task-specific adaptation. The outer-level corresponds to the meta-training objective of generalizing well (i.e. low test error) on the query set of each task after adaptation.

Since $\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)$ corresponds to single or multiple gradient descent steps. In case of a single gradient descent, $\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)$ can be perceived as follwing:

$$\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S) = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^S) \tag{2}$$

where α is a learning rate. As shown above, the meta-training objective assumes equal weights to each task for generalization, which may not be ideal in the case of adversaries in the training tasks.

4 Methodology

4.1 Problem Formulation

This section discusses a more generalized meta-learning framework, where we weigh all the data instances in the query set of a task. One of the significant purposes for considering weighted meta-learning is to make it more robust to adversaries during training.

In meta-learning, the support and query datasets $\{\mathcal{D}_i^S, \mathcal{D}_i^Q\}$ for each task $\mathcal{T}_i$ are usually sampled from an underlying dataset $\mathcal{D}$. In *instance-level weighting*, we associate each data instance $\{\mathcal{D}_{ik}^Q \mid k \in [K]\}$ in the query set of task $\mathcal{T}_i$ with a particular weight w_{ik}, where K is the number of datapoints (instances) in the query set $\mathcal{D}_i^Q$. The problem can be formulated as follows:

$$\boldsymbol{\theta}^*_{ML} = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{F}_w(\boldsymbol{\theta}) \tag{3}$$

$$\text{where } \mathcal{F}_w(\boldsymbol{\theta}) = \frac{1}{M} \sum\nolimits_{i=1}^{M} \sum\nolimits_{k=1}^{K} w_{ik} \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{ik}^Q) = \frac{1}{M} \sum\nolimits_{i=1}^{M} \mathbf{w}_i \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) \tag{4}$$

In the expression above,

$$\mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) = \left[\ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{i1}^Q), \ldots, \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{ik}^Q), \ldots, \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{iK}^Q)\right]^T$$

and $\mathbf{w}_i = [w_{i1}, \ldots, w_{iK}]$ is the weight vector corresponding to the query set of task $\mathcal{T}_i$. The *instance-level weighting* is useful in the scenarios where our underlying dataset $\mathcal{D}$ is prone to noisy labeled instances where an appropriate instance-level weighting can be used to distinguish the noisy samples with corrupted labels in the task. An ideal weight assignment is assigning large weight values to clean samples and small weight values to noisy samples in a task.

Likewise, we discuss a special case of the instance weighting scheme called *task-level weighting*, where we assign equal weights to every instance in the query set of a single task. *Task-level weighting* is applied in scenarios where every instance in a task's query set is from an OOD task distribution or an In-Distribution (ID) task. In this case, the optimal weight assignment assigns small weight values to an OOD task and large weight values to an ID task.

4.2 NESTED BI-LEVEL Optimization

Since we do not know the optimal weight assignment for real-world datasets, we need to learn the weights before training the *instance-level weighting* model using the bi-level optimization problem defined in Eq.(3).

NESTEDMAML solves for optimal weight assignments by posing them as hyper-parameters using the optimization problem defined in Eq.(5). As seen in the optimization equation, NESTEDMAML uses a clean held-out meta-validation task set $\{\mathcal{T}_j^{\mathcal{V}} = \{\mathcal{V}_j^S, \mathcal{V}_j^Q\}\}_{j=1}^N$ that is assumed to be relevant to test task distribution for generalization performance. In practice, the meta-validation task set's size is small compared to that of the meta-training tasks set ($N \ll M$). Hence, NESTEDMAML tries to select the weight hyper-parameters minimizing the model's meta-validation loss after taking a few gradient steps from the initial model parameters set using the instance-level weighting scheme.

The weight optimization objective for the instance-weighted MAML schema is as follows:

$$\begin{aligned} W^* &= \arg\min_{\mathbf{w}} \frac{1}{N} \sum\nolimits_{j=1}^{N} \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^*, \mathcal{V}_j^S), \mathcal{V}_j^Q) \\ \text{where } \boldsymbol{\theta}_W^* &= \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{M} \sum\nolimits_{i=1}^{M} \mathbf{w}_i^* \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_i^Q) \end{aligned} \tag{5}$$

and $W = [\mathbf{w}_1, \ldots, \mathbf{w}_M]^\intercal$. Since the optimization problem for $\boldsymbol{\theta}_W^*$ is a standard bi-level optimization problem (*i.e.* MAML), the complete optimization problem (Eq.(5)) turns out to be a **nested bi-level** optimization problem. It involves solving a standard bi-level optimization problem for every weight configuration, and hence naively solving this **nested bi-level** optimization problem is intractable. Hence, we adopt an online and one-step meta-gradient based approach to solve the optimization problem more efficiently.

4.3 The NESTEDMAML Algorithm

To reduce the optimization problem's (Eq.(5)) computation complexity, we solve the optimization problem in an iterative manner where we optimize the model parameters and weight hyperparameter by taking a single gradient step. This process is repeated until we reach convergence. Hence, we approximate the solution to the model parameters optimization in Eq.(5) first by adapting to each task using a single gradient step towards the inner task adaptation objective's descent direction and then taking a single gradient step towards the meta objective's descent direction.

Assuming that at every iterate t of training, a mini-batch of training tasks $\{\mathcal{T}_i \mid 1 \leq i \leq m\}$ is sampled, where m is the mini-batch size and $m \ll M$, the optimal model parameters update of the above problem is as follows:

$$\boldsymbol{\theta}_W^{(t)} = \boldsymbol{\theta}^{(t)} - \eta \frac{1}{m} \sum\nolimits_{i=1}^{m} \mathbf{w}_i^{(t)} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))|_{\boldsymbol{\theta}^{(t)}} \tag{6}$$

where η is meta objective's step-size and α is the inner objective's step-size. After this, the optimal weight optimization problem will be as follows:

$$W^* = \arg\min_{W} \frac{1}{N} \sum\nolimits_{j=1}^{N} \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S)) \tag{7}$$

Similarly, we optimize the weight hyperparameters by taking a single gradient step towards the meta-validation loss descent. We want to evaluate the impact of training a model on the weighted MAML objective against the meta-objective of sampled validation tasks $\{\mathcal{T}_j^V \mid 1 \leq j \leq n\}$ where, n is the mini-batch size and $n \ll N$. The weight update equation for the instance weighting scheme is as follows:

$$W^{(t+1)} = W^{(t)} - \frac{\gamma}{n} \sum\nolimits_{j=1}^{n} \nabla_W \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S)) \tag{8}$$

where γ is the weight update's step size. The Lemma below provides the gradient of the meta-validation loss $\frac{1}{n} \sum_{j=1}^{n} \nabla_W \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S))$ w.r.t. the weight vector $\mathbf{w}_i$, therefore giving the full update equation.

Lemma 1. *The weight update for an individual weight vector $\mathbf{w}_i$ of the task $\mathcal{T}_i$ from time step t to $t+1$ is as follows:*

$$\begin{aligned} \mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn} \sum\nolimits_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j} \Big(& \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \\ & - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j}|_{\boldsymbol{\theta}_W^{(t)}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \Big) \end{aligned} \tag{9}$$

where $\phi_j = \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S)$.

The proof is in Appendix B. Once the optimal weights $\mathbf{w}^{(t+1)}$ at $t+1$ are achieved, we train the model using the new weights:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \frac{\eta}{m} \sum\nolimits_{i=1}^{m} \mathbf{w}_i^{(t+1)} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}^{(t)}, \mathcal{D}_i^S)) \tag{10}$$

We repeat the steps given in the equation (6) from $t = 1$ until convergence. See Algorithm 1 for the full pseudo-code of NESTEDMAML.

First-Order Approximation (NESTEDMAML-FO). To show a faster way to solve the *nested bi-level* weight optimization problem, we use the approximated weight update takes the following form:

$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn} \sum\nolimits_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \tag{11}$$

This approximation is similar to the first-order approximation given in [7] where the second and higher-order terms are neglected. Details are shown in Appendix E.3.

Algorithm 1 NESTEDMAML

Require: p_{tr}, p_{val} distribution over training, validation tasks
Require: m, n (batch sizes) and α, η, γ (learning rates)
Randomly initialize $\boldsymbol{\theta}$ and W
while not done **do**
  Sample mini-batch of tasks $\{\mathcal{D}_i^S, \mathcal{D}_i^Q\}_{i=1}^m \sim p_{tr}$
  Sample mini-batch of tasks $\{\mathcal{V}_j^S, \mathcal{V}_j^Q\}_{j=1}^n \sim p_{val}$
  for each task $\mathcal{T}_i, \forall i \in [1, m]$ **do**
    Compute adapted parameters $\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)$ with gradient descent by Eq. (2)
    Compute the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))$ using $\mathcal{D}_i^Q$
    Formulate the $\boldsymbol{\theta}$ as a function of weights $\boldsymbol{\theta}_W^{(t)}$ by Eq. (6)
    Update $\mathbf{w}_i^{(t)}$ by Eq.(9) using $\{\mathcal{V}_j^S, \mathcal{V}_j^Q\}_{j=1}^n$
  end for
  Update $\boldsymbol{\theta}^{(t+1)}$ by Eq. (10) using $\{\mathcal{D}_i^Q\}_{i=1}^m$
end while

Weights Sharing. The number of weight hyper-parameters in the instance-level weighting scheme correlates to the number of data instances in the query sets of the meta-training tasks. We need to determine a significant amount of hyper-parameters if the number of training tasks or data instances is enormous, which in turn affects the hyper-parameter optimization algorithm, leading to instabilities during training. Accordingly, we seek to evaluate a smaller number of hyper-parameters by sharing the weights among instances. The task-weighting scheme is an occurrence of weight sharing where we share the same weight among all the instances in the query set. Apart from the task-level weighting scheme, we try to cluster tasks based on some similarity criteria to share the same weight among all the data instances in a cluster's query sets. We likewise present a sensitive analysis in the experiment section illustrating how the number of clusters in the training tasks or instances affects the NESTEDMAML algorithm's performance.

Convergence of NESTEDMAML Algorithm. In this work, we show that NESTEDMAML achieves a convergence rate of $\mathcal{O}(1/\epsilon^2)$ in the case of convex losses, as long as the inner learning rate is not too high. Detailed discussions are shown in Appendix C.

5 Experiments

In order to corroborate NESTEDMAML, we aim to study two questions: **Q1**: Can NESTEDMAML be successfully applied to problems where task distribution in the training domain is partially shifted from the task distribution in the testing domain? **Q2**: Instead of learning task weights, can NESTEDMAML deal with problems where data instances with noisy labels are used during the meta-training stage by learning weights in an instance-level scheme?

To answer these questions, we conduct the following experiments: (1) Mix OOD tasks with the meta-training tasks to evaluate the *task-level weighting scheme* of NESTEDMAML and (2) corrupt the labels of some training samples to evaluate the *instance-level weighting scheme* of NESTEDMAML. We follow the classification experiments in [7] to do few-shot learning to evaluate both the task-level and the instance-level weighting schemes. In addition, a synthetic regression experiment is conducted for the task-level weighting scheme as well. Due to the space limitation, we list synthetic regression experiments and *instance-level weighting scheme* for noisy labels experiments in Appendix E. We performed all the experiments using PyTorch, and the code is available at `https://github.com/Hugo101/NestedMAML`.

Table 1: Few-shot classification accuracies for the OOD experiment on various evaluation setups. ***mini*-Imagenet** is used as an in-distribution dataset ($\mathcal{D}_{in}$) for all experiments.

$\mathcal{D}_{out}$	5-way 3-shot					
	SVHN			**FashionMNIST**		
OOD Ratio	30%	60%	90%	30%	60%	90%
MAML-OOD-RM(Skyline)	57.73±0.76	55.29±0.78	54.38±0.12	56.78±0.75	55.29±0.78	53.43±0.51
MAML	55.41±0.75	53.93±0.76	44.10±0.68	54.65±0.77	54.52±0.76	41.52±0.74
MMAML	51.04±0.87	50.28±0.97	41.56±0.96	50.32±0.93	47.54±1.05	42.09±0.97
B-TAML	53.87±0.18	49.84±0.23	42.00±0.21	51.14±0.23	46.59±0.20	36.69±0.21
L2R	47.13±0.13	40.69±0.62	47.26±0.72	33.14±0.60	44.03±0.70	33.06±0.60
Transductive Fine-tuning	55.36±0.73	54.08±0.47	45.21±0.54	55.34±0.45	51.12±0.65	47.42±0.82
NESTEDMAML-FO(ours)	54.76±1.19	45.86±1.19	43.55±1.20	**57.00**±1.20	55.18±1.16	48.52±1.21
NESTEDMAML (ours)	**57.12**±0.81	**55.66**±0.78	**52.16**±0.76	56.66±0.78	**56.04**±0.79	**49.71**±0.78
$\mathcal{D}_{out}$	5-way 5-shot					
	SVHN			**FashionMNIST**		
OOD Ratio	30%	60%	90%	30%	60%	90%
MAML-OOD-RM(Skyline)	61.89±0.69	61.31±0.75	57.79±0.69	59.83±0.76	61.31±0.75	59.61±0.75
MAML	58.90±0.71	58.66±0.75	49.94±0.69	59.06±0.68	59.25±0.73	49.84±0.69
MMAML	52.45±1.00	52.17±1.05	46.51±1.09	51.46±0.91	54.13±0.93	50.27±1.00
B-TAML	58.34±0.20	56.07±0.21	49.84±0.20	55.19±0.20	52.10±0.19	40.02±0.19
L2R	47.11±0.51	48.01±0.70	51.53±0.71	46.03±0.30	49.15±0.68	55.03±0.46
Transductive Fine-tuning	59.16±0.76	57.84±0.58	53.64±0.42	56.54±0.87	56.23±0.70	54.28±0.32
NESTEDMAML-FO(ours)	57.96±0.94	53.66±0.95	47.58±0.96	**60.59**±0.99	**60.55**±0.95	49.23±0.98
NESTEDMAML (ours)	**60.76**±0.70	**60.53**±0.71	**57.88**±0.70	60.41±0.72	**60.54**±0.72	**57.95**±0.71

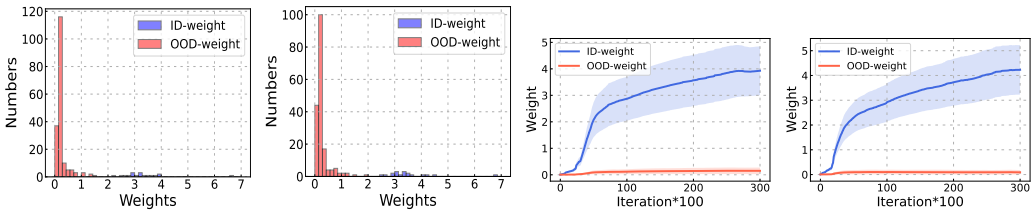
Datasets. We use *mini*-ImageNet [22], SVHN [18], FashionMNIST [33] datasets in our experiments. For the task-level weighting scheme, *mini*-ImageNet is considered as the ID tasks source ($\mathcal{D}_{in}$). Both the SVHN and the FashionMNIST datasets are used as OOD tasks source ($\mathcal{D}_{out}$) for *mini*-ImageNet. For instance-level weighting, *mini*-ImageNet is considered with corrupted labels. Additional details about datasets are given in Appendix E.4.

5.1 Task-level Weighting for OOD Tasks

Settings. We implement image classification experiments in 5-way, 3-shot (5-shot) settings. And we use a model with similar backbone architecture given in [30, 7] for all baselines. We consider a total of 20,000 training tasks containing both ID and OOD tasks where the split of ID and OOD tasks is determined by OOD ratio(0.3, 0.6, and 0.9 in this setting). At each iteration, ID tasks and OOD tasks will be sampled according to the OOD ratio. We sample the ID tasks (meta-training, meta-validation, and meta-test) from the *mini*-ImageNet dataset and sample OOD tasks from the SVHN or the FashionMNIST dataset. We process all images to be of size $84\times84\times3$. As mentioned before, in the task-level weighting, all the data instances in a task share the same weight, reducing the weight hyper-parameters count. To further reduce them, we use the K-means clustering method to cluster the tasks and assign a single weight value to all the same cluster tasks.

Baselines. In addition to **MAML**, we have **MAML-OOD-RM** which basically removes the OOD tasks during meta-training and hence is a skyline to our model. **MMAML** [31] leverages the strengths of meta-learners by identifying the mode of the task distribution and modulating the meta-learned prior in the parameter space. **B-TAML** [14] uses relocated initial parameters for new arriving tasks to handle OOD tasks. We adapted **L2R** [23] to assign weights for different tasks and optimize these weights through stochastic gradient descent. We consider **Transductive Fine-tuning** [5] as a baseline where we finetune the parameters of the model that is obtained by adding a new classifier on top of a pre-trained deep network, which is pre-trained on support and query sets of the meta-training set, using the meta-test set's support and unlabeled query set.

Results. Results in Table 1 show that NESTEDMAML significantly outperforms all baseline techniques and achieves performance competitive to the skyline method (MAML-OOD-RM) in the experiment of SVHN as OOD. For FashionMNIST OOD, NESTEDMAML still outperforms all baseline techniques for 60% and 90% ratio. For 30% ratio, the first-order approximation, NESTEDMAML-FO, has the best accuracy, and NESTEDMAML's accuracy is also comparable. Besides, the variance of NESTEDMAML is smaller than NESTEDMAML-FO, which means NESTEDMAML is



(a) 5-way 3-shot (b) 5-way 5-shot (c) 5-way 3-shot (d) 5-way 5-shot

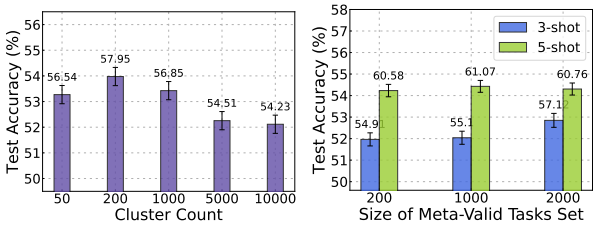
Figure 3: (a) and (b) show task weight distribution under 90% ratio (SVHN). (c) and (d) show weights trend as the iterations progress under 30% ratio (SVHN).

more stable than NESTEDMAML-FO and NESTEDMAML still has the best performance overall. From the perspective of training time, we observed that NESTEDMAML takes $\mathbf{1.7}\times$ and NESTEDMAML-FO takes $\mathbf{1.4}\times$ the time taken by MAML for training. Figure 3 (a)(b) show weight distribution for OOD and ID tasks under 90% ratio when SVHN is viewed as the OOD dataset for 5-way 3-shot (5-shot) settings after the meta-training phase. Both settings show that OOD tasks have much smaller weights than ID tasks: the weights belonging to OOD tasks approximately range from 0 to 1; however, the assigned weights for ID tasks are from 2 to 5, sometimes going up to 7.

To showcase the weights adaptation process during the training phase, we plot the weights trend as the iterations progress under the 30% OOD ratio (SVHN) in Figure 3(c)(d). The Blue (Red) curve denotes the mean weights for ID (OOD) tasks. The shade reflects the variance. Results show that the mean weight assigned to ID tasks would increase as the iterations progress, whereas the weights assigned to OOD tasks remain close to zero, which validates the effectiveness of the NESTEDMAML.

5.2 Sensitivity Analysis

We perform an ablation study to determine how the number of hyper-parameters and meta-validation sets' size can affect the NESTEDMAML algorithm's performance. To that extent, we evaluate the NESTEDMAML algorithm's performance using a different number of clusters in a 5-way 5-shot 90% FashionMNIST OOD setting. Figure (4a) shows test accuracies versus different numbers of clusters. We observed the best performance when the cluster count is 200. It is evident that the test accuracy decreases with an increase in the number of clusters that need to be determined. Contrarily, using a tiny number of clusters will also decrease the performance due to decreased clustering efficiency. We used 200 clusters for all our experiments. We also evaluate NESTEDMAML algorithm's performance using different sizes of the meta-validation set in 5-way 3-shot (5-shot) 30% SVHN OOD setting. Figure (4b) shows that NESTEDMAML algorithm performs well even when the meta-validation set size is tiny(i.e., 1% of meta-training set).



(a) FashionMNIST (90%) (b) SVHN (30%)

Figure 4: (a) shows accuracies under 90% FashionMNIST OOD level with different cluster values, (b) shows accuracies under 30% SVHN OOD level with different sizes of meta-validation tasks set.

6 Conclusion

We propose a novel robust meta-learning algorithm for reweighting tasks/instances of corrupted data in the meta-training phase. Our method is model-agnostic, can be directly applied to any deep learning architecture in an end-to-end manner. To the best of our knowledge, NESTEDMAML is the first algorithm to solve a *nested bi-level* optimization problem in an online manner with a convergence result. Finally, empirical evaluation results in OOD task and noisy label scenarios show that NESTEDMAML outperforms state-of-the-art meta-learning methods by efficiently mitigating the effects of unwanted instances or tasks.

References

[1] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433. PMLR, 2019.

[2] Harkirat Singh Behl, Atılım Güneş Baydin, and Philip HS Torr. Alpha maml: Adaptive model-agnostic meta-learning. *arXiv preprint arXiv:1905.07435*, 2019.

[3] Diana Cai, Rishit Sheth, Lester Mackey, and Nicolo Fusi. Weighted meta-learning. *arXiv preprint arXiv:2003.09465*, 2020.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *International Conference on Learning Representations (ICLR)*, 2019.

[5] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

[6] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR, 2020.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[8] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[9] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930, 2019.

[10] Micah Goldblum, Liam Fowl, and Tom Goldstein. Adversarially robust few-shot learning: A meta-learning approach. *Advances in Neural Information Processing Systems*, 33, 2020.

[11] Ajil Jalal, Andrew Ilyas, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

[12] Taewon Jeong and Heeyoung Kim. Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification. *Advances in Neural Information Processing Systems*, 33, 2020.

[13] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *arXiv preprint arXiv:2012.10630*, 2020.

[14] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *ICLR*, 2020.

[15] Jiang Lu, Sheng Jin, Jian Liang, and Changshui Zhang. Robust few-shot learning for user-provided data. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[16] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

[17] Devang K. Naik and R. Mammone. Meta-neural networks that learn by learning. *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 1:437–442 vol.1, 1992.

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[19] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[20] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[21] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[22] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2016.

[23] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

[24] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[25] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[26] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987. URL `http://www.idsia.ch/~juergen/diploma.html`.

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

[28] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

[29] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *ICLR*, 2020.

[30] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[31] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, pages 1–12, 2019.

[32] Huaxia Wang and Chun-Nam Yu. A direct approach to robust deep learning using adversarial networks. *arXiv preprint arXiv:1905.09591*, 2019.

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[34] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. *ICLR*, 2020.

[35] Chengxiang Yin, Jian Tang, Zhiyuan Xu, and Yanzhi Wang. Adversarial meta-learning. *arXiv preprint arXiv:1806.03316*, 2018.

[36] Chen Zhao, Changbin Li, Jincheng Li, and Feng Chen. Fair meta-learning for few-shot classification. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pages 275–282. IEEE, 2020.

[37] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations*, 2021.

Supplementary Material

A Notations

For clear interpretation, we list the notations used in this paper and their corresponding explanation, as shown in Table 2.

Notation	Description
$p_{tr}(\mathcal{T})$	probability distribution of meta-training tasks
$p_{val}(\mathcal{T})$	probability distribution of meta-validation tasks
M, N	the number of meta-training, meta-validation tasks, respectively
m, n	batch size for M, N, respectively
$\mathcal{T}_i$	i-th meta-training task
$\mathcal{T}_j^{\mathcal{V}}$	j-th meta-validation task
$\{\mathcal{D}_i^S, \mathcal{D}_i^Q\}$	support set and query set of meta-training task $\mathcal{T}_i$
$\{\mathcal{V}_j^S, \mathcal{V}_j^Q\}$	support set and query set of meta-validation task $\mathcal{T}_j^{\mathcal{V}}$
$\{\boldsymbol{x}_i^k, y_i^k\}_{k=1}^K$	K samples in the query set $\mathcal{D}_i^Q$ of meta-training task $\mathcal{T}_i$
$\boldsymbol{\theta}$	initial parameters of base learner
$\boldsymbol{\phi}_i$	task-specific parameters for task $\mathcal{T}_i$
$\boldsymbol{\theta}_W^*$	optimal initial parameters of base learner as a function of W
W	weight matrix for all query set samples of all meta-training tasks
$\mathbf{w}_i$	weight vector for query set samples of task $\mathcal{T}_i$
w_{ik}	weight for query sample k for task $\mathcal{T}_i$
W^*	optimal weights matrix
$\mathcal{L}(\boldsymbol{\phi}, \mathcal{D})$	loss function on dataset $\mathcal{D}$ characterized by model parameter $\boldsymbol{\phi}$
$\ell(\boldsymbol{\theta}, d)$	loss function on the query data point d characterized by model parameter $\boldsymbol{\theta}$
$\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D})$	one or multiple steps of gradient descent initialized at $\boldsymbol{\theta}$ on dataset $\mathcal{D}$
α, β, γ	step sizes

Table 2: Important Notations and Descriptions

- $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M]^\intercal$ is a matrix: $M \times K$
- $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{ik}, \ldots, w_{iK}]$ is a vector: $1 \times K$, weights for task $\mathcal{T}_i$

In addtion to the notations above, for notation convenience, we usually use the following notation simplicity:

$$\mathcal{L}_i(\phi) := \mathcal{L}(\phi, \mathcal{D}_i^Q), \widehat{\mathcal{L}}_i(\phi) := \mathcal{L}(\phi, \mathcal{D}_i^S)$$
$$\mathcal{L}_{V_j}(\phi) := \mathcal{L}(\phi, \mathcal{V}_j^Q), \widehat{\mathcal{L}}_{V_j}(\phi) := \mathcal{L}(\phi, \mathcal{V}_j^S)$$

B Weight Update of Instance-level and Task-level Weighting Scheme

B.1 Proof of Lemma 1

In this section, We restate the Lemma: 1 and present the detailed proof of Lemma: 1 below:

Lemma. *The weight update for an individual weight vector $\mathbf{w}_i$ of the task $\mathcal{T}_i$ from time step t to $t+1$ is as follows:*

$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn} \sum_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j} \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j}|_{\boldsymbol{\theta}_W^{(t)}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \right)$$

where $\phi_j = \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S)$.

Proof. Our goal is to find the optimal weights by using the set of meta-validation tasks. The weight optimization objective function is as follows:

$$W^* = \arg\min_{\mathbf{w}} \frac{1}{N}\sum_{j=1}^{N} \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S), \mathcal{V}_j^Q)$$

$$\text{where } \boldsymbol{\theta}^*_W = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{1}{M}\sum_{i=1}^{M} \mathbf{w}_i^* \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_i^Q)$$

Remark:

$$\mathbf{w}_i = [w_{i1}, \ldots, w_{iK}], \qquad \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_i^Q) = \begin{bmatrix} \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{i1}^Q) \\ \cdots \\ \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{ik}^Q) \\ \cdots \\ \ell(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{iK}^Q) \end{bmatrix}$$

Let

$$F(W, \boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{w}_i \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_i^Q) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{w}_i \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) \tag{12}$$

$$G(W, \boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N} \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S), \mathcal{V}_j^Q) = \frac{1}{N}\sum_{j=1}^{N} \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S)) \tag{13}$$

We only consider one-step gradient update:

$$\widehat{W} = W - \gamma \frac{\partial G(W, \boldsymbol{\theta})}{\partial W} = W - \gamma \frac{1}{N}\sum_{j=1}^{N} \nabla_W \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S)) \tag{14}$$

$$\widehat{\boldsymbol{\theta}}_W = \boldsymbol{\theta} - \eta \frac{\partial F(W, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta} - \eta \frac{1}{M}\sum_{i=1}^{M} \mathbf{w}_i \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) \tag{15}$$

$$\begin{aligned}
\nabla_{\mathbf{w}_i} \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}^*_{\mathbf{w}}, \mathcal{V}_j^S)) &= \frac{\partial \mathcal{L}_{V_j}}{\partial \mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S)} \frac{\partial \mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S)}{\partial \boldsymbol{\theta}_W} \frac{d\boldsymbol{\theta}_W}{d\mathbf{w}_i} \\
&= \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \frac{\partial(\boldsymbol{\theta}^*_W - \alpha \nabla \widehat{\mathcal{L}}_{V_j}(\boldsymbol{\theta}_W))}{\partial \boldsymbol{\theta}_W} (-\eta \frac{1}{M}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \\
&= \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \cdot \left(I - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j}\right) (-\eta \frac{1}{M}) \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \\
&= (-\frac{\eta}{M}) \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \cdot \left(I - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j}\right) \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \\
&= -\frac{\eta}{M} \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \cdot \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j} \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal\right)
\end{aligned} \tag{16}$$

Thus the weight update for task $\mathcal{T}_i$ can be:

$$\begin{aligned}
\mathbf{w}_i^{(t+1)} &= \mathbf{w}_i^{(t)} - \gamma \frac{1}{n}\sum_{j=1}^{n} \nabla_{\mathbf{w}_i} \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}^*_W, \mathcal{V}_j^S)) \\
&= \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn}\sum_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal - \alpha \nabla^2 \widehat{\mathcal{L}}_{V_j}|_{\boldsymbol{\theta}_W^{(t)}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal\right)
\end{aligned} \tag{17}$$

$$\approx \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn}\sum_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j}(\phi_j) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal \tag{18}$$

Eq. (17) is the exact update, namely, lemma 1. Eq. (18) is the approximation update. □

B.2 Weight Update in Task-level Weighting Scheme

As mentioned in the paper, task weighting scheme is a special case of instance weighting scheme. In instance weighting scheme, each task $\mathcal{T}_i$ has a vector weight $\mathbf{w}_i$. In task weighting scheme, all query samples have the same weight. In other words, each task $\mathcal{T}_i$ has a scalar weight w_i. And the loss of training tasks used for the update of $\boldsymbol{\theta}$ would be the average loss for all query samples in task $\mathcal{T}_i$:

$$\mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) = \frac{1}{K}\sum_{k=1}^{K} \ell(\mathcal{A}lg((\boldsymbol{\theta}, \mathcal{D}_i^S), \mathcal{D}_{ik}^Q)$$

The weight update follows the same strategy in Lemma 1.

C Convergence of NESTEDMAML Algorithm

Table 3: Convergence Rates of MAML and NESTEDMAML

Algorithm	Strongly Convex Loss	Non-Convex Loss
MAML	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$
NESTEDMAML	$\mathcal{O}(1/\epsilon^2)$	Open

Although the MAML algorithm's convergence rate is studied [1, 6, 9], those results do not directly hold in our case since we have a *nested bi-level* optimization objective instead of standard bi-level objective of the MAML. Recall that in the case of strongly convex losses, MAML admits a convergence rate of $\mathcal{O}(1/\epsilon)$ [1, 9]. In contrast, for the non-convex case, [6] show a weaker convergence rate of $\mathcal{O}(1/\epsilon^2)$ to a first order stationary point. In this work, we show that NESTEDMAML achieves a convergence rate of $\mathcal{O}(1/\epsilon^2)$ in the case of convex losses, as long as the inner learning rate is not too high. Furthermore, we show that NESTEDMAML converges to a critical point of meta-validation loss and not the meta-training loss since we are optimizing the meta-validation loss in the nested bi-level setting. Table 3 shows the convergence rates of MAML and NESTEDMAML algorithms for strongly convex and non-convex loss functions.

Theorem 1. *Suppose the loss function $\mathcal{L}(\cdot)$ is Lipschitz smooth with constant L, μ-strongly convex, and is a twice differential function with a ρ-bounded gradient and $\mathcal{B}$-Lipschitz Hessian. Denote σ as the variance of drawing uniformly mini-batch sample at random. Assume that the learning rate η_t satisfies $\eta_t = \min(1, k/T)$ for some $k > 0$ such that $k/T < 1$ and γ_t, $1 \leq t \leq T$, is a monotone descent sequence. Let $\gamma_t = \min(\frac{1}{L}, \frac{C}{\sigma\sqrt{T}})$ for some $C > 0$ such that $\frac{\sigma\sqrt{T}}{C} \geq L$ and $\sum_{t=0}^{\infty}\gamma_t \leq \infty$, $\sum_{t=0}^{\infty}\gamma_t^2 \leq \infty$. Then,* NESTEDMAML *satisfies:* $\mathbb{E}\left[\left\|\frac{1}{N}\sum_{j=1}^{N}\nabla_W \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q)\right\|^2\right] \leq \epsilon$ *in* $\mathcal{O}(\frac{1}{\epsilon^2})$ *steps. More specifically,*

$$\min_{0\leq t\leq T}\mathbb{E}\left[\left\|\frac{1}{N}\sum\nolimits_{j=1}^{N}\nabla_W \mathcal{L}(Alg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q)\right\|^2\right] \leq \mathcal{O}(\frac{1}{\sqrt{T}})$$

Proof is given in Appendix D. The difference in convergence rates between MAML and NESTEDMAML is due to the additional complexity involved in solving a *nested bi-level* optimization problem. The convergence analysis of NESTEDMAML for non-convex functions is challenging and currently unknown. Even though most deep learning problems have a non-convex landscape, the algorithms initially developed for convex cases have shown promising empirical results in non-convex cases. Under this assumption, we provide an implementation that can be generalized to any deep learning architecture in Algorithm 1.

D Detailed Convergence Analysis

In this section, we present the detailed proof of convergence. Before that, we first give two assumptions and several lemmas which could help for the proof of convergence.

The meta validation loss is as follows:

$$\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)}) = \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S))$$

Assuming that the whole weight matrix W is flattened to a column matrix, the weighted meta-training loss can be written as follows:

$$\begin{aligned} & \mathcal{L}_W^{meta}(\boldsymbol{\theta}, W) = W^\intercal \mathcal{L}_T^{meta}(\boldsymbol{\theta}) \\ \text{where} \quad & \mathcal{L}_T^{meta}(\boldsymbol{\theta}) = \frac{1}{m}[\mathcal{L}_1(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_1^S)) \dots \mathcal{L}_m(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_m^S))]^\intercal \end{aligned}$$

The weight update equation at time step t can be written as follows:

$$\begin{aligned} & W^{(t+1)} = W^{(t)} - \gamma \nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)}) \\ \text{where} \quad & \boldsymbol{\theta}_W^{(t)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)}) \end{aligned} \tag{19}$$

Assumption 1. (C^2-smoothness) *Suppose that $\mathcal{L}(\cdot)$:*

- *is twice differentiable*
- *is ρ-Lipschitz in function value, i.e., $\|\nabla\mathcal{L}(\boldsymbol{\theta})\| \leq \rho$*
- *is L-smooth, or has L-Lipschitz gradients, i.e., $\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\phi})\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\phi}\| \; \forall \boldsymbol{\theta}, \boldsymbol{\phi}$*
- *has $\mathcal{B}$-Lipschitz hessian, i.e., $\left\|\nabla^2\mathcal{L}(\boldsymbol{\theta}) - \nabla^2\mathcal{L}(\boldsymbol{\phi})\right\| \leq \mathcal{B}\|\boldsymbol{\theta} - \boldsymbol{\phi}\| \; \forall \boldsymbol{\theta}, \boldsymbol{\phi}$*

Assumption 2. (Strong convexity) *Suppose that $\mathcal{L}(\cdot)$ is convex. Further, μ-strongly convex. i.e., $\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\boldsymbol{\phi})\| \geq \mu\|\boldsymbol{\theta} - \boldsymbol{\phi}\| \; \forall \boldsymbol{\theta}, \boldsymbol{\phi}$*

Lemma 2. [9] *Suppose $\mathcal{L}$ and $\hat{\mathcal{L}} : \mathbb{R}^d \longrightarrow \mathbb{R}$ satisfy assumptions 1 and 2. Let $\tilde{\mathcal{L}}$ be the function evaluated after a one step gradient update procedure, i.e.*

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) := \mathcal{L}(\boldsymbol{\theta} - \alpha\nabla\hat{\mathcal{L}}(\boldsymbol{\theta}))$$

If the step size is selected as $\alpha \leq \min\{\frac{1}{2L}, \frac{\mu}{8\rho\mathcal{B}}\}$, then $\tilde{\mathcal{L}}$ is convex. Furthermore, it is also $\tilde{L} = 9L/8$ smooth and $\tilde{\mu} = \mu/8$ strongly convex.

Lemma 3. *Suppose the loss function $\mathcal{L}$ is Lipschitz smooth with constant L, then the meta-validation loss $\mathcal{L}_V^{meta}$ is Lipschitz smooth with constant $\frac{9L}{8}$.*

Proof. Since we know that,

$$\begin{aligned} \mathcal{L}_V^{meta}(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S)) \\ &= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S), \mathcal{V}_j^Q) \end{aligned} \tag{20}$$

From Lemma: 2, we can say that $\forall j \in [1, n]$, $\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S), \mathcal{V}_j^Q)$ is also lipschitz smooth with a constant of $\frac{9L}{8}$.

$$\begin{aligned}
\left\|\nabla\mathcal{L}_V^{meta}(\boldsymbol{\theta}) - \nabla\mathcal{L}_V^{meta}(\boldsymbol{\phi})\right\| &= \left\|\frac{1}{n}\sum_{j=1}^{n}\left(\nabla\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S),\mathcal{V}_j^Q) - \nabla\mathcal{L}(\mathcal{A}lg(\boldsymbol{\phi},\mathcal{V}_j^S),\mathcal{V}_j^Q)\right)\right\| \\
&\leq \frac{1}{n}\sum_{j=1}^{n}\left\|\left(\nabla\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S),\mathcal{V}_j^Q) - \nabla\mathcal{L}(\mathcal{A}lg(\boldsymbol{\phi},\mathcal{V}_j^S),\mathcal{V}_j^Q)\right)\right\| \\
&\leq \frac{1}{n}\sum_{j=1}^{n}\frac{9L}{8}\left\|\boldsymbol{\theta}-\boldsymbol{\phi}\right\| \\
&= \frac{9L}{8}\left\|\boldsymbol{\theta}-\boldsymbol{\phi}\right\| \qquad (21)
\end{aligned}$$

Therefore the meta-validation loss function $\mathcal{L}_V^{meta}$ is also lipschitz smooth with constant $\frac{9L}{8}$. □

Lemma 4. *Suppose the loss function $\mathcal{L}$ satisfies assumption 1 and 2, then the query set loss $\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S))$ and $\mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{D}_i^S))$ are $\rho(1+\alpha L)$-gradient bounded functions.*

Proof. Since we know that,

$$\begin{aligned}
\mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{D}_i^S)) &= \mathcal{L}_i(\boldsymbol{\theta}-\alpha\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S))) \\
&= \mathcal{L}(\boldsymbol{\theta}-\alpha\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S),\mathcal{D}_i^Q) \qquad (22)
\end{aligned}$$

Suppose:

$$\boldsymbol{\phi} = \boldsymbol{\theta} - \alpha\nabla\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S)$$

$$\begin{aligned}
\left\|\nabla\mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{D}_i^S))\right\| &= \left\|\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{\theta}}\frac{\partial\mathcal{L}(\boldsymbol{\phi},\mathcal{D}_i^Q)}{\partial\boldsymbol{\phi}}\right\| \\
&= \left\|(1-\alpha\nabla^2\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S))\nabla\mathcal{L}(\phi,\mathcal{D}_i^Q)\right\| \\
&\leq \left\|(1-\alpha\nabla^2\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S))\right\|\left\|\nabla\mathcal{L}(\phi,\mathcal{D}_i^Q)\right\| \\
&\leq (1+\left\|(\alpha\nabla^2\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_i^S))\right\|)\left\|\nabla\mathcal{L}(\phi,\mathcal{D}_i^Q)\right\| \\
&\leq \rho(1+\alpha L) \qquad (23)
\end{aligned}$$

Similarly for $\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S))$. □

Lemma 5. *Suppose the loss function $\mathcal{L}$ is ρ-gradient bounded, then the meta-validation loss $\mathcal{L}_V^{meta}$, the meta-training loss $\mathcal{L}_T^{meta}$ and the weighted meta-training loss $\mathcal{L}_W^{meta}$ are $\rho(1+\alpha L)$-gradient bounded functions.*

Proof. Since we know that,

$$\begin{aligned}
\mathcal{L}_V^{meta}(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S)) \\
&= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta},\mathcal{V}_j^S),\mathcal{V}_j^Q)
\end{aligned}$$

$$
\begin{aligned}
\left\|\nabla \mathcal{L}_V^{meta}(\boldsymbol{\theta})\right\| &= \left\| \frac{1}{n} \sum_{j=1}^{n} \nabla \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S), \mathcal{V}_j^Q) \right\| \\
&\leq \frac{1}{n} \sum_{j=1}^{n} \left\| \nabla \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{V}_j^S), \mathcal{V}_j^Q) \right\| \\
&\leq \frac{1}{n} \sum_{j=1}^{n} \rho(1+\alpha L) \quad \text{(From Lemma: 4)} \\
&= \rho(1+\alpha L) \qquad (24)
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\mathcal{L}_T^{meta}(\boldsymbol{\theta}) &= \frac{1}{m}[\mathcal{L}_1(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_1^S)), \ldots, \mathcal{L}_m(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_m^S))]^\intercal \\
&= \frac{1}{m}[\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_1^S), \mathcal{D}_1^S), \ldots, \mathcal{L}_m(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_m^S), \mathcal{D}_m^Q)]^\intercal \qquad (25)
\end{aligned}
$$

$$
\begin{aligned}
\left\|\nabla \mathcal{L}_T^{meta}(\boldsymbol{\theta})\right\| &= \left\| \frac{1}{m} \nabla [\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_1^S), \mathcal{D}_1^S), \ldots, \mathcal{L}_m(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_m^S), \mathcal{D}_m^Q)]^\intercal \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \left\| \nabla \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_j^S), \mathcal{D}_j^Q) \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \rho(1+\alpha L) \quad \text{(From Lemma: 4)} \\
&= \rho(1+\alpha L) \qquad (26)
\end{aligned}
$$

The weighted meta-training loss is as follows:

$$
\begin{aligned}
\mathcal{L}_W^{meta}(\boldsymbol{\theta}) &= [\mathbf{w}_1 \ldots \mathbf{w}_m] \cdot \frac{1}{m}[\mathcal{L}_1(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_1^S)) \ldots \mathcal{L}_m(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_m^S))]^\intercal \\
&= \frac{1}{m} \sum_{i=1}^{m} \mathbf{w}_i^T \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) \qquad (27)
\end{aligned}
$$

Since, $\mathbf{w}_i$ weight vector is normalized at every iteration such that $\|\mathbf{w}_i\| = 1$, we have :

$$
\begin{aligned}
\left\|\nabla \mathcal{L}_W^{meta}(\boldsymbol{\theta})\right\| &= \left\| \frac{1}{m} \sum_{i=1}^{m} \mathbf{w}_i^T \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S)) \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \left\| \mathbf{w}_i^T \nabla \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_j^S)) \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \|\mathbf{w}_i\| \left\| \nabla \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_j^S)) \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \left\| \nabla L_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_j^S)) \right\| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \rho(1+\alpha L) \quad \text{(From Lemma: 4)} \\
&= \rho(1+\alpha L) \qquad (28)
\end{aligned}
$$

Therefore the meta-validation loss function $\mathcal{L}_V^{meta}$, meta-training loss function $\mathcal{L}_T^{meta}$ and weighted meta-training loss $\mathcal{L}_W^{meta}$are $\rho(1+\alpha L)$-gradient bounded functions. □

Lemma 6. *Suppose the meta validation loss function $\mathcal{L}_V^{meta}$ is Lipschitz smooth with constant L, and the meta training loss function $\mathcal{L}_T^{meta}$ have ρ-bounded gradients with respect to training/validation data. Then the gradient of meta validation loss with respect to W is Lipschitz continuous.*

Proof. For meta approximation method, then the gradient of meta-validation loss with respect to W can be written as follows:

$$\begin{aligned}\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W) &= \frac{\partial \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W} \cdot \frac{\partial \boldsymbol{\theta}_W}{\partial W} \\ &= \frac{\partial \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W} \cdot \frac{\partial \left(\boldsymbol{\theta} - \eta \frac{\partial \mathcal{L}_W^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)}{\partial W} \\ &= -\eta \frac{\partial \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W} \cdot \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\end{aligned} \tag{29}$$

Taking gradient with respect to W on both sides of Eq (29), we have:

$$\begin{aligned}\|\nabla_W^2 \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)\| &= \eta \Big\| \frac{\partial}{\partial W}\Big(\frac{\partial \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W} \cdot \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big)\Big\| \\ &= \eta \Big\| \frac{\partial \boldsymbol{\theta}_W}{\partial W}\Big(\frac{\partial^2 \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W \partial \boldsymbol{\theta}_W}\Big) \cdot \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big\| \\ &= \eta \Big\| -\eta \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big(\frac{\partial^2 \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W \partial \boldsymbol{\theta}_W}\Big) \cdot \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big\| \\ &= \eta^2 \Big\| \frac{\partial^2 \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W \partial \boldsymbol{\theta}_W} \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big\| \\ &\leq \frac{9L\eta^2\rho^2(1+\alpha L)^2}{8} \quad \text{(From Lemma: 3 and Lemma: 5)}\end{aligned} \tag{30}$$

Since $\|\frac{\partial^2 \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W)}{\partial \boldsymbol{\theta}_W \partial \boldsymbol{\theta}_W}\| \leq \frac{9L}{8}, \|\frac{\mathcal{L}_T^{meta}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\| \leq \rho(1+\alpha L)$. Define $\tilde{L} = \frac{9\eta^2\rho^2(1+\alpha L)^2 L}{8}$, based on Lagrange mean value theorem, we have,

$$\|\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_{W_i}) - \nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_{W_j})\| \leq \tilde{L}\|W_i - W_j\|, \quad \text{for all } W_i, W_j \tag{31}$$

where $\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_{W_i}^{(t)}) = \frac{1}{n}\sum_{j=1}^n \nabla_W \mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W_i), \mathcal{V}_j^S))$ □

We restate the Theorem: 1 and present the detailed proof of Theorem: 1 below:

Theorem. *Suppose the loss function $\mathcal{L}$ is Lipschitz smooth with constant L and is a differential function with a ρ-bounded gradient, twice differential and $\mathcal{B}$-lipschitz hessian. Assume that the learning rate η_t satisfies $\eta_t = \min(1, k/T)$ for some $k > 0$, such that $k/T < 1$ and γ_t, $1 \leq t \leq T$ is a monotone descent sequence, $\gamma_t = \min(\frac{1}{L}, \frac{C}{\sigma\sqrt{T}})$ for some $C > 0$, such that $\frac{\sigma\sqrt{T}}{C} \geq L$ and $\sum_{t=0}^{\infty} \gamma_t \leq \infty$, $\sum_{t=0}^{\infty} \gamma_t^2 \leq \infty$. Then* NESTEDMAML *satisfies: $\mathbb{E}\left[\left\|\frac{1}{N}\sum_{j=1}^N \nabla_W \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q)\right\|^2\right] \leq \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps. More specifically,*

$$\min_{0\leq t\leq T} \mathbb{E}\left[\left\|\frac{1}{N}\sum_{j=1}^N \nabla_W \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q)\right\|^2\right] \leq \mathcal{O}(\frac{1}{\sqrt{T}}) \tag{32}$$

where C is some constant independent of the convergence process, σ is the variance of drawing uniformly mini-batch sample at random.

Proof. We rewrite the weight update equation at time step t (Eq. 19) as follows:

$$\begin{aligned} & W^{(t+1)} = W^{(t)} - \gamma \nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)}) \\ \text{where} \quad & \boldsymbol{\theta}_W^{(t)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)}) \end{aligned}$$

Based on the update equations we can write,

$$\begin{aligned}\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t+1)}) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t)}) &= \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t-1)}, W^{(t-1)})) \\ &= \Big(\underbrace{\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)}))}_{\text{(a)}}\Big) + \\ &\quad \Big(\underbrace{\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t-1)}, W^{(t-1)}))}_{\text{(b)}}\Big)\end{aligned}$$

From lemma 3, the functions $\mathcal{L}_V^{meta}(\boldsymbol{\theta})$ and $\mathcal{L}_W^{meta}(\boldsymbol{\theta})$ are lipschitz smooth with lipschitz constant L provided the loss function $\mathcal{L}(\theta)$ is lipschitz smooth with lipschitz constant L.

For term(a) using the lipschitz smoothness property, we have:

$$\begin{aligned}&\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) \\ &\leq (\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)})^T(\nabla_{\boldsymbol{\theta}}\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)}))) + \frac{L}{2}\left\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)}\right\|^2 \end{aligned} \tag{33}$$

Since, $\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)} = -\eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W(\boldsymbol{\theta}^{(t-1)}, W^{(t)})$, $\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}, W) \leq \rho(1+\alpha L)$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_V^{meta}(\boldsymbol{\theta}) \leq \rho(1+\alpha L)$. We have:

$$\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) \tag{34}$$

$$\leq (-\eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W(\boldsymbol{\theta}^{(t-1)}, W^t))^T(\nabla_{\boldsymbol{\theta}}\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)}))) + \frac{L}{2}\left\|-\eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W(\boldsymbol{\theta}^{(t-1)}, W^{(t)})\right\|^2 \tag{35}$$

$$\leq \eta\rho^2(1+\alpha L)^2(1+\frac{\eta L}{2}) \tag{36}$$

For term(b) using the lipschitz smoothness property, we have:

$$\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t)}, W^{(t)})) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(t-1)} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}_W^{meta}(\boldsymbol{\theta}^{(t-1)}, W^{(t-1)})) \tag{37}$$

$$= \mathcal{L}_V^{meta}(\theta_W^{(t)}) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)}) \tag{38}$$

$$\leq (W^{(t)} - W^{(t-1)})^T\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)}) + \frac{\tilde{L}}{2}\left\|W^{(t)} - W^{(t-1)}\right\|^2 \quad \text{(From Lemma: 6)}$$

$$= -\gamma\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)})^T\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)}) + \frac{\tilde{L}}{2}\left\|-\gamma\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)})\right\|^2 \tag{39}$$

$$= (\frac{\tilde{L}\gamma^2}{2} - \gamma)\left\|\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)})\right\|^2 \tag{40}$$

Combining both the inequalities for form(a) and form(b), we have:

$$\mathcal{L}_V^{meta}(\boldsymbol{\theta}^{t+1}) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^t) \leq \eta\rho^2(1+\alpha L)^2(1+\frac{\eta L}{2}) + (\frac{\tilde{L}\gamma^2}{2} - \gamma)\left\|\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t-1)})\right\|^2 \tag{41}$$

Summing up the above inequality from $t = 1$ to $t = T-1$ and rearranging the terms, we can obtain

$$\begin{aligned}\sum_{t=1}^{T-1}(\gamma - \frac{\tilde{L}\gamma^2}{2})\|\nabla_W\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)})\|_2^2 &\leq \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(1)}) - \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(T)}) + \eta\rho^2(1+\alpha L)^2(\frac{\eta L(T-1)}{2} + T - 1) \\ &\leq \mathcal{L}_V^{meta}(\boldsymbol{\theta}^{(1)}) + \eta\rho^2(1+\alpha L)^2(\frac{\eta LT}{2} + T)\end{aligned} \tag{42}$$

Furthermore, we can deduce that,

$$
\begin{aligned}
\min_t \mathbb{E}[\|\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)})\|_2^2] &\leq \frac{\sum_{t=1}^{T-1}(\gamma - \frac{\tilde{L}\gamma^2}{2})\|\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)})\|_2^2}{\sum_{t=1}^{T}(\gamma - \frac{\tilde{L}}{2}\gamma^2)} \\
&= \frac{\sum_{t=1}^{T-1}(\gamma - \frac{\tilde{L}}{2}\gamma^2)\|\nabla_W \mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)})\|_2^2}{\sum_{t=1}^{T-1}(\gamma - \frac{\tilde{L}}{2}\gamma^2)} \\
&\leq \frac{\left[2\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(1)}) + \eta\rho^2(1+\alpha L)^2(\eta LT + 2T)\right]}{\sum_{t=1}^{T}(2\gamma - \tilde{L}\gamma^2)} \\
&\leq \frac{2\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(1)}) + \eta\rho^2(1+\alpha L)^2(\eta LT + 2T)}{\sum_{t=1}^{T}\gamma} \\
&\leq \frac{2\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(1)})}{\gamma T} + \frac{\eta\rho^2(1+\alpha L)^2(L+2)}{\gamma} \quad (\eta = \min\{1, \frac{k}{T}\} \text{ and } \eta \leq 1) \\
&= \frac{2\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(1)})}{T}\max\{L, \frac{\sqrt{T}}{C}\} + \min\{1, \frac{k}{T}\}\max\{L, \frac{\sqrt{T}}{C}\}\rho^2(1+\alpha L)^2(L+2) \\
&= \frac{2\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(1)})}{C\sqrt{T}} + \frac{k\rho^2(1+\alpha L)^2(L+2)}{C\sqrt{T}} = \mathcal{O}(\frac{1}{\sqrt{T}}) \qquad (43)
\end{aligned}
$$

The third inequality holds for $\sum_{t=1}^{T}\gamma \leq \sum_{t=1}^{T}(2\gamma - \tilde{L}\gamma^2)$ which made us choose a functional form of γ to be $\gamma_t = \min\left(\frac{1}{L}, \frac{C}{\sigma\sqrt{T}}\right)$.

We know that,

$$
\begin{aligned}
\mathcal{L}_V^{meta}(\boldsymbol{\theta}_W^{(t)}) &= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_{V_j}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S)) \qquad (44) \\
&= \frac{1}{n}\sum_{j=1}^{n}\mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q) \qquad (45)
\end{aligned}
$$

Therefore, we can conclude that our algorithm achieves $\min_{0\leq t\leq T}\mathbb{E}[\|\frac{1}{n}\sum_{j=1}^{n}\nabla_W \mathcal{L}(\mathcal{A}lg(\boldsymbol{\theta}_W^{(t)}, \mathcal{V}_j^S), \mathcal{V}_j^Q)\|_2^2] \leq \mathcal{O}(\frac{1}{\sqrt{T}})$ in T steps. □

E Additional Experiments

E.1 Synthetic Regression

Regression Setting. To show our proposed model's robustness in the OOD task scenario, we start with a simple regression problem with outliers in the **synthetic dataset**. Specifically, during the meta-training time, each task involves K samples as input and a sine wave as output, where the amplitude and phase of each sine wave are varied between tasks. More concretely, the amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$. Datapoints from sine waves are sampled uniformly from $[-5.0, 5.0]$. In addition to in-distribution data (*i.e.* data points sampled from sine waves), outliers or data points out of sine distributions (*i.e.* OOD) are added into meta-training stage. To generate OOD data, we set outputs that are linear to the corresponding inputs. It is notable that, during meta-val and meta-test stages, all tasks are without any outliers. Our proposed model's intuition behind such a setting learns weights based on validation tasks and will assign higher weights to sinusoid tasks in meta-training, which could have better results. Instead, MAML uses equal weights for each meta-training task, which may not generalize good performance to unseen tasks when OOD is mixed during training. The loss function of Mean Squared Error (MSE) between prediction and the true value is applied for optimization. During meta-validation/test time, all tasks are without any outliers. Intuition: our model could learn weights based on validation tasks (sine wave) and assign higher weights to sinusoid tasks in meta-training tasks, resulting in better results. Instead, MAML uses the same weights for each task in meta-training tasks, which will not have good generalization results.

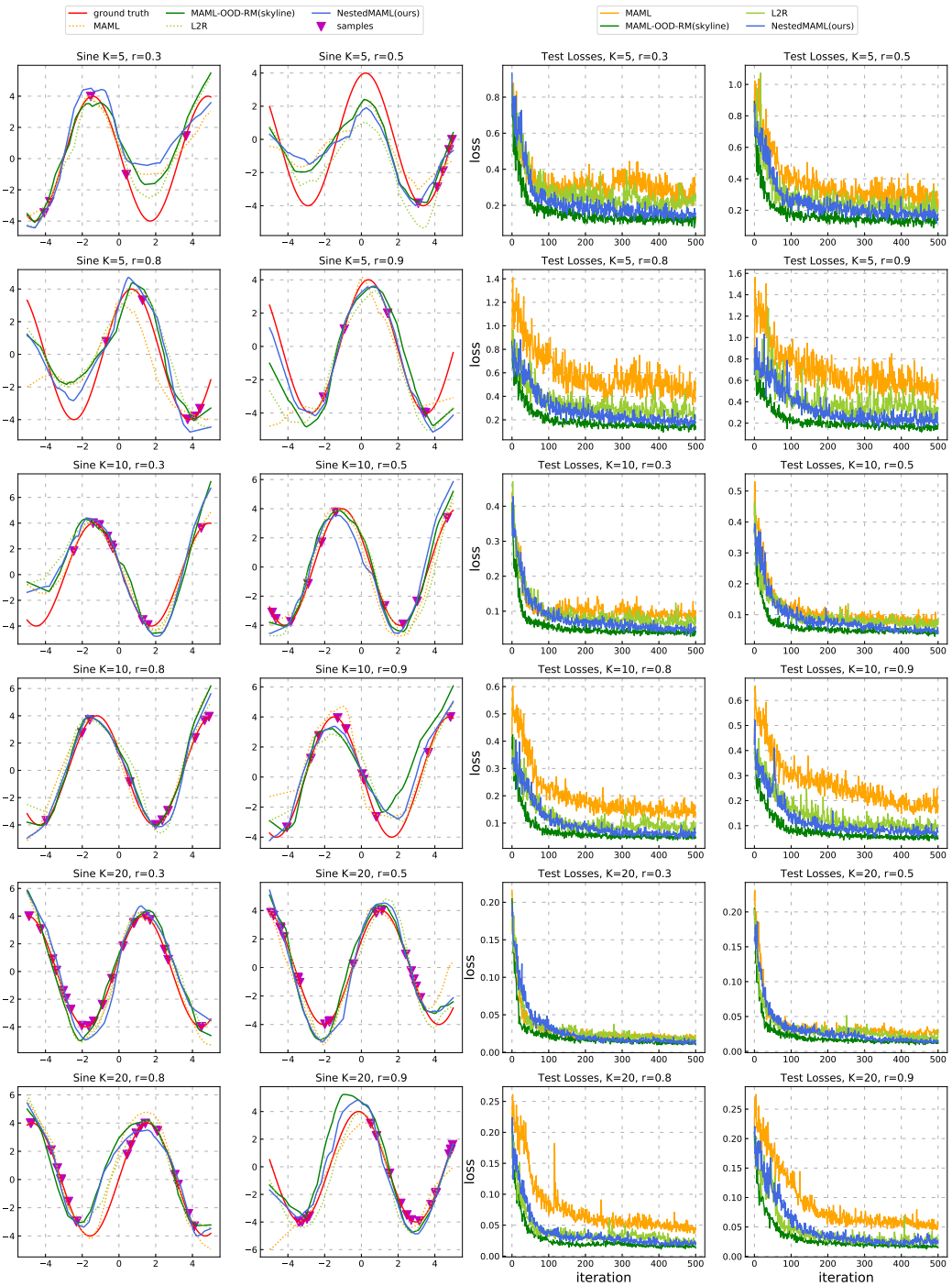


Figure 5: Results of few-shot (K=5, 10, 20) for the simple sinusoid regression task including the loss curves with respect to the number of iterations. Plotted by different levels of OOD Tasks (r = 0.3, 0.5, 0.8, 0.9).

Figure 5 shows the results of the NESTEDMAML and other baselines: **MAML** [7] and **L2R** [23]. The baseline **MAML-OOD-RM** corresponds to a MAML model trained just on In-Distribution (ID) tasks and will act as skyline. Table 4 shows the MSE loss for the OOD experiment on various evaluation setups. From Figure 5 and Table 4, it is evident that NESTEDMAML algorithm performs better than other baseline methods and achieved low MSE error values.

Table 4: MSE loss for the OOD experiment on various evaluation setups. **sinusoid** is used as an in-distribution dataset ($\mathcal{D}_{in}$) for all experiments.

Shots K	Methods	r=0.3	r=0.5	r=0.8	r=0.9
5	MAML-OOD-RM(skyline)	0.1357	0.1460	0.1457	0.1830
	MAML	0.2448	0.2658	0.5200	0.5807
	L2R	0.2228	0.2225	0.2137	0.3361
	NESTEDMAML (ours)	**0.1548**	**0.1725**	**0.1761**	**0.1971**
10	MAML-OOD-RM(skyline)	0.0430	0.0425	0.0466	0.0485
	MAML	0.1015	0.0865	0.1397	0.1831
	L2R	0.0723	0.0888	0.1022	0.0978
	NESTEDMAML (ours)	**0.0552**	**0.0458**	**0.0653**	**0.0743**
20	MAML-OOD-RM(skyline)	0.0102	0.0120	0.0131	0.0150
	MAML	0.0228	0.0278	0.0432	0.0553
	L2R	0.0169	0.0314	0.0219	0.0289
	NESTEDMAML (ours)	**0.0152**	**0.0153**	**0.0221**	**0.0231**

E.2 Instance-level Weighting For Noisy Labels

Implementation Settings. Similar to OOD experiments, we implement 5-way 3-shot (5-shot) experiments to evaluate the instance-level weighting scheme. We conduct experiments on noisy labels generated by randomly corrupting the original labels in ***mini*-ImageNet**. Specifically, different percentages (20%,30%, 50%) of training samples are selected randomly to flip their labels to simulate the noisy corrupted samples. Intuitively, a deep model robust to noise tries to ignore the data with noisy labels. Note that data containing noisy labels only exist in the meta-training stage. Hyper-parameters are shown in Appendix E.

Baselines. We compare our NESTEDMAML with the following baselines: (1) **MAML-Noise-RM** serves as a skyline. It is simply modified from MAML, and we manually fix zero weights to instances with noisy labels. (2) **MAML**.

Results. From the results shown in Table 5, we can conclude that NESTEDMAML performs better than MAML with high accuracies. Furthermore, to circumvent overfitting and reduce computational complexity due to the weight matrix's high dimension, we group instance weights with 200 clusters by K-means, where instances in each cluster share the same weight initialized at 0.005.

Table 5: Test accuracies on ***mini*-Imagenet** with 20%, 30%, and 50% flipped noisy labels during the meta-training phase.

	5-way 3-shot			**5-way 5-shot**		
Noise Ratio	20%	30%	50%	20%	30%	50%
MAML-Noise-RM	$60.2_{\pm 0.02}$	$59.35_{\pm 0.01}$	$58.21_{\pm 0.71}$	$61.2_{\pm 0.21}$	$60.3_{\pm 0.32}$	$59.1_{\pm 0.68}$
MAML	$54.8_{\pm 0.64}$	$53.9_{\pm 1.10}$	$51.8_{\pm 0.12}$	$59.2_{\pm 0.28}$	$57.6_{\pm 0.36}$	$53.5_{\pm 0.48}$
NESTEDMAML (ours)	$\mathbf{55.24}_{\pm 0.72}$	$\mathbf{54.7}_{\pm 1.20}$	$\mathbf{53.68}_{\pm 0.21}$	$\mathbf{59.6}_{\pm 0.54}$	$\mathbf{58.16}_{\pm 0.87}$	$\mathbf{55.61}_{\pm 1.32}$

E.3 First-Order Approximation (NESTEDMAML-FO)

Even after the one step gradient approximation, the weight gradient calculation involves calculating multiple Hessian vector products, which is expensive. Since the coefficient of the Hessian vector-product term in the weight update (Eq. (9)) involves the product of three learning rate terms $\eta\alpha\gamma$, we can make an approximation that the term involving the Hessian vector-product term is close to 0, given that the above learning rates are small. The approximated weight update takes the following

form (Eq.(11)):

$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \frac{\eta\gamma}{mn} \sum\nolimits_{j=1}^{n} \nabla_{\phi_j} \mathcal{L}_{V_j} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^S))^\intercal$$

This approximation is similar to the first-order approximation given in [7] where the second and higher-order terms are neglected. We want to show a faster way to solve the *nested bi-level* weight optimization problem with a tradeoff in performance. Our experimental results show that we achieve state-of-the-art performance using NESTEDMAML. Our results also show that NESTEDMAML-FO leads to a loss in performance with a commensurate gain in speed compared to the unmodified NESTEDMAML version.

E.4 More Experimental Details

Datasets. ***Mini*-ImageNet** [22] contains 60,000 images of size $84 \times 84 \times 3$ from 100 classes. We use the split proposed in [22]: 64 classes for training, 12 classes for validation and 24 classes for testing. **SVHN** [18], a street view house numbers dataset, contains 26,032 images of size $32 \times 32 \times 3$ from 10 digits classes. **FashionMNIST** [33], a fashion dataset(*i.e.* clothes, shoes, *etc*), contains 60,000 grayscale images of size 28×28 pixels from 10 classes.

Details of Settings for Task-level Weighting. As aforementioned, our backbone follows the same architecture as the embedding function used by [7]. Specially, the backbone structure consists of 4 modules, each of which contains a 3×3 convolutions and 64 filters, followed by batch normalization, a ReLU, and a 2×2 max-pooling with stride 2. To reduce overfitting, 32 filters per layer are considered. We use the same model for OOD and ID tasks during the meta-training stage, so it's necessary to make sure the image sizes are consistent. We resize the image size of SVHN and FashionMNIST to $84 \times 84 \times 3$ which is consistent with *mini*-ImageNet when evaluating the task-level weighting scheme. We also use the same backbone when evaluating the instance-level weighting scheme. Cross entropy loss function is used for these two schemes.

Parameter Tuning for Task-level Scheme in Section 5.1. All baseline approaches follow the original implementation including hyper-parameters. For our NESTEDMAML algorithm, all step sizes (α, η, γ) are chosen from $\{0.0001, 0.001, 0.01, 0.1\}$. Batch size (m, n) are chose from $\{4, 10, 20, 25, 32\}$. The number of iterations are chosen from $\{10,000,\ 20,000,\ 30,000,\ 40,000,\ 60,000\}$. The number of clusters used in K-means is chosen from $\{50,\ 200,\ 1,000,\ 5,000,\ 10,000\}$. The selected best ones are: Fast model parameters step size $\alpha = 0.01$, meta parameters step size $\eta = 0.001$, weight update step size $\gamma = 0.1$; mini-batch size $m = n = 10$; the number of iterations in $30\%, 60\%$ are $30,000$, 90% is $60,000$ respectively. The number of clusters is 200.

Parameter Tuning for Instance-level Scheme in Section E.2. Tuning hyper-parameters follows the same aforementioned strategy. The selected best ones are: Fast model parameters step size $\alpha = 0.01$, meta parameters step size $\eta = 0.001$, weight update step size $\gamma = 0.01$; mini-batch size $m = n = 10$; the number of iterations in $20\%, 30\%, 50\%$ are $20,000$. The number of clusters is 200.

Other related hyperparameters are kept the same with MAML. For example, 5 gradient steps are used when training the backbone in these two schemes, and 10 gradient steps during the meta-test stage. The number of instances in the query set of each task is 15.