2021

Visual Learning Beyond Human Curated Datasets

Muhammad Abdullah Jamal
University of Central Florida

VISUAL LEARNING BEYOND HUMAN CURATED DATASETS

by

MUHAMMAD ABDULLAH JAMAL
M.S. University of Central Florida, 2020
B.E. National University of Sciences and Technology, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2021

Major Professor: Liqiang Wang, Boqing Gong

ABSTRACT

The success of deep neural networks in a variety of computer vision tasks heavily relies on large-scale datasets. However, it is expensive to manually acquire labels for large datasets. Given the human annotation cost and scarcity of data, the challenge is to learn efficiently with insufficiently labeled data. In this dissertation, we propose several approaches towards data-efficient learning in the context of few-shot learning, long-tailed visual recognition, and unsupervised and semi-supervised learning. In the first part, we propose a novel paradigm of Task-Agnostic Meta-Learning (TAML) algorithms to improve few-shot learning. Furthermore, in the second part, we analyze the long-tailed problem from a domain adaptation perspective and propose to augment the classic class-balanced learning for longtails by explicitly estimating the differences between the class-conditioned distributions with a meta-learning approach. Following this, we propose our lazy approach based on an intuitive teacher-student scheme to enable the gradient-based meta-learning algorithms to explore long horizons. Finally, in the third part, we propose a novel face detector adaptation approach that is applicable whenever the target domain supplies many representative images, no matter they are labeled or not. Experiments on several benchmark datasets verify the efficacy of the proposed methods under all settings.

To my family.

ACKNOWLEDGMENTS

First of all, I would like to thank my academic advisors Dr. Liqiang Wang and Dr. Boqing Gong for their endless support during the course of my PhD. Secondly, I would also like to thank my advisory committee members, Dr. Ulas Bagci and Dr. Gita Sukthankar for their time and comments. Moreover, I would also like to thank Dr. Guo-Jun Qi for mentoring me during the spring, and summer 2018.

I would also want to mention my friends, and labmates Mohamed Elfeki, Yandong Li, Dongdong Wang, Yifan Ding, Muhammad Waqas, and Ahmed Abbas, for their friendship and conversations for interesting ideas. Last but not least, I especially want to thank my mother, my siblings for supporting me during my PhD studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation

Thanks to deep learning, we have achieved remarkable results in computer vision tasks such as object recognition [27], semantic segmentation [191, 185], and detection [133, 130] in recent years. Unfortunately, this success comes at a cost. Deep learning requires a large amount of labeled data (e.g., thousands of examples for a class) for training to solve a particular task. However, human annotation cost and scarcity of data can significantly hinder the performance of deep learning models. For example, in the case of few-shot learning, a deep learning model cannot perform well given few examples. On the other hand, a human can learn and adapt quickly from few examples using prior knowledge. Another example is long-tailed training sets where many tail classes have fewer examples and deep learning models can be over-fitted to the majority classes. Last, but not the least, is the case of unsupervised and semi-supervised learning [28] where the goal is to learn from totally unlabeled data or partially labeled and unlabeled data respectively. These examples motivate us towards the development of data-efficient models which can generalize well given insufficient labeled data. In this proposal, we are proposing different approaches to tackling few-shot learning, long-tailed visual recognition, unsupervised, and semi-supervised learning.

1.2 Contribution

Meta-learning approaches have been proposed to tackle the few-shot learning problem. Typically, a meta-learner is trained on a variety of tasks in the hopes of being generalizable to new tasks. However, the generalizability on new tasks of a meta-learner could be fragile when it is over-trained on existing tasks during the meta-training phase. In other words, the initial model of a

meta-learner could be too biased toward existing tasks to adapt to new tasks, especially when only very few examples are available to update the model. To avoid a biased meta-learner and improve its generalizability, we propose a novel paradigm of Task-Agnostic Meta-Learning (TAML) algorithms [71]. Specifically, we present an entropy-based approach that meta-learns an unbiased initial model with the largest uncertainty over the output labels by preventing it from over-performing in classification tasks. Alternatively, a more general inequality-minimization TAML is presented for more ubiquitous scenarios by directly minimizing the inequality of initial losses beyond the classification tasks wherever a suitable loss can be defined. Experiments on benchmarked datasets demonstrate that the proposed approaches outperform compared meta-learning algorithms in both few-shot classification and reinforcement learning tasks.

Besides few-shot learning, deep learning models can also under-perform in the case of long-tailed training sets. Longtails are a challenging problem as it poses a mismatch between the training set seen by a machine learning model and our expectation of the model to perform well on all classes. We analyze this mismatch from a domain adaptation point of view. First of all, we connect existing class-balanced methods for long-tailed classification to target shift, a well-studied scenario in domain adaptation. The connection reveals that these methods implicitly assume that the training data and test data share the same class-conditioned distribution, which does not hold in general and especially for the tail classes. While a head class could contain abundant and diverse training examples that well represent the expected data at inference time, the tail classes are often short of representative training data. To this end, we propose to augment the classic class-balanced learning by explicitly estimating the differences between the class-conditioned distributions with a meta-learning approach [73]. We validate our approach with six benchmark datasets and three loss functions.

Gradient-based meta-learning first trains task-specific models by an inner loop and then backpropagates meta-gradients through the loop to update the meta-model. To avoid high-order gradients,

existing methods either take a small number of inner steps or approximate the meta-updates for the situations that the meta-model and task models lie in the same space. To enable long inner horizons for more general meta-learning problems, we instead propose an intuitive teacher-student strategy. The key idea is to employ a student network to adequately explore the search space of task-specific models, followed by a teacher's "leap" toward the regions probed by the student. The teacher not only arrives at a high-quality model but also defines a lightweight computational graph for the meta-gradients. Our approach is generic; it performs well when applied to four meta-learning algorithms over three tasks: few-shot learning, long-tailed object recognition, and adversarial blackbox attack.

Moreover, we propose a novel face detector adaptation approach [72] that works as long as there are representative images of the target domain no matter they are labeled or not and, more importantly, without the need of accessing the training data of the source domain. It is often desirable to have some built-in schemes for a face detector to automatically adapt, e.g., to a particular user's photo album (the target domain). Our approach explicitly accounts for the notorious negative transfer caveat in domain adaptation thanks to a residual loss by design. Moreover, it does not incur catastrophic interference with the knowledge learned from the source domain and, therefore, the adapted face detectors maintain about the same performance as the old detectors in the original source domain. As such, our adaption approach to face detectors is analogous to the popular interpolation techniques for language models; it may opens a new direction for progressively training the face detectors domain by domain. We report extensive experimental results to verify our approach on two massively bench-marked face detectors.

1.3 Outline

This proposal is organized as follows.

Chapter 1 discusses the motivation of learning from insufficiently labeled data and highlights the contributions of this proposal.

Chapter 2 presents related works on few-shot learning, long-tail visual recognition, and unsupervised and semi-supervised learning for face detector adaptation.

Chapter 3 proposes a novel paradigm of Task-Agnostic Meta-Learning (TAML) algorithms to tackle few-shot learning. Specifically, we present an entropy-based approach, and a more general inequality-minimization TAML for more ubiquitous scenarios.

Chapter 4 analyzes the long-tail problem from a domain adaptation perspective by proposing to augment the classic class-balanced learning by explicitly estimating the differences between the class-conditioned distributions with a meta-learning approach.

Chapter 5 proposes an intuitive teacher student scheme to enable the gradient-based meta-learning algorithms to explore long horizons by the inner loop.

Chapter 6 proposes a novel face detector adaptation approach that works as long as there are representative images of the target domain no matter they are labeled or not and, more importantly, without the need of accessing the training data of the source domain.

Chapter 7 concludes this dissertation and highlights the future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Few-Shot Learning

Recently, meta-learning approaches have been proposed to solve the few-shot learning. The idea of meta-learning has been proposed more than a couple of decades ago [143, 120, 161]. Most of the approaches to meta-learning include learning a learner's model by training a meta-learner. Recent studies towards meta-learning for deep neural networks include learning a hand-designed optimizer like SGD by parameterizing it through recurrent neural networks. Li [89], and Andrychowicz [4] studied a LSTM based meta-learner that takes the gradients from learner and performs an optimization step. To solve few-shot classification problems, [129] used the same LSTM based meta-learner approach in which LSTM meta-learner takes the gradient of a learner and proposed an update to the learner's parameters. The approach learns both weight initialization and an optimizer of the model weights. Finn [41] proposed a more general approach for meta-learning known as MAML by simply learning weight initialization for a learner through a fixed gradient descent. It trains a model on a variety of tasks to have a good initialization point that can be quickly adapted (few or one gradient steps) to a new task using few training examples. Meta-SGD [98] extends the MAML, which not only learns weight initialization but also the learner's update step size. [117] proposes a temporal convolution and attention based meta-learner called SNAIL that achieves state-of-the-art performance for few-shot classification tasks and reinforcement learning tasks.

Other paradigms of meta-learning approaches include training a memory augmented neural network on existing tasks by coupling with LSTM or feed-forward neural network controller [141, 119]. There are also several non-meta-learning approaches to few-shot classification problem by designing specific neural architectures. For example, [81] trains a Siamese network to compare new examples with existing ones in a learned metric space. Vinyals [166] used a differentiable

nearest neighbour loss by utilizing the cosine similarities between the features produced by a convolutional neural network. [154] proposed a similar approach to matching net but used a square euclidean distance metric instead. In this proposal, we mainly focus on the meta-learning approaches and their applications to few-shot classification and reinforcement tasks.

2.2 Long-Tailed Visual Recognition

Our work is closely related to the class-balanced methods briefly reviewed in Chapter 4. In this section, we discuss domain adaptation and the works of other types for tackling the long-tailed visual recognition.

Metric learning, hinge loss, and head-to-tail knowledge transfer. Hinge loss, focal loss [184] and metric learning are flexible tools for one to handle the long-tailed problem [14, 66, 183, 189, 62, 173]. They mostly contain two major steps. One is to sample or group the data being aware of the long-tailed property, and the other is to construct large-margin losses. Our approach is loss-agnostic, and we show it can benefit different loss functions in the experiments. Another line of research is to transfer knowledge [60, 31, 93] from the head classes to the tail. Yin et al. transfer intra-class variance from the head to tail [181], Liu et al. add a memory module to the neural networks to transfer semantic features [104], and Wang et al. employ a meta network to regress network weights between different classes [172].

Hard example mining and weighting. Hard example mining is prevalent and effective in object detection [39, 111, 133, 101]. While it is not particularly designed for the long-tailed recognition, it can indirectly shift the model's focus to the tail classes, from which the hard examples usually originate (cf. [25, 32, 45] and our experiments). Nonetheless, such methods could be sensitive to outliers or unnecessarily allow a minority of examples to dominate the training. The recently pro-

posed instance weighting by meta-learning methods [132, 153] alleviate those issues. Following the general meta-learning principle [41, 71, 99], they set aside a validation set to guide how to weigh the training examples by gradient descent. Similar schemes are used in learning from noisy data [75, 26, 171, 177].

Domain adaptation. In real-world applications, there often exist mismatches between the distributions of training data and test data for various reasons [162, 190, 59]. There has been a rich line of works on domain adaptation for generic visual recognition [56, 23], such as object recognition [138], action recognition [90], Webly-supervised learning [9, 35, 17], attribute detection [47], etc. Domain adaptation methods aim to mitigate the mismatches so that the learned models can generalize well to the inference-time data [152, 52, 51]. There are some approaches that handle the imbalance problem in domain adaptation. Zou et al. [195] deal with the class imbalance by controlling the pseudo-label learning and generation using the confidence scores that are normalized class-wise. Yan et al. [178] use a weighted maximum mean discrepancy to handle the class imbalance in unsupervised domain adaptation. We understand the long-tail challenge in visual recognition from the perspective of domain adaptation. While domain adaptation methods need to access a large amount of unlabeled (and sometimes also a small portion of labeled) target domain data, we do not access any inference-time data in our approach. Unlike existing weighting methods in domain adaptation [21, 67, 186], we meta-learn the weights.

2.3 Long-Horizon Gradient based Meta-learning

Meta-learning has been a long-standing sub-field in machine learning [143, 161, 120]. Early approaches update a model's parameters by training a meta-learner [7, 8, 144]. This has been well studied in optimizing neural networks, and one such family of meta-learning learns an optimizer [129, 89, 4]. A specialized neural network takes gradients as input and outputs an update

rule for the learner. In addition to the update rule, [129] also learn the weight initialization for few-shot learning. Finally, there are several approaches [115, 175] for training generic optimizers that can be applied broadly to different neural networks and datasets.

Under the context of few-shot learning, another family of meta-learning involves metric-learning based methods [166, 154, 117, 81, 125], which learn a metric space to benefit different few-shot learning tasks. The goal is to find the similarity between two samples regardless of their classes using some distance metric so that the similarity function can be used to classify the unseen classes at the test stage. Some recent studies along this line include Relation Network [158], which uses a relation module as the similarity function, ridge regression [10], and graph neural networks [142]. Besides these approaches, some of the methods have been reviewed in 2.1.

More recently, gradient-based meta-learning gains its momentum, and a variety of methods have been proposed in this vein. The most notable one among them might be [41], where the goal is to learn the network weight initialization so that it can adapt to unseen tasks rapidly. There have been extensions to improve MAML. Meta-SGD [99] learns the learning rates along with the weight initialization. Regularization techniques [180, 71] are introduced to MAML to mitigate over-fitting. [126] preconditions on the gradients in the inner loop by learning a curvature. Despite MAML's popularity, it is still computationally expensive and consumes large memory due to the computation of high-order derivatives. The authors show that the first-order approximation, which neglects the gradients of the inner loop during meta-optimization, performs about the same as the original MAML. Another first-order meta-learning method is Reptile [122], which decouples the inner and outer optimization steps. iMAML [127] provides an approximate solution for meta-gradients by using an algorithm based on conjugate gradients, and its low-level optimization is similar to Meta-MinibatchProx [193]. The idea is to add an ℓ_2 regularizer in the inner loop, allowing the updated parameters close to the initial parameters.

2.4 Face Detector Adaptation under Unsupervised and Semi-supervised Setting

Face detector adaptation. Jain and Learned-Miller use a Gaussian process to update the low detection scores by assuming smoothness of the detections and that the detected regions of high scores are more likely correct than the others [70]. Wang et al. [170] and Li et al. [87] make similar assumptions and yet use the regions of high detection scores to re-train a new detector for the target domain using vocabulary trees and probabilistic elastic part models, respectively. When the target domain comprises video sequences, the motion and tracking cues are usually very effective for adapting the detectors [159, 148, 169, 140, 136].

Domain adaptation. Most of the domain adaptation work is briefly reviewed in the section 2.2. The goal is to minimize the discrepancy between the source and target by exploring the data from both domains. However, the modern face detectors are often trained from an extreme-scale training set, making it hard to carry the source data to the adaptation stage. Domain adaptation in the absence of the source data [19, 84] is the most relevant to ours. Such methods use the source models either for regularization [84] or to augment the features of the target data [19], while we consider a different problem, deep face detectors, and refer to the source model in both the cost function and the classifier of the target face detector.

Negative transfer is a notorious caveat in domain adaptation [135, 50, 146, 147]. Whereas existing works attempt to solve this problem by defining intuitive statistical measures, we directly tackle it with a novel cost function motivated by the safe semi-supervised learning [94, 95, 108]. Nonetheless, we devise the cost function in such a way of seamlessly integrating it with the deep models. Besides, we derive an analytic form for the unsupervised adaptation, getting rid of the cumbersome EM style optimization.

Catastrophic forgetting or interference [114, 128, 44, 113] refers to that a pre-trained network

cannot perform well on the old tasks after it is fine-tuned for a new task. Recent years witness an upsurge of interest in this problem, including the exploitation of a local winner-takes-all activation function [156], dropout [55, 155], a knowledge distillation loss [97, 134, 65, 168, 167], pathway connections [40], and progressive networks [137]. We argue that it is probably easier to deal with the catastrophic forgetting problem for domain adaptation which can be seen as a special case of sequential multi-task learning, due to that the source and target domains share the same semantic labels. We leverage exactly this idiosyncrasy to re-parameterize the target classifier as the source classifier plus an offset.

CHAPTER 3: FEW-SHOT LEARNING

The key to achieving human level intelligence is to learn from a few labeled examples. Human can learn and adapt quickly from a few examples using prior experience. We want our learner to be able to learn from a few examples and quickly adapt to a changing task. All these concerns motivate to study the few-shot learning problem. The advantage of studying the few-shot problem is that it only relies on few examples and it alleviates the need to collect large amount of labeled training set which is a cumbersome process.

Recently, meta-learning approach is being used to tackle the problem of few-shot learning. A meta-learning model usually contains two parts – an initial model, and an updating strategy (e.g., a parameterized model) to train the initial model to a new task with few examples.

Then the goal of meta-learning is to automatically meta-learn the optimal parameters for both the initial model and the updating strategy that are generalizable across a variety of tasks. There are many meta-learning approaches that show promising results on few-shot learning problems. For example, Meta-LSTM [129] uses LSTM meta-learner that not only learns initial model but also the updating rule. On the contrary, MAML [41] only learns an initial model since its updating rule is fixed to a classic gradient descent method as a meta-learner.

The problem with existing meta-learning approaches is that the initial model can be trained biased towards some tasks, particularly those sampled in meta-training phase. Such a biased initial model may not be well generalizable to an unseen task that has a large deviation from meta-training tasks, especially when very few examples are available on the new task. This inspires us to meta-train an unbiased initial model by preventing it from overperforming on some tasks or directly minimizing the inequality of performances across different tasks, in a hope to make it more generalizable to

unseen tasks. To this end, we propose a Task-Agnostic Meta-Learning (TAML) algorithms [71][1] in this chapter.

Specifically, we propose two novel paradigms of TAML algorithms – an entropy-based TAML and inequality-minimization measures based TAML. The idea of using entropy based approach is to maximize the entropy of labels predicted by the initial model to prevent it from overperforming on some tasks. However, the entropy-based approach is limited to discrete outputs from a model, making it more amenable to classification tasks.

The second paradigm is inspired by inequality measures used in Economics. The idea is to meta-train an initial model in such a way that it directly minimizes the inequality of losses by the initial model across a variety of tasks. This will force the meta-learner to learn a unbiased initial model without over-performing on some particular tasks. Meanwhile, any form of losses can be adopted for involved task without having to rely on discrete outputs. This makes this paradigm more ubiquitous to many scenarios beyond classification tasks.

3.1 Approach

Our goal is to train a model that can be task-agnostic in a way that it prevents the initial model or learner to over-perform on a particular task. In this section, we will first describe our entropy based and inequality-minimization measures based approach to the problem, and then we will discuss some of the inequality measures that we used in the paper.

[1]This chapter's material has been accepted in 2019 in Conference on Computer Vision and Pattern Recognition (CVPR 2019), "Task Agnostic Meta-Learning for Few-Shot Learning" authored by Muhammad Abdullah Jamal and Guo-Jun Qi.

3.1.1 Task-Agnostic Meta-Learning

In this section, we propose a task-agnostic approach for few-shot meta-learning. The goal of few-shot meta-learning is to train a model in such a way that it can learn to adapt rapidly using few samples for a new task. In this meta-learning approach, a learner is trained during a meta-learning phase on variety of sampled tasks so that it can learn new tasks , while a meta-learner trains the learner and is responsible for learning the update rule and initial model.

The problem with the current meta-learning approach is that the initial model or learner can be biased towards some tasks sampled during the meta-training phase, particularly when future tasks in the test phase may have discrepancy from those in the training tasks. In this case, we wish to avoid an initial model over-performing on some tasks. Moreover, an over-performed initial model could also prevent the meta-learner to learn a better update rule with consistent performance across tasks.

To address this problem, we impose an unbiased task-agnostic prior on the initial model by preventing it from over-performing on some tasks so that a meta-learner can achieve a more competitive update rule. There have been many meta-learning approaches to few-shot learning problems that have been briefly discussed in the section 2. While the task-agnostic prior is a widely applicable principle for many meta-learning algorithms, we mainly choose Model-Agnostic Meta Learning approach (MAML) as an example to present the idea, and it is not hard to extend to other meta-learning approaches.

In the following, we will depict the idea by presenting two paradigms of task-agnostic meta-learning (TAML) algorithms – the entropy-maximization/reduction TAML and inequality-minimization TAML.

3.1.2 Entropy-Maximization/Reduction TAML

For simplicity, we express the model as a function f_θ that is parameterized by θ. For example, it can be a classifier that takes an input example and outputs its discrete label. During meta-training, a batch of tasks are sampled from a task distribution $p(\mathcal{T})$, and each task is K-shot N-way problem where K represents the number of training examples while N represent the number of classes depending on the problem setting. In the MAML, a model is trained on a task $\mathcal{T}_i$ using K examples and then tested on a few new examples $\mathcal{D}_{val}$ for this task.

A model has an initial parameter θ and when it is trained on the task $\mathcal{T}_i$, its parameter is updated from θ to θ_i by following an updating rule. For example, for K-shot classification, stochastic gradient descent can be used to update model parameter by $\theta_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ that attempts to minimize the cross-entropy loss $\mathcal{L}_{\mathcal{T}_i}(f_\theta)$ for the classification task $\mathcal{T}_i$ over K examples.

To prevent the initial model f_θ from over-performing on a task, we prefer it makes a random guess over predicted labels with an equal probability so that it is not biased towards the task. This can be expressed as a maximum-entropy prior over θ so that the initial model should have a large entropy over the predicted labels over samples from task $\mathcal{T}_i$.

The entropy for task $\mathcal{T}_i$ is computed by sampling x_i from $P_{\mathcal{T}_i}(x)$ over its output probabilities $y_{i,n}$ over N predicted labels:

$$\mathcal{H}_{\mathcal{T}_i}(f_\theta) = -\mathbb{E}_{x_i \sim P_{\mathcal{T}_i}(x)} \sum_{n=1}^{N} \hat{y}_{i,n} \log(\hat{y}_{i,n}) \tag{3.1}$$

where $[y_{i,1}, \cdots, y_{i,N}] = f_\theta(x_i)$ is the predictions by f_θ, which are often an output from a softmax layer in a classification task. The above expectation is taken over x_i's sampled from task $\mathcal{T}_i$.

Alternatively, one can not only maximize the entropy before the update of initial model's parame-

ter, but also minimize the entropy after the update. So overall, we maximize the entropy reduction for each task $\mathcal{T}_i$ as $\mathcal{H}_{\mathcal{T}_i}(f_\theta) - \mathcal{H}_{\mathcal{T}_i}(f_{\theta_i})$. The minimization of $\mathcal{H}_{\mathcal{T}_i}(f_{\theta_i})$ means that the model can become more certain about the labels with a higher confidence after updating the parameter θ to θ_i. This entropy term can be combined with the typical meta-training objective term as a regularizer to find the optimal θ, which is

$$\min_\theta \mathbb{E}_{\mathcal{T}_i \sim P(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}) + \lambda[-\mathcal{H}_{\mathcal{T}_i}(f_\theta) + \mathcal{H}_{\mathcal{T}_i}(f_{\theta_i})]$$

where λ is a positive balancing coefficient, and the first term is the expected loss for the updated model f_{θ_i}. The entropy-reduction algorithm is summarized in 1.

Algorithm 1 TAML for Few-Shot Classification

Require: $p(\mathcal{T})$: distribution over tasks.
Require: α, β: hyperparameters
  Randomly Initialize θ
  while not done **do**
    Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ **do**
      Sample K samples from $\mathcal{T}_i$
      Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ and $\mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using K samples.
      Compute adapted parameters using gradient descent.
      $\theta_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}$
      Sample $\mathcal{D}_{val}$ from $\mathcal{T}_i$ for meta update.
    end for
    if Entropy-Reduction TAML **then**
      Update $\theta \leftarrow \theta - \beta \nabla_\theta \{\mathbb{E}_{\mathcal{T}_i \sim P(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ + $\lambda[-\mathcal{H}_{\mathcal{T}_i}(f_\theta) + \mathcal{H}_{\mathcal{T}_i}(f_{\theta_i})]\}$ using $\mathcal{D}_{val}$, $\mathcal{L}_{\mathcal{T}_i}$, and $\mathcal{H}_{\mathcal{T}_i}$.
    else if Inequality Measures Based TAML **then**
      Update $\theta \leftarrow \theta - \beta \nabla_\theta [\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ + $\lambda \mathcal{I}_\mathcal{E}(\{\mathcal{L}_{\mathcal{T}_i}(f_\theta)\})]$ using $\mathcal{D}_{val,i}$, $\mathcal{L}_{\mathcal{T}_i}$, and $\mathcal{I}_\mathcal{E}$
    end if
  end while

Unfortunately, the entropy-based TAML is subject to a critical limitation – it is only amenable to discrete labels in classification tasks to compute the entropy. In contrast, many other learning problems, such as regression and reinforcement learning problems, it is often trained by minimizing some loss or error functions directly without explicitly accessing a particular form of outputs like discrete labels. To make the TAML widely applicable, we need to define an alternative metric to measure and minimize the bias across tasks.

3.1.3 Inequality-Minimization TAML

We wish to train a task-agnostic model in meta-learning such that its initial performance is unbiased towards any particular task $\mathcal{T}_i$. Such a task-agnostic meta-learner would do so by minimizing the inequality of its performances over different tasks.

To this end, we propose an approach based on a large family of statistics used to measure the "economic inequalities" to measure the "task bias". The idea is that the loss of an initial model on each task $\mathcal{T}_i$ is viewed as an income for that task. Then for the TAML model, its loss inequality over multiple tasks is minimized to make the meta-learner task-agnostic.

Specifically, the bias of the initial model towards any particular tasks is minimized during meta-training by minimizing the inequality over the losses of sampled tasks in a batch. So, given an unseen task during testing phase, a better generalization performance is expected on the new task by updating from an unbiased initial model with few examples. The key difference between both TAMLs lies that for entropy, we only consider one task at a time by computing the entropy of its output labels. Moreover, entropy depends on a particular form or explanation of output function, e.g., the SoftMax output. On the contrary, the inequality only depends on the loss, thus it is more ubiquitous.

The complete algorithm is explained in 1. Formally, consider a batch of sampled tasks $\{\mathcal{T}_i\}$ and their losses $\{\mathcal{L}_{\mathcal{T}_i}(f_\theta)\}$ by the initial model f_θ, one can compute the inequality measure by $\mathcal{I}_\mathcal{E}(\{\mathcal{L}_{\mathcal{T}_i}(f_\theta)\})$ as discussed later. Then the initial model parameter θ is meta-learned by minimizing the following objective

$$\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}\left[\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\right] + \lambda \mathcal{I}_\mathcal{E}(\{\mathcal{L}_{\mathcal{T}_i}(f_\theta)\})$$

through gradient descent as shown in Algorithm 1. It is worth noting that the inequality measure is computed over a set of losses from sampled tasks. The first term is the expected loss by the model f_{θ_i} after the update, while the second is the inequality of losses by the initial model f_θ before the update. Both terms are a function of the initial model parameter θ since θ_i is updated from θ. In the following, we will elaborate on some choices on inequality measures $\mathcal{I}_\mathcal{E}$.

3.1.3.1 Inequality Measures

Inequality measures [150] are instrumental towards calculating the economic inequalities in the outcomes that can be wealth, incomes, or health related metrics. In meta-learning context, we use $\ell_i = \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ to represent the loss of a task $\mathcal{T}_i$, $\bar{\ell}$ represents the mean of the losses over sampled tasks, and M is the number of tasks in a single batch. The inequality measures used in TAML are briefly described below.

The **Theil Index** [160] measure has been derived from redundancy in information theory, which is defined as the difference between the maximum entropy of the data and an observed entropy. Suppose that we have M losses $\{\ell_i | i = 1, \cdots, M\}$, then Thiel Index is defined as

$$T_T = \frac{1}{M} \sum_{i=1}^{M} \frac{\ell_i}{\bar{\ell}} \ln \frac{\ell_i}{\bar{\ell}} \tag{3.2}$$

Generalized Entropy index [22] has been proposed to measure the income inequality. It is not a single inequality measure, but it is a family that includes many inequality measures like Thiel Index, Thiel L etc. For some real value α, it is defined as:

$$GE(\alpha) = \begin{cases} \dfrac{1}{M\alpha(\alpha-1)} \displaystyle\sum_{i=1}^{M} \left[\left(\frac{\ell_i}{\bar{\ell}} \right)^{\alpha} - 1 \right], & \alpha \neq 0, 1, \\ \dfrac{1}{M} \displaystyle\sum_{i=1}^{M} \frac{\ell_i}{\bar{\ell}} \ln \frac{\ell_i}{\bar{\ell}}, & \alpha = 1, \\ -\dfrac{1}{M} \displaystyle\sum_{i=1}^{M} \ln \frac{\ell_i}{\bar{\ell}}, & \alpha = 0, \end{cases} \tag{3.3}$$

From the equation, we can see that it does represent a family of inequality measures. When α is zero, it is called a mean log deviation of Thiel L, and when α is one, it is actually Thiel Index. A larger GE α value makes this index more sensitive to differences at the upper part of the distribution, and a smaller α value makes it more sensitive to differences at the bottom of the distribution. The **Atkinson Index** [6] is another measure for income inequality which is useful in determining which end of the distribution contributed the most to the observed inequality. It is defined as:

$$A_{\epsilon} = \begin{cases} 1 - \dfrac{1}{\bar{\ell}} \left(\dfrac{1}{N} \displaystyle\sum_{i=1}^{N} \ell_i^{1-\epsilon} \right)^{\frac{1}{1-\epsilon_{at}}}, & \text{for } 0 \leq \epsilon_{at} \neq 1, \\ 1 - \dfrac{1}{\bar{\ell}} \left(\dfrac{1}{N} \displaystyle\prod_{i=1}^{N} \ell_i \right)^{\frac{1}{N}}, & \text{for } \epsilon_{at} = 1, \end{cases} \tag{3.4}$$

where ϵ is called "inequality aversion parameter". When $\epsilon = 0$ the index becomes more sensitive to the changes in upper end of the distribution ,and when it approaches to 1, the index becomes more sensitive to the changes in lower end of the distribution. **Gini-Coefficient** [3] is usually defined as the half of the relative absolute mean difference. In terms of meta-learning, if there are M tasks in

a single batch and a task $\mathcal{T}_i$ loss is represented by ℓ_i, then Gini-Coefficient is defined as:

$$G = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} |\ell_i - \ell_j|}{2n \sum_{i=1}^{M} \ell_i} \tag{3.5}$$

Gini- coefficient is more sensitive to deviation around the middle of the distribution than at the upper or lower part of the distribution.

The **variance of logarithms** [124] is another common inequality measure defined as:

$$V_L(\ell) = \frac{1}{M} \sum_{i=1}^{M} [\ln \ell_i - \ln g(\ell)]^2 \tag{3.6}$$

where g(ℓ) is the geometric mean of ℓ which is defined as $(\prod_{i=1}^{M} \ell_i)^{1/M}$. The geometric mean put greater emphasis on the lower losses of the distribution.

Algorithm 2 Inequality Measures Based TAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks.
Require: α, β: hyperparameters
  Randomly Initialize θ
  while not done **do**
    Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ **do**
      Sample K trajectories $(x_1, a_1, ..., x_T)$ using f_θ in $\mathcal{T}_i$.
      Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ and $\mathcal{L}_{\mathcal{T}_i}$ using K trajectories in Equation 3.7
      Compute adapted parameters using gradient descent : $\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}$.
      Sample trajectories $\mathcal{D}_{val,i}$ using f_{θ_i} in $\mathcal{T}_i$.
    end for
    Update $\theta \leftarrow \theta - \beta \nabla_\theta [\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$
    + $\lambda \mathcal{I}_\mathcal{E}(\{\mathcal{L}_{\mathcal{T}_i}(f_\theta)\})]$ using $\mathcal{D}_{val,i}$, $\mathcal{L}_{\mathcal{T}_i}$, and $\mathcal{I}_\mathcal{E}$
  end while

Table 3.1: Few Shot Classification results on Omniglot dataset for fully connected network and convolutional network on 5-way setting, where * means re-run results as there is no general training/test splitting available for Omniglot, thus we re-run compared models with the same splitting used in running the TAML for a fair comparison. The ± shows 95% confidence interval over tasks.

Methods	5-way	
	1-shot	5-shot
MANN, no conv [141]	82.8%	94.9%
MAML, no conv [41]	89.7 ± 1.1%	97.5 ± 0.6 %(96.1 ± 0.4)%*
TAML(Entropy), no conv	91.19 ± 1.03%	**97.40 ± 0.34%**
TAML(Theil), no conv	91.37 ± 0.97%	96.84 ± 0.36%
TAML(GE(2)), no conv	91.3 ± 1.0%	96.76 ± 0.4%
TAML(Atkinson), no conv	91.77 ± 0.97%	97.0 ± 0.4%
Siamese Nets [81]	97.3%	98.4%
Matching Nets [166]	98.1%	98.9%
Neural Statistician [37]	98.1%	99.5%
Memory Mod. [77]	98.4%	99.6%
Prototypical Nets [154]	98.8%	99.7%
Meta Nets [119]	98.9%	-
Snail [117]	99.07 ± 0.16%	99.78 ± 0.09%
MAML [41]	98.7 ± 0.4%	**99.9± 0.1%**
MAML+L2 [41]	98.77 ± 0.5%	99.31± 0.1%
Meta-SGD* [98]	97.97 ± 0.7%	98.96± 0.2%
TAML(Entropy)	99.23 ± 0.35%	99.71 ± 0.1%
TAML(Theil)	**99.5 ± 0.3%**	**99.81 ± 0.1 %**
TAML(GE(2))	**99.47 ± 0.25 %**	**99.83 ± 0.09%**
TAML(Atkinson)	99.37 ± 0.3%	99.77 ± 0.1%
TAML (Gini-Coefficient)	99.3 ± 0.32%	99.70 ± 0.1%
TAML(GE(0))	99.33 ± 0.31%	99.75 ± 0.09%
TAML (VL)	99.1 ± 0.36%	99.6 ± 0.1%

3.2 Experiments

We report experiment results in this section to evaluate the efficacy of the proposed TAML approaches on a variety of few-shot learning problems on classification and reinforcement learning.

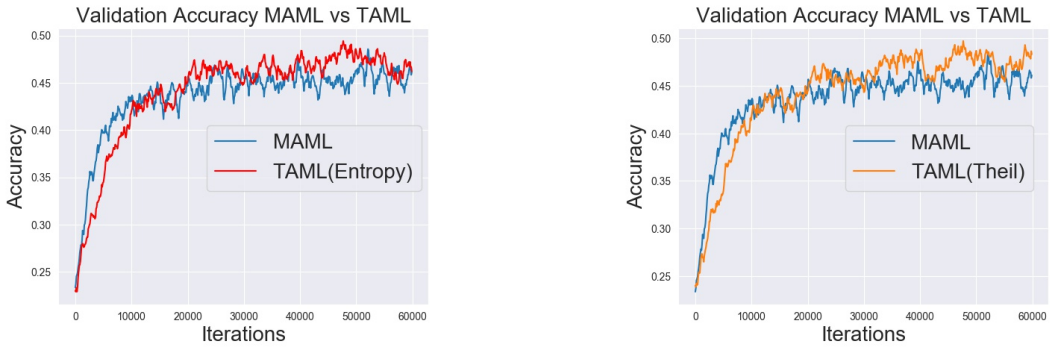
Table 3.2: Few Shot Classification results on Omniglot dataset for CNN on 20-way setting. For a fair comparison, * denotes re-run results by both meta-learning approaches on the same training/test split used in TAML models. The proposed TAML approaches outperform both MAML and Meta-SGD.

Methods	20-way	
	1-shot	5-shot
Siamese Nets [81]	88.2%	97.0%
Matching Nets [166]	93.8%	98.5%
Neural Statistician [37]	93.2%	98.1%
Memory Mod. [77]	95.0%	98.6%
MAML* [41]	90.81 $\pm$ 0.5%	97.49 $\pm$ 0.15%
MAML+L2* [41]	90.93 $\pm$ 0.6%	97.65 $\pm$ 0.18%
Meta-SGD* [98]	93.98 $\pm$ 0.43%	98.42 $\pm$ 0.11%
TAML(Entropy + MAML)	**95.62 $\pm$ 0.5%**	**98.64 $\pm$ 0.13%**
TAML(Theil + Meta-SGD)	**95.15 $\pm$ 0.39%**	**98.56 $\pm$ 0.1%**
TAML(Atkinson + Meta-SGD)	94.91 $\pm$ 0.42%	98.50 $\pm$ 0.1%
TAML (VL + Meta-SGD)	**95.12 $\pm$ 0.39%**	**98.58 $\pm$ 0.1%**
TAML(Theil + MAML)	92.61 $\pm$ 0.46%	98.4 $\pm$ 0.1%
TAML(GE(2) + MAML)	91.78 $\pm$ 0.5%	97.93 $\pm$ 0.1%
TAML(Atkinson + MAML)	93.01 $\pm$ 0.47%	98.21 $\pm$ 0.1%
TAML(GE(0) + MAML)	92.95 $\pm$ 0.5%	98.2 $\pm$ 0.1%
TAML (VL + MAML)	93.38 $\pm$ 0.47%	98.54 $\pm$ 0.1%

3.2.1 Classification

We use two benchmark datasets Omniglot and MiniImagenet for few-shot classification problem. The Omniglot dataset has 1623 characters from 50 alphabets. Each character has 20 instances which are drawn by different individuals. We randomly select 1200 characters for training and remaining for testing. From 1200 characters, we randomly sample 100 for validation. As proposed in [141], the dataset is augmented with rotations by multiple of 90 degrees.

The Mini-Imagenet dataset was proposed by [166] and it consists of 100 classes from Imagenet dataset. We used the same split proposed by [129] for fair comparison. It involves 64 training classes, 12 validation classes and 20 test classes. We consider 5-way and 20-way classification for both 1-shot and 5-shot.



(a) Entropy

(b) Thiel

Figure 3.1: Validation Accuracy of TAML vs MAML on Mini-Imagenet 1-shot 5-way.

For K-shot N-way classification, we first sample N unseen classes from training set and for every N unseen class, we sample K different instances. We follow the same model architecture used by [166]. The Omniglot dataset images are downsampled by 28x28 and we use a strided convolutions instead of max-pooling. The MiniImagenet images are downsampled to 84x84 and we used 32 filters in the convolutional layers for 5-shot setting. For 1-shot setting, we used 64 filters in convolutional layers and we added two dropouts. We also used Leaky-ReLU instead of ReLU as non-linearity. We re-run the MAML on MiniImagenet 1-shot setting for this customized architecture too. We also evaluate the proposed approach on non-convolutional neural network. For a fair comparison with MANN [141] and MAML [41], we follow the same architecture used by MAML [41]. We use Leaky-ReLU as non-linearity instead of ReLU non-linearity.

We train and evaluate the meta-models based on TAML that are unbiased and show they can be adapted to new tasks in few iterations as how they are meta-trained. For Omniglot dataset, we use a batch size of 32 and 16 for 5-way and 20-way classification, respectively. We follow [41] for other training settings. For fair comparison with Meta-SGD on 20-way classification, the model was trained with 1 gradient step. For 5-way Mini-Imagenet, we use a batch size of 4 for both 1-shot and 5-shot settings. For 5-way 5-shot setting, we used a learning rate α of 0.05. For 20-way

classification on MiniImagenet, the learning rate was set to 0.01 for both 1-shot and 5-shot, and each task is updated using one-gradient step. All the models are trained for 60000 iterations. We use the validation set to tune the hyper-parameter λ for both the approaches.

Table 3.3: Few Shot Classification results on Mini-Imagenet dataset on 5-way and 20-way setting. The results for other methods on 5-way are reported from MAML, and for 20-way, the results are reported from Meta-SGD. TAML approaches outperform MAML on both settings and Meta-SGD on 20-way setting.* Accuracy using the comparable network architecture.

Methods	5-way		20 way	
	1-shot	5-shot	1-shot	5-shot
Fine-tune	28.86 ± 0.54%	49.79 ± 0.79%	-	-
Nearest Neighbors	41.08 ± 0.70%	51.04 ± 0.65%	-	-
Matching Nets [166]	43.56 ± 0.84%	55.31 ± 0.73%	17.31 ± 0.22%	22.69 ± 0.20%
Meta-Learn LSTM [129]	43.44 ± 0.77%	60.60 ± 0.71%	16.70 ± 0.23%	26.06 ± 0.25%
TAML(Theil + Meta-Learn LSTM)	**46.28 ± 0.79%**	**62.92 ± 0.66%**	-	-
MAML (firstorderapprox.) [41]	48.07 ± 1.75%	63.15 ± 0.91%	-	-
MAML [41]	48.70 ± 1.84%	63.11 ± 0.92%	16.49 ± 0.58%	19.29 ± 0.29%
MAML (64 filters) [41]	49.5 ± 1.8%	-	-	-
Meta-SGD [98]	50.47 ± 1.87%	64.03 ± 0.94%	17.56 ± 0.64%	28.92 ± 0.35%
Prototypical network [154]	46.61 ± 0.78%	65.77 ± 0.70%	-	-
Reptile [122]	49.97 ± 0.32%	**65.99 ± 0.58%**	-	-
LLAMA [58]	49.40 ± 1.83%	-	-	-
SNAIL* [117]	45.1%	55.2%	-	-
GNN [142]	50.33 ± 0.36%	**66.41 ± 0.63%**	-	-
Relation Network [158]	50.44 ± 0.82%	65.32 ± 0.70%	-	-
TAML(Entropy + MAML)	**51.73 ± 1.88%**	**66.05 ± 0.85%**	-	-
TAML(Theil + MAML)	**51.5 ± 1.86%**	**65.94 ± 0.9%**	**18.74 ± 0.65%**	**25.77 ± 0.33%**
TAML(GE(2) + MAML)	50.87 ± 1.86%	65.18 ± 0.9%	18.22 ± 0.67%	24.89 ± 0.34%
TAML(Atkinson + MAML)	51.03 ± 1.83%	65.24 ± 0.91%	-	-
TAML(GE(0) + MAML)	50.93 ± 1.9%	65.71 ± 0.9%	**18.95 ± 0.68%**	24.53± 0.33%
TAML (VL + MAML)	51.13 ± 1.85%	**66.0 ± 0.89%**	18.13 ± 0.64%	25.33 ± 0.32%
TAML(GE(0) + Meta-SGD)	51.1 ± 1.88%	65.51 ± 0.93%	19.45 ± 0.67%	29.75± 0.34%
TAML (VL + Meta-SGD)	**51.77 ± 1.86%**	65.6 ± 0.93%	**19.73 ± 0.65%**	**29.81 ± 0.35%**

3.2.1.1 Results

We report the results for 5-way Omniglot for both fully connected network and convolutional network. We added one more baseline in which we add L2 regularizer in the MAML's cost function

and from Table 3.1 3.2, it shows that the performance is about the same as MAML for both 5-way and 20-way classification settings.The convolutional network learned by TAML outperforms all the state-of-the-art methods in Table 3.1. For 20-way classification, we re-ran the Meta-SGD algorithm with our own training/test splitting for fair comparison since the Meta-SGD is not open-sourced and their training/test split is neither available. The results are reported in the Table 3.2. It can be shown that TAML outperforms MAML and Meta-SGD for both 1-shot and 5-shot settings. The results also show that TAML achieves much more competitive rule during the training.

For MiniImagenet, the proposed TAML approaches outperform the compared ones for 1-shot 5-way classification problem. For 5-shot 5 way setting, our entropy based approach still outperforms all the other methods except for GNN which is still within the variance of entropy based approach. The entropy based TAML achieves the best performance compared with inequality-minimization TAML for 5-shot problem. For 20-way setting, we use the reported results from Meta-SGD for both MAML and Meta-SGD. We outperform both MAML and Meta-SGD for both 1-shot and 5-shot settings. It is interesting to note that MAML performs poor compared with matching nets and Meta-learner LSTM when it is trained using one gradient step as reported in Table 3.3. The test accuracy for prototypical results which is reported in Table 3.3 for models matches train and test ”shot” and ”way”. The results reported by [154] requires 30-way 15 queries per training episode for 1-shot and, 20-way 15 queries per training episode for 5-shot results.

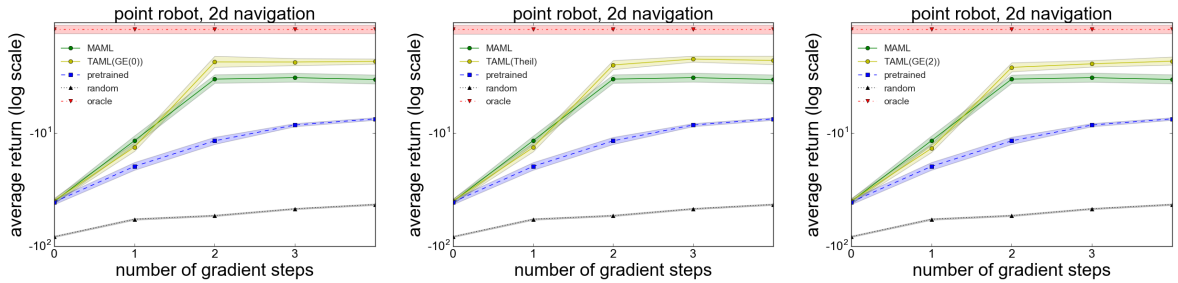
We also compare the performance of TAML when applied to Meta-Learn LSTM [129]. For this experiment, we added dropout after the last convolution layer and use leaky ReLU instead of ReLU non-linearity. In each iteration, We sample 5 datasets where each dataset is $\{\mathcal{D}_{train}, \mathcal{D}_{test}\}$ from $\mathscr{D}_{meta-train}$, and then calculate loss for each test set $\mathcal{D}_{test}$ of the dataset using the initial parameter of Meta-Learner. We optimize the parameters of the Meta-learner based on the both classification loss and TAML based inequality measure. We report the result in table 3.3. For both 1-shot and 5-shot experiment, we outperform Meta-Learn LSTM and achieve almost more than 3% accuracy

on both the settings. This shows that TAML can be applicable to any meta-learning algorithm.

Figure 3.1 shows the curve of validation accuracies of our entropy approach on the left panel and Theil based approach on right panel versus MAML for Mini-Imagenet 5-way 1-shot at gradient step 5. It can be seen that our both approaches achieve much better validation accuracy as compared to MAML meaning TAML achieves much better initialization point.

3.2.1.2 Analyses

Entropy based approach performs better than the inequality based approach. For 5- way Omniglot, there is negligible difference between the entropy based approach and inequality based approach. For 1 shot 5-way MiniImagenet experiment, entropy based TAML still beats inequality based TAML for MAML algorithms. VL based TAML has negligible improvement as compare to entropy based TAML because it uses Meta-SGD algorithm. When it uses MAML, its performance is lower than the entropy based approach. Every inequality has some properties as mentioned in section 3.1.3.1. Some of the inequalities are more sensitive to upper part of the distribution means it is more sensitive towards those tasks which have higher loss value and some of the inequalities are sensitive towards changes to those tasks which have lower loss values. The idea is to increase the uncertainty of the initial model on different tasks. Theil inequality is a part of larger family of GE. When alpha is 1 in equation 3.3, it becomes Theil Index. Moreover, As we can see from Table 3.1 3.2 3.3, VL, GE(0) and Thiel perform better than GE(2). For Omniglot 5-way experiment, the margin is negligible because MAML already achieved 99% accuracy on 1 shot and 99.9% on 5-shot. Atkinson index can also be derived from generalized entropy index family by setting epsilon = 1 - alpha as mentioned in equation 3.4 and 3.3. The high epsilon corresponds to GE index with small alpha means it becomes sensitive to lower end of the distribution.



(a) GE(0) (b) Theil (c) GE(2)

Figure 3.2: Results on 2D Navigation task.

3.2.2 Reinforcement Learning

In reinforcement learning, the goal is to learn the optimal policy given fewer trajectories or experiences. A reinforcement learning task $\mathcal{T}_i$ is defined as Markov Decision Process that consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, the reward function $\mathcal{R}$, and state-transition probabilities $q_i(x_{t+1}|x_t, a_t)$ where a_t is the action at time step t [118, 149, 151, 105, 103]. In our experiments, we are using the same settings as proposed in [41] where we are sampling trajectories using policy f_θ. The loss function used is the negative of the expectation of the sum of the rewards,

$$\mathcal{L}_{\mathcal{T}_i} = -\mathbb{E}_{a_t \sim f_\theta, x_t, q_{\mathcal{T}_i}} \left(\sum_{t=1}^{T} \mathcal{R}_i(x_t, a_t) \right) \tag{3.7}$$

Experiments were performed using rllab suite [36]. Vanilla policy gradient [176] is used to for inner gradient updates while trust region policy optimizer (TRPO) [145] is used as meta-optimizer. The algorithm is the same as mentioned in algorithm 1 with the only difference bing that trajectories were sampled instead of images.

For reinforcement learning experiment, we evaluate TAML on a 2D navigation task. The policy network that was used in performing this task is identical to the policy network that was used in

[41] for a fair comparison, which is a three-layered network using ReLU while setting the step size $\alpha = 0.1$. The experiment consists an agent moving in two-dimensional environment and the goal of the agent is to reach the goal state that is randomly sampled from a unit square. For evaluation purposes, we compare the results of TAML with MAML, oracle policy, conventional pre-training and random initialization. Our results have shown that GE(0), Theil, and GE(2) TAML perform on-par with MAML after 2 gradient steps but start to outperform it afterwards as shown in figure 3.2.

CHAPTER 4: LONG-TAILED VISUAL RECOGNITION

Big curated datasets, deep learning, and unprecedented computing power are often referred to as the three pillars of recent advances in visual recognition [83, 133, 106]. As we continue to build the big-dataset pillar, however, the power law emerges as an inevitable challenge. Object frequency in the real world often exhibits a long-tailed distribution where a small number of classes dominate, such as plants and animals [164, 1], landmarks around the globe [123], and common and uncommon objects in contexts [100, 61].

In this chapter, we propose to investigate long-tailed visual recognition from a domain adaptation point of view[1]. The long-tail challenge is essentially a mismatch problem between datasets with long-tailed class distributions seen by a machine learning model and our expectation of the model to perform well on all classes (and not bias toward the head classes). Conventional visual recognition methods, for instance, training neural networks by a cross-entropy loss, overly fit the dominant classes and fail in the underrepresented tail classes as they implicitly assume that the test sets are drawn i.i.d. from the same underlying distribution as the long-tailed training set. Domain adaptation explicitly breaks the assumption [152, 138, 52]. It discloses the inference-time data or distribution (*target domain*) to the machine learning models when they learn from the training data (*source domain*).

Denote by $P_s(x, y)$ and $P_t(x, y)$ the distributions of a source domain and a target domain, respectively, where x and y are respectively an instance and its class label. In long-tailed visual recognition, the marginal class distribution $P_s(y)$ of the source domain is long-tailed, and yet the

[1]This chapter's material has been accepted in 2020 in Conference on Computer Vision and Pattern Recognition (CVPR 2020), "Rethinking Class-Balanced Methods for Long-Tailed Visual Recognition from a Domain Adaptation Perspective" authored by Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong.

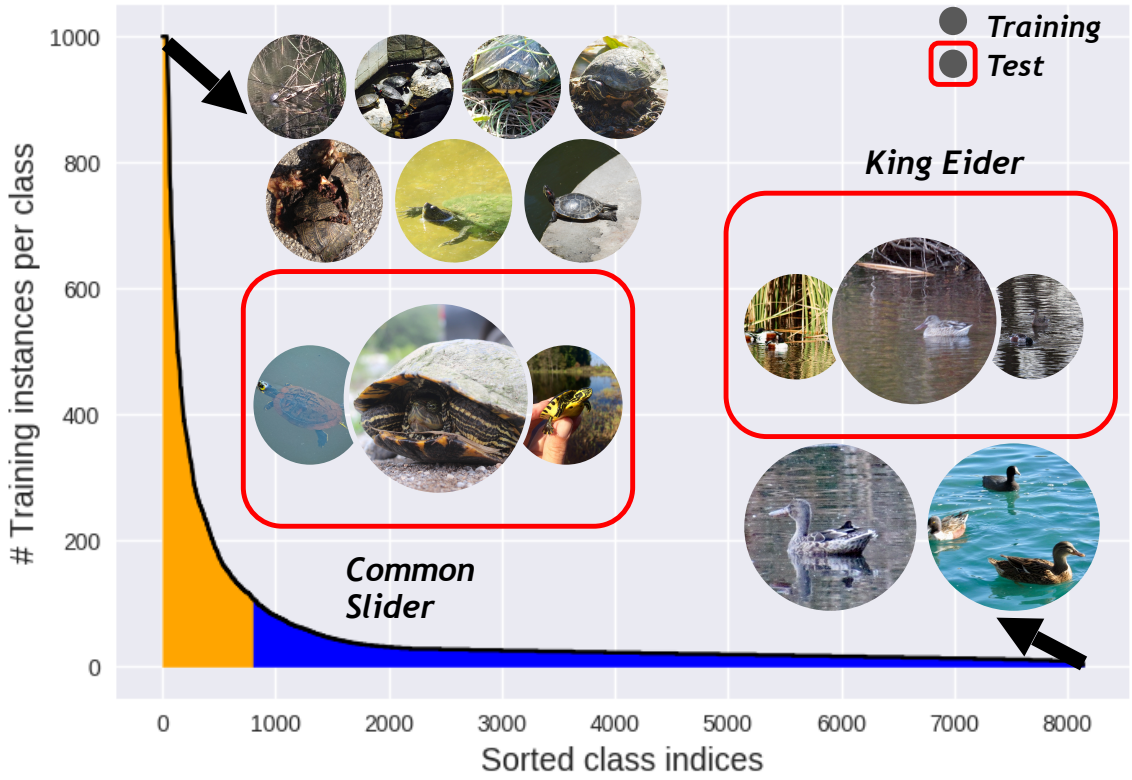


Figure 4.1: The training set of iNaturalist 2018 exhibits a long-tailed class distribution [1]. We connect domain adaptation with the mismatch between the long-tailed training set and our expectation of the trained classifier to perform equally well in all classes. We also view the prevalent class-balanced methods in long-tailed classification as the target shift in domain adaptation, i.e., $P_s(y) \neq P_t(y)$ and $P_s(x|y) = P_t(x|y)$, where P_s and P_t are respectively the distributions of the source domain and the target domain, and x and y respectively stand for the input and output of a classifier. We contend that the second part of the target shift assumption does not hold for tail classes, e.g., $P_s(x|\textit{King Eider}) \neq P_t(x|\textit{King Eider})$, because the limited training images of *King Eider* cannot well represent the data at inference time.

class distribution $P_t(y)$ of the target domain is more balanced, e.g., a uniform distribution.

In generic domain adaptation, there could be multiple causes of mismatches between two domains. Covariate shift [152] causes domain discrepancy on the marginal distribution of input, i.e., $P_s(x) \neq P_t(x)$, but often maintains the same predictive function across the domains, i.e., $P_s(y|x) = P_t(y|x)$. Under the target-shift cause [186], the domains differ only by the class distri-

butions, i.e., $P_s(y) \neq P_t(y)$ and $P_s(x|y) = P_t(x|y)$, partially explaining the rationale of designing class-balanced weights to tackle the long-tail challenge [25, 110, 38, 194, 116, 24, 12, 33, 111, 39].

These class-balanced methods enable the tail classes to play a bigger role than their sizes suggest in determining the model's decision boundaries. The class-wise weights are inversely related to the class sizes [66, 116, 110]. Alternatively, one can derive these weights from the cost of misclassifying an example of one class to another [38, 194]. Cui et al. proposed an interesting weighting scheme by counting the "effective number" of examples per class [25]. Finally, over/under-sampling head/tail classes [24, 111, 12, 33, 39] effectively belongs to the same family as the class-balanced weights, although they lead to practically different training algorithms. Chapter 2 reviews other methods for coping with the long-tail challenge.

One the one hand, the plethora of works reviewed above indicate that the target shift, i.e., $P_s(y) \neq P_t(y)$ and $P_s(x|y) = P_t(x|y)$, is generally a reasonable assumption based on which one can design effective algorithms for learning unbiased models from a training set with a long-tailed class distribution. On the other hand, however, our intuition challenges the second part of the target shift assumption; in other words, $P_s(x|y) = P_t(x|y)$ may not hold. While a head class (e.g., *Dog*) of the training set could contain abundant and diverse examples that well represent the expected data at inference time, the tail classes (e.g., *King Eider*) are often short of representative training examples. As a result, training examples drawn from the conditional distribution $P_s(x|\textit{Dog})$ of the source domain can probably well approximate the conditional distribution $P_t(x|\textit{Dog})$ of the target domain, but the discrepancy between the conditional distributions $P_s(x|\textit{King Eider})$ and $P_t(x|\textit{King Eider})$ of the two domains is likely big because it is hard to collect training examples for *King Eider* (cf. Figure 4.1).

To this end, we propose to augment the class-balanced learning by relaxing the assumption that the source and target domains share the same conditional distributions $P_s(x|y)$ and $P_t(x|y)$. By ex-

plicitly accounting for the differences between them, we arrive at a two-component weight [73] for each training example. The first part is inherited from the classic class-wise weighting, carrying on its effectiveness in various applications. The second part corresponds to the conditional distributions, and we estimate it by the meta-learning framework of learning to re-weight examples [132]. We make two critical improvements over this framework. One is that we can initialize the weights close to the optima because we have substantial prior knowledge about the two-component weights as a result of our analysis of the long-tailed problem. The other is that we remove two constraints from the framework such that the search space is big enough to cover the optima with a bigger chance.

We conduct extensive experiments on several datasets, including both long-tailed CIFAR [82], ImageNet [27], and Places-2 [192], which are artificially made long-tailed [25, 104], and iNaturalist 2017 and 2018 [164, 1], which are long-tailed by nature. We test our approach with three different losses (cross-entropy, focal loss [101], and a label-distribution-aware margin loss [14]). Results validate that our two-component weighting [73] is advantageous over the class-balanced methods.

4.1 Class Balancing as Domain Adaptation

In this section, we present a detailed analysis of the class-balanced methods [66, 110, 25, 24, 111] for long-tailed visual recognition from the domain adaptation point of view.

Suppose we have a training set (source domain) $\{(x_i, y_i)\}_{i=1}^n$ drawn i.i.d. from a long-tailed distribution $P_s(x, y)$ — more precisely, the marginal distribution $P_s(y)$ of classes are heavy-tailed because, in visual recognition, it is often difficult to collect examples for rare classes. Nonetheless,

we expect to learn a visual recognition model to make as few mistakes as possible on all classes:

$$\texttt{error} = \mathbb{E}_{P_t(x,y)} L(f(x;\theta), y), \tag{4.1}$$

where we desire a target domain $P_t(x, y)$ whose marginal class distribution $P_t(y)$ is more balanced (e.g., a uniform distribution) at the inference time, $f(\cdot;\theta)$ is the recognition model parameterized by θ, and $L(\cdot,\cdot)$ is a 0-1 loss. We abuse the notation $L(\cdot,\cdot)$ a little and let it be a differentiable surrogate loss (i.e., cross-entropy) during training.

Next, we apply the importance sampling trick to connect the expected `error` with the long-tailed source domain,

$$\begin{aligned}
\texttt{error} &= \mathbb{E}_{P_t(x,y)} L(f(x;\theta), y) && (4.2)\\
&= \mathbb{E}_{P_s(x,y)} L(f(x;\theta), y) P_t(x,y)/P_s(x,y) && (4.3)\\
&= \mathbb{E}_{P_s(x,y)} L(f(x;\theta), y) \frac{P_t(y)P_t(x|y)}{P_s(y)P_s(x|y)} && (4.4)\\
&:= \mathbb{E}_{P_s(x,y)} L(f(x;\theta), y) w_y (1 + \tilde{\epsilon}_{x,y}), && (4.5)
\end{aligned}$$

where $w_y = P_t(y)/P_s(y)$ and $\tilde{\epsilon}_{x,y} = P_t(x|y)/P_s(x|y) - 1$.

Existing class-balanced methods focus on how to determine the class-wise weights $\{w_y\}$ and result in the following objective function for training,

$$\min_{\theta} \quad \frac{1}{n} \sum_{i=1}^{n} w_{y_i} L(f(x_i;\theta), y_i), \tag{4.6}$$

which approximates the expected inference `error` (eq. (4.5)) by assuming $\tilde{\epsilon}_{x,y} = 0$ or, in other words, by assuming $P_s(x|y) = P_t(x|y)$ for any class y. This assumption is referred to as target shift [186] in domain adaptation.

We contend that the assumption of a shared conditional distribution, $P_s(x|y) = P_t(x|y)$, does not hold in general, especially for the tail classes. One may easily compile a representative training set for *Dog*, but not for *King Eider*. We propose to explicitly model the difference $\tilde{\epsilon}_{x,y}$ between the source and target conditional distributions and arrive at an improved algorithm upon the class-balanced methods.

4.2 Modeling the Conditional Differences

For simplicity, we introduce a conditional weight $\epsilon_{x,y} := w_y \tilde{\epsilon}_{x,y}$ and re-write the expected inference `error` as

$$\texttt{error} = \mathbb{E}_{P_s(x,y)} L(f(x;\theta), y)(w_y + \epsilon_{x,y}) \tag{4.7}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} (w_{y_i} + \epsilon_i) L(f(x_i;\theta), y_i), \tag{4.8}$$

where the last term is an unbiased estimation of the `error`. Notably, we do not make the assumption that the conditional distributions of the source and target domains are the same, i.e., we allow $P_s(x|y) \neq P_t(x|y)$ and $\epsilon_i \neq 0$. Hence, the weight for each training example consists of two parts. One component is the class-wise weight w_{y_i}, and the other is the conditional weight ϵ_i. We need to estimate both components to derive a practical algorithm from eq. (4.8) because the underlying distributions of data are unknown — although we believe the class distribution of the training set must be long-tailed.

4.2.1 *Estimating the Class-wise Weights* $\{w_y\}$

We let the class-wise weights resemble the empirically successful design in the literature. In particular, we estimate them by the recently proposed "effective numbers" [25]. Supposing there are n_y training examples for the y-th class, we have $w_y \approx (1-\beta)/(1-\beta^{n_y})$ where $\beta \in [0, 1)$ is a hyper-parameter with the recommended value $\beta = (n-1)/n$, and n is the number of training examples.

4.2.2 *Meta-learning the Conditional Weights* $\{\epsilon_i\}$

We estimate the conditional weights by customizing a meta-learning framework [132]. We describe our approach below and then discuss two critical differences from the original framework in Section 4.2.3.

The main idea is to hold out a balanced development set D from the training set and use it to guide the search for the conditional weights that give rise to the best-performing recognition model $f(\cdot;\theta)$ on the development set. Denote by T the remaining training data. We seek the conditional weights $\boldsymbol{\epsilon} := \{\epsilon_i\}$ by solving the following problem,

$$\min_{\boldsymbol{\epsilon}} \quad \frac{1}{|D|}\sum_{i\in D} L(f(x_i;\theta^*(\boldsymbol{\epsilon})), y_i) \text{ with} \tag{4.9}$$

$$\theta^*(\boldsymbol{\epsilon}) \leftarrow \arg\min_{\theta} \frac{1}{|T|}\sum_{i\in T}(w_{y_i}+\epsilon_i)L(f(x_i;\theta), y_i) \tag{4.10}$$

where we do *not* weigh the losses over the development set which is already balanced. Essentially, the problem above searches for the optimal conditional weights such that, after we learn a recognition model $f(\cdot;\theta)$ by minimizing the `error` estimation (eqs (4.10) and (4.8)), the model performs the best on the development set (eq. (4.9)).

It would be daunting to solve the problem above by brute-force search, e.g., iterating all the possible sets $\{\boldsymbol{\epsilon}\}$ of conditional weights. Even if we can, it is computationally prohibitive to train for each set of weights a recognition model $f(\cdot; \theta^*(\boldsymbol{\epsilon}))$ and then find out the best model from all.

Instead, we modify the meta-learning framework [132] and search for the conditional weights in a greedy manner. It interleaves the quest for the weights $\boldsymbol{\epsilon}$ with the updates to the model parameters θ, given current time step t,

$$\begin{aligned}
\tilde{\theta}^{t+1}(\boldsymbol{\epsilon}^t) &\leftarrow \theta^t - \eta \frac{\partial \sum_{i \in T} (w_{y_i} + \epsilon_i^t) L(f(x_i; \theta^t), y_i)}{\partial \theta} \\
\boldsymbol{\epsilon}^{t+1} &\leftarrow \boldsymbol{\epsilon}^t - \tau \frac{\partial \sum_{i \in D} L(f(x_i; \tilde{\theta}^{t+1}(\boldsymbol{\epsilon}^t)), y_i)}{\partial \boldsymbol{\epsilon}} \\
\theta^{t+1} &\leftarrow \theta^t - \eta \frac{\partial \sum_{i \in T} (w_{y_i} + \epsilon_i^{t+1}) L(f(x_i; \theta^t), y_i)}{\partial \theta}.
\end{aligned}$$

The first equation tries a one-step gradient descent for θ^t using the losses weighted by the current conditional weights $\boldsymbol{\epsilon}^t$ (plus the class-wise weights). The updated model parameters $\tilde{\theta}^{t+1}(\boldsymbol{\epsilon}^t)$ are then scrutinized on the balanced development set D, which updates the conditional weights by one step. The updated weights $\boldsymbol{\epsilon}^{t+1}$ are better than the old ones, meaning that the model parameters θ^{t+1} returned by the last equation should give rise to smaller recognition error on the development set than $\tilde{\theta}^{t+1}$ do. Starting from θ^{t+1} and $\boldsymbol{\epsilon}^{t+1}$, we then move on to the next round of updates. We present our overall algorithm in the next section.

4.2.3 Overall Algorithm and Discussion

We are ready to present **Algorithm** 5 for long-tailed visual recognition. The discussions in the previous sections consider all the training examples in a batch setting. **Algorithm** 5 customizes it into a stochastic setting so that we can easily integrate it with deep learning frameworks.

Algorithm 3 Meta-learning for long-tailed recognition

Require: Training set T, balanced development set D
Require: Class-wise weights $\{w_y\}$ estimated by using [25]
Require: Learning rates η and τ, stopping steps t_1 and t_2
Require: Initial parameters θ of the recognition network

1: **for** $t = 1, 2, \cdots, t_1$ **do**
2: Sample a mini-batch B from the training set T
3: Compute loss $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} L(f(x_i; \theta), y_i)$
4: Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_B$
5: **end for**
6: **for** $t = t_1 + 1, \cdots, t_1 + t_2$ **do**
7: Sample a mini-batch B from the training set T
8: Set $\epsilon_i \leftarrow 0, \forall i \in B$, and denote by $\boldsymbol{\epsilon} := \{\epsilon_i, i \in B\}$
9: Compute $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} (w_{y_i} + \epsilon_i) L(f(x_i; \theta), y_i)$
10: Update $\tilde{\theta}(\boldsymbol{\epsilon}) \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_B$
11: Sample B_d from the balanced development set D
12: Compute $\mathcal{L}_{B_d} = \frac{1}{|B_d|} \sum_{i \in B_d} L(f(x_i; \tilde{\theta}(\boldsymbol{\epsilon})), y_i)$
13: Update $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} - \tau \nabla_\epsilon \mathcal{L}_{B_d}$
14: Compute new loss with the updated $\boldsymbol{\epsilon}$
$\tilde{\mathcal{L}}_B = \frac{1}{|B|} \sum_{i \in B} (w_{y_i} + \epsilon_i) L(f(x_i; \theta), y_i)$
15: Update $\theta \leftarrow \theta - \eta \nabla_\theta \tilde{\mathcal{L}}_B$
16: **end for**

There are two learning stages in the algorithm. In the first stage (lines 1–5), we train the neural recognition network $f(\cdot; \theta)$ by using the conventional cross-entropy loss over the long-tailed training set. The second stage (lines 6–16) meta-learns the conditional weights by resorting to a balanced development set and meanwhile continues to update the recognition model. We highlight the part for updating the conditional weights in lines 11–13.

It is worth noting some seemingly small and yet fundamental differences between our algorithm and the learning to re-weight (L2RW) method [132]. Conceptually, while we share the same meta-learning framework as L2RW, both the class-wise weight, $w_y = P_t(y)/P_s(y)$, and the conditional weight, $\epsilon_{x,y} = w_y \tilde{\epsilon}_{x,y} = P_t(y)/P_s(y)\big(P_t(x|y)/P_s(x|y) - 1\big)$, have principled interpretations as oppose to a general per-example weight in L2RW. We will explore other machine learning frameworks (e.g., [11, 157]) to learn the conditional weights in future work, but the interpretations of

them remain the same.

Algorithmically, unlike L2RW, we employ two-component weights, estimate the class-wise components by a different method [25], do *not* clip negative weights $\{\epsilon_i\}$ to 0, and do *not* normalize them such that they sum to 1 within a mini-batch. The clipping and normalization operations in L2RW unexpectedly reduce the search space of the weights, and the normalization is especially troublesome as it depends on the mini-batch size. Hence, if the optimal weights actually lie outside of the reduced search space, there is no chance to hit the optima by L2RW. In contrast, our algorithm searches for each conditional weight ϵ_i in the full real space. One may wonder whether or not our total effective weight, $w_{y_i} + \epsilon_i$, could become negative. Careful investigation reveals that it never goes below 0 in our experiments, likely due to that the good initialization (as explained below) to the conditional weights makes it unnecessary to update the weights too wildly by line 13 in **Algorithm** 5.

Computationally, we provide proper initialization to both the conditional weights, by $\epsilon_i \leftarrow 0$ (line 8), and the model parameters θ of the recognition network, by pre-training the network with a vanilla cross-entropy loss (lines 1–5). As a result, our algorithm is more stable than L2RW (cf. Section 4.3.1). Note that 0 is a reasonable *a priori* value for the conditional weights thanks to the promising results obtained by existing class-balanced methods. Those methods assume that the discrepancy is as small as 0 between the conditional distributions of the source and target domains, meaning that $P_t(x|y)/P_s(x|y) - 1$ is close to 0, so are the conditional weights $\{\epsilon_i\}$. Hence, our approach should perform at worst the same as the class-balanced method [25] by initializing the conditional weights to 0 (and the class-wise weights by [25]).

4.3 Experiments

We evaluate and ablate our approach on six datasets of various scales, ranging from the manually created long-tailed CIFAR-10 and CIFAR-100 [25], ImageNet-LT, and Places-LT [104], to the naturally long-tailed iNaturalist 2017 [164] and 2018 [1]. Following [25], we define the *imbalance factor (IF)* of a dataset as the class size of the first head class divided by the size of the last tail class.

Long-Tailed CIFAR (CIFAR-LT): The original CIFAR-10 (CIFAR-100) dataset contains 50,000 training images and 10,000 test images of size 32x32 uniformly falling into 10 (100) classes [82]. Cui et al. [25] created long-tailed versions by randomly removing training examples. In particular, the number of examples dropped from the y-th class is $n_y \mu^y$, where n_y is the original number of training examples in the class and $\mu \in (0, 1)$. By varying μ, we arrive at six training sets, respectively, with the imbalance factors (IFs) of 200, 100, 50, 20, 10, and 1, where IF=1 corresponds to the original datasets. We do not change the test sets, which are balanced. We randomly select ten training images per class as our development set D.

ImageNet-LT: In spirit similar to the long-tailed CIFAR datasets, Liu et al. [104] introduced a long-tailed version of ImageNet-2012 [27] called ImageNet-LT. It is created by firstly sampling the class sizes from a Pareto distribution with the power value $\alpha = 6$, followed by sampling the corresponding number of images for each class. The resultant dataset has 115.8K training images in 1,000 classes, and its imbalance factor is 1280/5. The authors have also provided a validation set with 20 images per class, from which we sample ten images to construct our development set D. The original balanced ImageNet-2012 validation set is used as the test set (50 images per class).

Places-LT: Liu et al. [104] have also created a Places-LT dataset by sampling from Places-2 [192] using the same strategy as above. It contains 62.5K training images from 365 classes with

Table 4.1: Overview of the six datasets used in our experiments. (IF stands for the imbalance factor)

Dataset	# Classes	IF	# Train. img.	Tail class size	Head class size	# Val. img.	# Test img.
CIFAR-LT-10	10	1.0–200.0	50,000–11,203	500–25	5,000	–	10,000
CIFAR-LT-100	100	1.0–200.0	50,000–9,502	500–2	500	–	10,000
iNat 2017	5,089	435.4	579,184	9	3,919	95,986	–
iNat 2018	8,142	500.0	437,513	2	1,000	24,426	–
ImageNet-LT	1,000	256.0	115,846	5	1,280	20,000	50,000
Places-LT	365	996.0	62,500	5	4,980	7,300	36,500

an imbalance factor 4980/5. This large imbalance factor indicates that it is more challenging than ImageNet-LT. Places-LT has 20 (100) validation (test) images per class. Our development set D contains ten images per class randomly selected from the validation set.

iNaturalist (iNat) 2017 and 2018: The iNat 2017 [164] and 2018 [1] are real-world fine-grained visual recognition datasets that naturally exhibit long-tailed class distributions. iNat 2017 (2018) consists of 579,184 (435,713) training images in 5,089 (8,142) classes, and its imbalance factor is 3919/9 (1000/2). We use the official validation sets to test our approach. We select five (two) images per class from the training set of iNat 2017 (2018) for the development set.

Table 4.1 gives an overview of the six datasets used in the following experiments.

For **evaluation metrics**, as the test sets are all balanced, we simply use the top-k error as the evaluation metric. We report results for $k = 1, 3, 5$.

4.3.1 Object Recognition with CIFAR-LT

We run both comparison experiments and ablation studies with CIFAR-LT-10 and CIFAR-LT-100. We use ResNet-32 [63] in the experiments. We compare our approach to the following competing

ones.

- **Cross-entropy training.** This is the baseline that trains ResNet-32 using the vanilla cross-entropy loss.
- **Class-balanced loss [25].** It weighs the conventional losses by class-wise weights, which are estimated based on effective numbers. We apply this class-balanced weighting to three different losses: cross-entropy, the focal loss [101], and the recently proposed label-distribution-aware margin loss [14].
- **Focal loss [101].** The focal loss can be understood as a smooth version of hard example mining. It does not directly tackle the long-tailed recognition problem. However, it can penalize the examples of tail classes more than those of the head classes if the network is biased toward the head classes during training.
- **Label-distribution-aware margin loss [14].** It dynamically tunes the margins between classes according to their degrees of dominance in the training set.
- **Class-balanced fine-tuning [24].** The main idea is to first train the neural network with the whole imbalanced training set and then fine-tune it on a balanced subset of the training set.
- **Learning to re-weight (L2RW) [132].** It weighs training examples by meta-learning. Please see Section 4.2.3 for more discussions about L2RW and our approach.
- **Meta-weight net [153].** Similarly to L2RW, it also weighs examples by a meta-learning method except that it regresses the weights by a multilayer perceptron.

4.3.1.1 Implementation details

For the first two baselines, we use the code of [25] to set the learning rates and other hyperparameters. We train the L2RW model using an initial learning rate of 1e-3. We decay the learning rate

Table 4.2: Test top-1 errors (%) of ResNet-32 on CIFAR-LT-10 under different imbalance settings. * indicates results reported in [153].

Imbalance factor	200	100	50	20	10	1
Cross-entropy training	34.32	29.64	25.19	17.77	13.61	7.53/7.11*
Class-balanced cross-entropy loss [25]	31.11	27.63	21.95	15.64	13.23	7.53/7.11*
Class-balanced fine-tuning [24]	33.76	28.66	22.56	16.78	16.83	7.08
Class-balanced fine-tuning [24]*	33.92	28.67	22.58	13.73	13.58	**6.77**
L2RW [132]	33.75	27.77	23.55	18.65	17.88	11.60
L2RW [132]*	33.49	25.84	21.07	16.90	14.81	10.75
Meta-weight net [153]	32.8	26.43	20.9	15.55	12.45	7.19
Ours with cross-entropy loss	**29.34**	**23.59**	**19.49**	**13.54**	**11.15**	7.21
Focal loss [101]	34.71	29.62	23.29	17.24	13.34	6.97
Class-balanced focal Loss [25]	31.85	25.43	20.78	16.22	12.52	6.97
Ours with focal Loss	**25.57**	**21.1**	**17.12**	**13.9**	**11.63**	7.19
LDAM loss [14] (results reported in paper)	-	26.65	-	-	13.04	11.37
LDAM-DRW [14] (results reported in paper)	-	22.97	-	-	**11.84**	-
Ours with LDAM loss	**22.77**	**20.0**	**17.77**	**15.63**	12.6	**10.29**

by 0.01 at the 160th and 180th epochs. For our approach, we use an initial learning rate of 0.1 and then also decay the learning rate at the 160th and 180th epochs by 0.01. The batch size is 100 for all experiments. We train all models on a single GPU using the stochastic gradient descent with momentum.

4.3.1.2 Results

Table 4.2 shows the classification errors of ResNet-32 on the long-tailed CIFAR-10 with different imbalance factors. We group the competing methods into three sessions according to which basic loss they use (cross-entropy, focal [101], or LDAM [14]). We test our approach with all three losses. We can see that our method outperforms the competing ones in each session by notable margins. Although the focal loss and the LDAM loss already have the capacity of mitigating the long-tailed issue, respectively, by penalizing hard examples and by distribution-aware margins, our method can further boost their performances. In general, the advantages of our approach over

Table 4.3: Test top-1 errors (%) of ResNet-32 on CIFAR-LT-100 under different imbalance settings. * indicates results reported in [153].

Imbalance factor	200	100	50	20	10	1
Cross-entropy training	65.16	61.68	56.15	48.86	44.29	29.50
Class-balanced cross-entropy loss [25]	64.30	61.44	55.45	48.47	42.88	29.50
Class-balanced fine-tuning [24]	61.34	58.5	53.78	47.70	42.43	29.37
Class-balanced fine-tuning [24]*	61.78	58.17	53.60	47.89	42.56	29.28
L2RW [132]	67.00	61.10	56.83	49.25	47.88	36.42
L2RW [132]*	66.62	59.77	55.56	48.36	46.27	35.89
Meta-weight net [153]	63.38	58.39	54.34	46.96	41.09	29.9
Ours with cross-entropy loss	**60.69**	**56.65**	**51.47**	**44.38**	**40.42**	**28.14**
Focal Loss [101]	64.38	61.59	55.68	48.05	44.22	**28.85**
Class-balanced focal Loss [25]	63.77	60.40	54.79	47.41	42.01	**28.85**
Ours with focal loss	**60.66**	**55.3**	**49.92**	**44.27**	**40.41**	29.15
LDAM Loss [14] (results reported in paper)	-	60.40	-	-	43.09	-
LDAM-DRW [14] (results reported in paper)	-	57.96	-	-	**41.29**	-
Ours with LDAM loss	**60.47**	**55.92**	**50.84**	**47.62**	42.0	-

existing ones become more significant as the imbalance factor increases. When the dataset is balanced (the last column), our approach does not hurt the performance of vanilla losses compared to L2RW. We can draw about the same observations as above for the long-tailed CIFAR-100 from Table 4.3. Finally, we further validate our approach by running each setting 5 times with different random seeds on long-tailed CIFAR-10. Table 4.7 shows the mean top-1 errors (%) and the standard deviations under the imbalance factors of 200, 100, and 50. We can see that the mean error rates are consistent with the results provided in Table 4.2.

4.3.1.3 Where does our approach work?

Figure 4.2 presents three confusion matrices respectively by the models of the cross-entropy training, L2RW, and our method on CIFAR-LT-10. The imbalance factor is 200. Compared with the cross-entropy model, L2RW improves the accuracies on the tail classes and yet sacrifices the

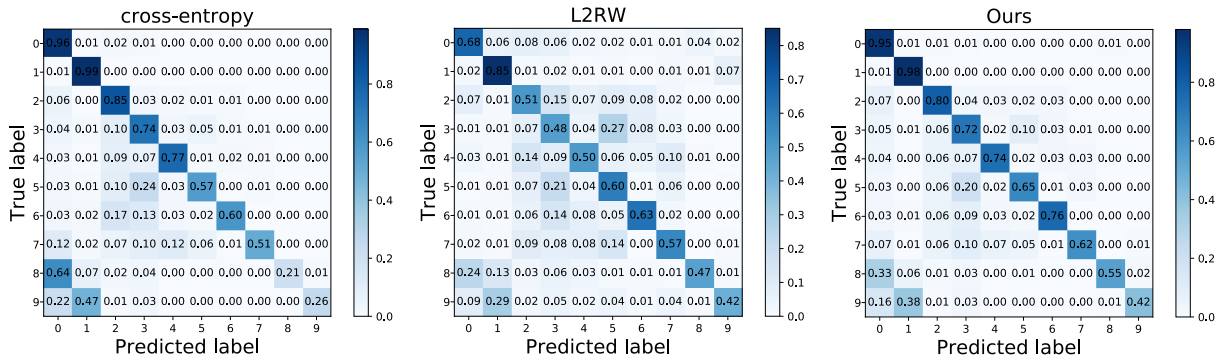


Figure 4.2: Confusion matrices by the cross-entropy training, L2RW, and our method on CIFAR-LT-10 (the imbalance factor is 200).

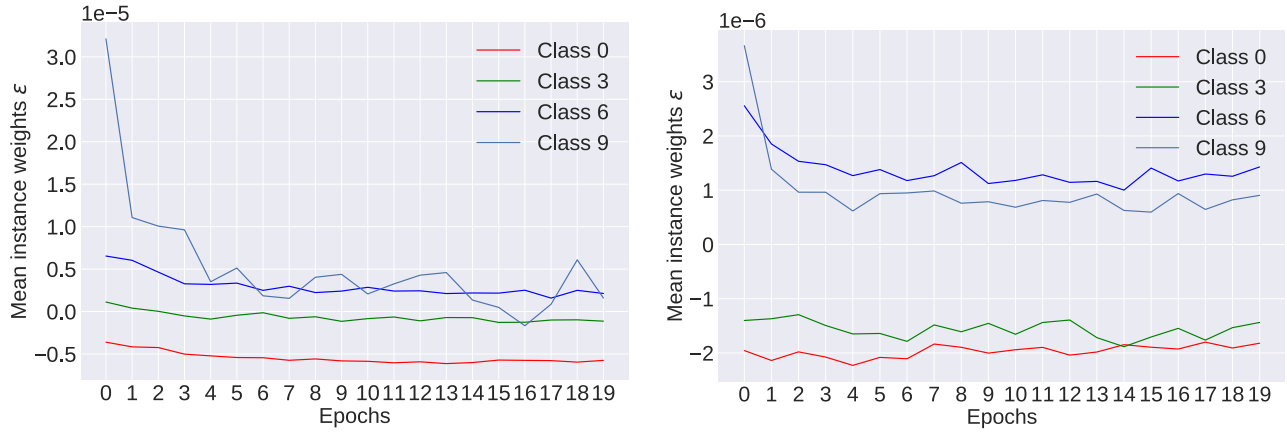


Figure 4.3: Mean conditional weights $\{\epsilon_i\}$ within each class vs. training epochs on CIFAR-LT-10 (left: IF = 100; right: IF = 10).

accuracies for the head classes. In contrast, ours maintains about the same performance as the cross-entropy model on the head classes and meanwhile significantly improves the accuracies for the last five tail classes.

Table 4.4: Ablation study of our approach by using the cross-entropy loss on CIFAR-LT-10. The results are test top-1 errors%.

Imbalance factor	100	50	20
L2RW [132]	27.77	23.55	18.65
L2RW, pre-training	25.96	22.04	15.67
L2RW, pre-training, init. by w_y	26.26	22.50	17.44
L2RW, pre-training, $w_{y_i} + \epsilon_i$	24.54	20.47	14.38
Ours	**23.59**	**19.49**	**13.54**
Ours updating w_y	25.42	20.13	15.62
Class-balanced [25]	27.63	21.95	15.64

4.3.1.4 What are the learned conditional weights?

We are interested in examining the conditional weights $\{\epsilon_i\}$ for each class throughout the training. For a visualization purpose, we average them within each class. Figure 4.3 demonstrates how they change over the last 20 epochs for the 1st, 4th, 7th, and 10th classes of CIFAR-LT-10. The two panels correspond to the imbalance factors of 100 and 10, respectively. Interestingly, the learned conditional weights of the tail classes are more prominent than those of the head classes in most epochs. Moreover, the conditional weights of the two head classes (the 1st and 4th) are even below 0 at certain epochs. Such results verify our intuition that the scarce training examples of the tail classes deserve more attention in training to make the neural network perform in a balanced fashion at the test phase.

4.3.1.5 Ablation study: ours vs. L2RW

Our overall algorithm differs from L2RW mainly in four ways: 1) pre-training the network, 2) initializing the weights by *a priori* knowledge, 3) two-component weights, and estimating the

class-wise components by a separate algorithm [25], and 4) no clipping or normalization of the weights. Table 4.4 examines these components by applying them one after another to L2RW. First, pre-training the neural network boosts the performance of the vanilla L2RW. Second, if we initialize the sample weights by our class-wise weights $\{w_y\}$, the errors increase a little probably because the clipping and normalization steps in L2RW require more careful initialization to the sample weights. Third, if we replace the sample weights by our two-component weights, we can bring the performance of L2RW closer to ours. Finally, after we remove the clipping and normalization, we arrive at our algorithm, which gives rise to the best results among all variations.

4.3.1.6 Ablation study: the two-component weights

By Table 4.4, we also highlight the importance of the two-component weights $\{w_{y_i} + \epsilon_i\}$ motivated from our domain adaptation point of view to the long-tailed visual recognition. First of all, they benefit L2RW (comparing "L2RW, pre-training, $w_{y_i} + \epsilon_i$" with "L2RW, pre-training" in Table 4.4). Besides, they are also vital for our approach. If we drop the class-wise weights, our results would be about the same as L2RW with pre-training. If we drop the conditional weights and meta-learn the class-wise weights (cf. "Ours updating w_y"), the errors become larger than our original algorithm. Nonetheless, the results are better than the class-balanced training (cf. last row in Table 4.4), implying that the learned class-wise weights give rise to better models than the effective-number-based class-wise weights [25].

4.3.2 Object Recognition with iNat 2017 and 2018

We use ResNet-50 [63] as the backbone network for the iNat 2017 and 2018 datasets. The networks are pre-trained on ImageNet for iNat 2017 and on ImageNet plus iNat 2017 for iNat 2018. We experiment with the mini-batch size of 64 and the learning rate of 0.01. We train all the models

Table 4.5: Classification errors on iNat 2017 and 2018. (*results reported in paper. CE=cross-entropy, CB=class-balanced)

Dataset	iNat 2017		iNat 2018	
Method	Top-1	Top-3/5	Top-1	Top-3/5
CE	43.49	26.60/21.00	36.20	19.40/15.85
CB CE [25]	42.59	25.92/20.60	34.69	19.22/15.83
Ours, CE	**40.62**	**23.70/18.40**	**32.45**	**18.02/13.83**
CB focal [25]*	41.92	–/20.92	38.88	–/18.97
LDAM [14]*	–	–	35.42	–/16.48
LDAM-drw*	–	–	32.00	–/14.82
cRT [78]*	–	–	34.8	–
cRT+epochs*	–	–	32.4	–

using the stochastic gradient descent with momentum. For the meta-learning stage of our approach, we switch to a small learning rate, 0.001.

Table 4.5 shows the results of our two-component weighting applied to the cross-entropy loss. We shrink the text size for iNat 2018 to signify that we advocate experiments with iNat *2017* instead because there are only three validation/test images per class in iNat 2018 (cf. Table 4.1). Our approach boosts the cross-entropy training by about 2% more than the class-balanced weighting [25] does. As we have reported similar effects for the focal loss and the LDAM loss on CIFAR-LT with extensive experiments, we do not run them on the large-scale iNat datasets to save computation costs. Nonetheless, we include the results reported in the literature of the focal loss, LDAM loss, and a classifier re-training method [78], which was published after we submitted the work to CVPR 2020. We reported more detailed comparison in table 4.10. As the experiment setups of the existing works vary by network initialization, the sampling strategy of mini-batches, losses, trainable layers of a network, etc., it is hard to have a fair comparison by the end results. Hence, besides their top-1 errors, we also report the experiment setups for each method. Our approach

Table 4.6: Classification errors on ImageNet-LT and Places-LT. (*reported in paper. CE=cross-entropy, CB=class-balanced)

Dataset	ImageNet-LT		Places-LT	
Method	Top-1	Top-3/5	Top-1	Top-3/5
CE	74.74	61.35/52.12	73.00	52.05/41.44
CB CE [25]	73.41	59.22/50.49	71.14	51.58/41.96
Ours, CE	**70.10**	**53.29/45.18**	**69.20**	**47.95/38.00**

outperforms the class-balanced weighting scheme for both the cross-entropy loss and the focal loss. Moreover, our results are on par with the best reported ones. Finally, we stress that almost all existing methods employ a class-balanced weighting or sampling strategy no matter what their main techniques are to tackle the long-tailed problem. Hence, given our consistent improvements over the class-balanced weighting, we expect the methods which have benefited from the class-balancing can gain further from our two-component weighting.

4.3.3 Experiments with ImageNet-LT and Places-LT

Following Liu et al.'s experiment setup [104], we employ ResNet-32 and ResNet-152 for the experiments on ImageNet-LT and Places-LT, respectively. For ImageNet-LT, we adopt an initial learning rate of 0.1 and decay it by 0.1 after every 35 epochs. For Places-LT, the initial learning rate is 0.01 and is decayed by 0.1 every 10 epochs. For our own approach, we switch from the cross-entropy training to the meta-learning stage when the first decay of the learning rate happens. The mini-batch size is 64, and the optimizer is stochastic gradient descent with momentum.

Table 4.7: Multiple runs of our approach by using the cross-entropy loss on CIFAR-LT-10. The results are top-1 errors% on the test sets.

Imbalance factor	200	100	50
Cross-entropy training	34.32	29.64	25.19
Class-balanced cross-entropy loss [25]	31.11	27.63	21.95
Class-balanced fine-tuning [24]	33.76	28.66	22.56
Class-balanced fine-tuning [24]*	33.92	28.67	22.58
L2RW [132]	33.75	27.77	23.55
L2RW [132]*	33.49	25.84	21.07
Meta-weight net [153]	32.8	26.43	20.9
Ours with cross-entropy loss	**29.32** $\pm$ **0.23**	**23.71** $\pm$ **0.22**	**19.45** $\pm$ **0.28**

4.3.3.1 Results

Table 4.6 shows that the class-balanced training improves the vanilla cross-entropy results, and our two-component weighting further boosts the results. We expect the same observation with the focal and LDAM losses. Finally, we find another improvement by updating the classification layers only in the meta-learning stage. We arrive at 62.90% top-1 error (39.86/29.87% top-3/5 error) on Places-LT, which is on par with 64.1% by OLTR [104] or 63.3% by cRT [78], while noting that our two-component weighting can be conveniently applied to both OLTR and cRT. Similar to iNaturalist, Table 4.8 and Table 4.9 shows the more detailed comparison on ImageNet-LT and Places-LT respectively.

Table 4.8: Test top-1 errors (%) of different methods on ImageNet-LT. * indicates the re-run results.

Methods	NN	Initialization	Sampling	Loss	Stage-1 Trainable Variables	Stage-2 Trainable Variables	Results
Vanilla Model	ResNet-10	No-pretrain	Class-Balanced	CE	All	-	80.0
Vanilla Model [101]	ResNet-10	No-pretrain	Class-Balanced	Focal	All	-	69.8
Vanilla Model	ResNet-10	No-pretrain	Class-Balanced	Lifted	All	-	69.2
Vanilla Model [189]	ResNet-10	No-pretrain	Class-Balanced	Range	All	-	69.3
Joint [78]	ResNet-10	No-pretrain	Class-Balanced	CE	All	All	65.2
NCM [78]	ResNet-10	No-pretrain	Class-Balanced	CE	All	Classifier layer	64.5
cRT [78]	ResNet-10	No-pretrain	Class-Balanced	CE	All	Classifier layer	**58.2**
τ-normalized [78]	ResNet-10	No-pretrain	Class-Balanced	CE	All	Classifier layer	59.4
OLTR* [104]	ResNet-10	No-pretrain	Class-Balanced	CE	All	All	65.6
OLTR [104]	ResNet-10	No-pretrain	Class-Balanced	CE	All	All	64.4
Ours	ResNet-10	No-pretrain	None	CE	All	Classifier layer	63.5
Ours	ResNet-10	No-pretrain	None	Focal	All	Classifier layer	63.3
Vanilla Model	ResNet-50	No-pretrain	None	CE	All	-	59.0
CB [25]	ResNet-50	No-pretrain	None	CE	All	-	58.2
Joint [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	All	58.4
NCM [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	55.7
cRT [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	52.7
τ-normalized [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	53.3
Ours	ResNet-50	No-pretrain	None	CE	All	Classifier layer	**52.0**

Table 4.9: Test top-1 errors (%) of different methods on Places-LT. * indicates the re-run results.

Methods	NN	Initialization	Sampling	Loss	Stage-1 Trainable Variables	Stage-2 Trainable Variables	Results
Vanilla Model	ResNet-152	ImageNet	Class-Balanced	CE	FC layers Last Block + FC	- -	72.1 69.7
Vanilla Model [101]	ResNet-152	ImageNet	Class-Balanced	Focal	FC layers Last Block + FC	- -	67.0 66.5
Vanilla Model	ResNet-152	ImageNet	Class-Balanced	Lifted	FC layers	-	64.8
Vanilla Model [189]	ResNet-152	ImageNet	Class-Balanced	Range	FC layers	-	64.9
Joint [78]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	Last block + FC	69.8
NCM [78]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	Classifier layer	63.7
cRT [78]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	Classifier layer	63.3
τ-normalized [78]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	Classifier layer	**62.1**
OLTR* [104]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	FC + memory	64.8
OLTR [104]	ResNet-152	ImageNet	Class-Balanced	CE	Last block + FC	FC + memory	64.1
Ours	ResNet-152	ImageNet	None	CE	Last block + FC	Classifier layer	62.9
Ours	ResNet-152	ImageNet	None	Focal	Last block + FC	Classifier layer	62.2

Table 4.10: Test top-1 errors (%) of different methods on iNaturalist 2018.

Methods	NN	Initialization	Sampling	Loss	Stage-1 Trainable Variables	Stage-2 Trainable Variables	Results
Vanilla Model	ResNet-50	No-pretrain	None	CE	All	-	42.9
Vanilla Model	ResNet-50	ImageNet+iNat'17	None	CE	All	-	36.2
LDAM [14]	ResNet-50	No-pretrain	None	LDAM	All	-	35.4
LDAM-DRW [14]	ResNet-50	No-pretrain	None	LDAM	All	-	**32.0**
CB [25]	ResNet-50	ImageNet+iNat'17	None	CE	All	-	34.7
CB [25]	ResNet-50	No-pretrain	None	Focal	All	-	38.9
Joint [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	All	38.3
NCM [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	41.8
cRT [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	34.8
τ-normalized [78]	ResNet-50	No-pretrain	Class-Balanced	CE	All	Classifier layer	34.4
Ours	ResNet-50	ImageNet+iNat'17	None	CE	All	All	32.4
Ours	ResNet-50	ImageNet+iNat'17	None	Focal	All	All	32.3
Vanilla Model	ResNet-101	ImageNet+iNat'17	None	CE	All	-	34.3
CB [25]	ResNet-101	ImageNet+iNat'17	None	CE	All	-	32.7
Ours	ResNet-101	ImageNet+iNat'17	None	CE	All	All	**31.5**

CHAPTER 5: LONG-HORIZON GRADIENT-BASED META-LEARNING

Humans can quickly learn the skills needed for new tasks by drawing from a fund of prior knowledge and experience. To grant machine learners this level of intelligence, meta-learning studies how to leverage past learning experiences to more efficiently learn for a new task [165]. A hallmark experiment design provides a meta-learner a variety of few-shot learning tasks (meta-training) and then desires it to solve previously unseen and yet related few-shot learning tasks (meta-test). This design enforces "learning to learn" because the few-shot training examples are insufficient for a learner to achieve high accuracy on any task in isolation.

Recent meta-learning methods focus on deep neural networks. Some learn recurrent neural networks as an update rule to a model [129, 4]. Some transfer attention schemes across tasks [117, 166]. Gradient-based meta-learning gains momenta recently following the seminal work [41]. It is model-agnostic meta-learning (MAML), learning a global model initialization from which a meta-learner can quickly derive task-specific models by using a few training examples.

In its core, MAML is a bilevel optimization problem [20]. The upper level searches for the best global initialization, and the lower level optimizes individual models, which all share the common initialization, for particular tasks sampled from a task distribution. This problem is hard to solve. [41] instead propose a "greedy" algorithm, which comprises two loops. The inner loop samples tasks and updates the task-specific models by k steps using the tasks' training examples. The k-step updates write a differentiable computation graph. The outer loop updates the common initialization by backpropagating meta-gradients through the computation graph. This method is "greedy" in that the number of inner steps is often small (e.g., $k = 1$). The outer loop takes actions before the inner loop sufficiently explores its search space.

This "greedy" algorithm is due to practical constraints that backpropagating meta-gradients through

the inner loop incurs high-order derivatives, big memory footprints, and the risk of vanishing or exploding gradients. For the same reason, some related work also turns to greedy strategies, such as meta-attack [34] and learning to reweigh examples [132].

To this end, two questions arise naturally. *Would a less greedy gradient-based meta-learner (say, k>10 inner steps) achieve better performance? How to make it less greedy?*

Some first-order algorithms [41, 122, 42] have provided an affirmative answer to the first question above. [127] proposed a less "greedy" MAML by regularizing the inner loop. However, they are highly tailored for that the meta-model and task models lie in the same space, preventing them from tackling other meta-learning problems, for example, the long-tailed classification described later.

To answer the questions for more general meta-learning scenarios, we provide some preliminary results by introducing a lookahead optimizer [187] into the inner loop. It can be viewed as a teacher-student scheme. We use a student neural network to explore the search space for a given task adequately (by a large number k of updates), and a teacher network then takes a "leap" toward the regions visited by the student. As a result, the teacher network not only arrives at a high-performing model but also defines a very lightweight computational graph for the outer loop. In contrast to the traditionally "greedy" meta-learning framework used in MAML [41], meta-attack [34], learning to reweigh examples [132], etc., the teacher is "lazy". It sends a student to optimize for a task up to many steps and moves only once after that.

Our approach improves the gradient-based meta-learning framework rather than a single algorithm. Hence, we evaluate it on different methods and tasks, including MAML and Reptile [122] for few-shot learning, a two-component weighting algorithm [73] for long-tailed classification, and meta-attack [34]. Extensive results provide an affirmative answer to the first question above: long-horizon exploration in the inner loop improves a meta-learner's performance. We expect our

approach, along with the compelling experimental results, can facilitate future work to address the second question above.

5.1 "Greedy" Gradient-Based Meta-Learning

We first review gradient-based meta-learning from the perspective of "search space carving".

Notations. Let $P_{\mathcal{T}}$ denote a task distribution. For each task drawn from the distribution $\mathcal{T} \sim P_{\mathcal{T}}$, we have a training set $\mathcal{D}_{tr}$ and a validation set $\mathcal{D}_{val}$, both in the form of $\{(x_1, y_1), (x_2, y_2), \cdots\}$ where x_m and y_m are respectively an input and a label. We learn a predictive model for a task by minimizing the empirical loss $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi)$ over the training set while using the validation set to choose hyper-parameters (e.g., early stopping), where ϕ collects all trainable parameters of the model. Similarly, we denote by $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi)$ the loss calculated over the validation set.

Meta-learning as "space carving". Instead of focusing on an isolated task, meta-learning takes a global view and introduces a meta-model, parameterized by θ, that can improve the learning efficiency for all individual tasks drawn from the task distribution $P_{\mathcal{T}}$. The underlying idea is to derive a task-specific model ϕ from not only the training set $\mathcal{D}_{tr}$ but also the meta-model θ, i.e., $\phi \in \mathcal{M}(\theta, \mathcal{D}_{tr})$. We refer to $\mathcal{M}(\theta, \mathcal{D}_{tr})$ the "carved" search space for the task-specific model ϕ, where the "carving" function is realized as an attention module in [166, 117], as a conditional neural process in [49, 57], as a gradient-based update rule in [41, 126, 99, 122], and as a regularized optimization problem in [127, 193].

An optimal meta-model θ^* is supposed to yield the best task-specific models in expectation,

$$\begin{aligned} &\theta^* \leftarrow \arg\min_{\theta} \ \mathbb{E}_{\mathcal{T}\sim P_{\mathcal{T}}, \mathcal{D}_{val}\sim\mathcal{T}} \, \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi^*(\theta)) \\ &\text{subject to } \phi^*(\theta) \leftarrow \arg \min_{\phi\in\mathcal{M}(\theta,\mathcal{D}_{tr})} \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi). \end{aligned} \tag{5.1}$$

One can estimate the optimal meta-model θ^* from some tasks and then use it to "carve" the search space, $\mathcal{M}(\theta^*, \mathcal{D}_{tr})$, for novel tasks' models.

Gradient-based meta-learning. One of the notable meta-learning methods is MAML [41], which uses a gradient-based update rule to "carve" the search space for a task-specific model,

$$\begin{aligned}\mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr}) := \{\phi_0 \leftarrow \theta\} \cup \{\phi_j \,|\, \phi_j \leftarrow \phi_{j-1} \\ -\alpha \nabla_\phi \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi_{j-1}),\ j = 1, 2, \cdots, k\}\end{aligned} \tag{5.2}$$

where the meta-model θ becomes an initialization to the task-specific model ϕ_0, the other candidate models $\phi_1, \cdots, \phi_k$ are obtained by gradient descent, and $\alpha > 0$ is a learning rate. Substituting it into equation (5.1), $\phi_k \in \mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr})$ is naturally a solution to the lower-level optimization problem, and MAML solves the upper-level optimization problem by meta-gradient descent,

$$\theta \leftarrow \theta - \beta \mathbb{E}_{\mathcal{T} \sim P_{\mathcal{T}}, \mathcal{D}_{val} \sim \mathcal{T}} \nabla_\theta \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi_k(\theta)), \tag{5.3}$$

where β is a learning rate, and $\phi_k(\theta)$ indicates the dependency on the meta-model θ. The meta-gradient must backpropagate through the chain of updates in eq. (5.2), which has to be short (e.g., $k = 1$) to avoid big memory footprints, high-order derivatives, and the risk of vanishing or exploding gradients.

We say MAML is "greedy" in that it descends meta-gradients for the meta-model θ before it runs adequate updates to the task-specific model ϕ. As an increasing number of works adopt the gradient-based "search space carving" for task-specific models [99, 127, 126, 43, 180], they also bear greedy algorithms. Relaxing the greedy strategy may benefit not one, but a variety of, high-order meta-learning methods.

5.2 A "Lazy" Approach to Gradient-Based Meta-Learning

In this section, we describe a "lazy" meta-learning approach, which is readily applicable to different gradient-based meta-learning algorithms. We first describe the general approach as an improvement to MAML and then customize it for few-shot learning, long-tailed classification, and meta-attack.

5.2.1 General Approach

Given a meta-model θ, we "carve" the search space for task-specific models $\phi \in \mathcal{M}(\theta, \mathcal{D}_{tr})$ by a teacher-student scheme. The key idea is to let a student explore the search space adequately using the training set of a task-specific model without worrying the length of the update chain because a teacher will examine the explored regions by the student, followed by a one-step "leap". Hence, one can update the meta-model by backpropagating meta-gradients through the teacher's "leap", not the student's update chain (ignoring that the chain starts from the meta-model). Figure 5.1 illustrates the main idea.

An exploratory student acts exactly the same as the gradient-based updates in MAML except that it explores the feasible space by a large number of steps ($k > 10$), resulting in $k+1$ checkpoints of a task-specific model $\phi \in \mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr}) = \{\phi_j, j = 0, \cdots, k\}$. It is clear from Section 5.1 that we cannot backpropagate the meta-gradients through the long chain of checkpoints, $\phi_0, \cdots, \phi_k$, made by the exploratory student.

A lazy teacher sits at the initialization $\phi_0 = \theta$ till the student stops. It then takes a "leap" towards the region explored by the student. The teacher essentially defines another "carved search space"

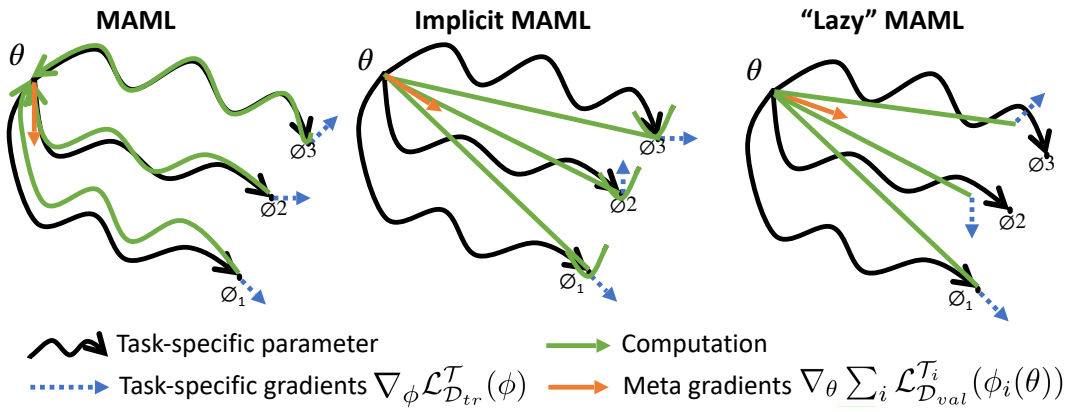


Figure 5.1: To compute the meta-gradients $\nabla_\theta \sum_i \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}(\phi_i(\theta))$, MAML [41] differentiates through the inner updates, the implicit MAML [127] approximates local curvatures, while we differentiate through the "lazy" teacher's one-step "leap". The exploratory student may make many steps of inner updates before the teacher's "leap".

for the task-specific model ϕ,

$$\mathcal{M}_{\text{LAZY}}(\theta, \mathcal{D}_{tr}) := \gamma\theta + (1-\gamma)\mathcal{R}_{k-b+1,\cdots,k} \tag{5.4}$$

where $\gamma \in [0, 1]$. The region $\mathcal{R}_{k-b+1,\cdots,k}$ is a convex hull of the last b checkpoints the student visited:

$$\begin{aligned}\mathcal{R}_{k-b+1,\cdots,k} := \alpha_{k-b+1}\phi_{k-b+1} + \alpha_{k-b+2}\phi_{k-b+2} + \\ \cdots + \alpha_k\phi_k,\end{aligned} \tag{5.5}$$

where the coefficients $\{\alpha\}$ are non-negative and their sum equals 1, i.e., $\alpha_{k-b+1} + \cdots + \alpha_k = 1$. The last b checkpoints presumably cover a high-quality task-specific model ϕ by a better chance than the first few checkpoints. We shall experiment with $b = 3$ and $b = 1$.

Any task-specific model ϕ in this "lazy" space $\mathcal{M}_{\text{LAZY}}(\theta, \mathcal{D}_{tr})$ is determined by the hyper-parameters γ and $\alpha_{k-b+1}, \cdots, \alpha_k$, over which we conduct a grid search to minimize the validation loss $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi)$.

This is similar in spirit to meta-SGD [99], which uses the validation data to search for the learning rates.

Denote by $\hat{\gamma}\theta + (1-\hat{\gamma})\hat{\phi}$ the task-specific model as a result of the grid search. Notably, it is only one hop away from the meta-model θ, making it easy to compute meta-gradients.

Concretely, the meta-gradient descent for the meta model θ becomes $\theta \leftarrow \theta - \beta \mathbb{E}_{\mathcal{T}\sim P_{\mathcal{T}}, \mathcal{D}_{val}\sim\mathcal{T}} \nabla_\theta \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\hat{\gamma}\theta + (1-\hat{\gamma})\hat{\phi})$, which is apparently more manageable than the gradients in eq. (5.3) when $k > 1$.

Algorithm 4 "Lazy" Meta-Learning

Require: A distribution over tasks $P_{\mathcal{T}}$

Require: Learning rates η, β

Ensure: The meta model θ

```
Randomly initialize the meta-model θ
while not done do
    Sample a batch of tasks {T_i ~ P_T}
    for all {T_i} do
        Sample data D_tr and D_val for T_i
        φ_{i,0} ← θ
        for j = 1, 2, ···, k do                                   //student
            φ_{i,j} ← φ_{i,j-1} − η∇_φ L^{T_i}_{D_tr}(φ_{i,j-1})
        end for
        Grid-search M_LAZY(θ, D_tr) such that L^{T_i}_{D_val} is minimized at γ̂_i θ + (1 − γ̂_i)φ̂_i   //teacher
        φ_i(θ) ← γ̂_i θ + (1 − γ̂_i)φ̂_i                              //teacher
    end for
    θ ← θ − β∇_θ Σ_i L^{T_i}_{D_val}(φ_i(θ))
end while
```

Algorithm 4 presents our "lazy" approach in detail. In the outer while-loop, we sample a batch of tasks $\{\mathcal{T}_i\}$ (Line 3, or L3) and use them to make a gradient update to the meta-model θ (L13). All task-specific models $\{\phi_{i,0}\}$ are initialized to the current meta-model θ (L6). For each task $\mathcal{T}_i$, the student first runs gradient descent with respect to the task-specific model ϕ_i up to k steps (L8), and the teacher then takes a "leap" from the initial meta-model θ according to the checkpoints visited by the student (L10–11)

Remarks. Our "lazy" teacher is motivated by the lookahead optimizer [187]. They have some key differences as follows due to the meta-learning setup. We initialize multiple task-specific models by the meta-model. Moreover, we dynamically choose the "leap" rate γ by a validation set. Finally, the validation data allows us to take advantage of not one checkpoint, but a region around the checkpoints visited by the student.

Like Reptile, our approach allows the inner loop to make many steps of updates to task-specific models. Moreover, we share the same update rule as Reptile by the end of the many-step exploration. However, we apply that rule to the task-specific models, while Reptile essentially uses it to update the meta-model. Unlike Reptile, we use meta-gradients to update the meta-model. This difference is subtle and vital, making it straightforward to apply our approach to the two-component weighting algorithm (**Algorithm 5**) for long-tailed classification (and other meta-like algorithms) but unclear how to do it for Reptile.

We share the same goal, to make MAML less "greedy", as the recently proposed implicit gradients (iMAML) [127]. iMAML changes the lower-level problem in eq. (5.1) to an ℓ_2-regularized problem, which lends an analytical expression for the meta-gradient. But it is expensive to compute and has to be approximated by a conjugate gradient algorithm. The ℓ_2 regularization also falls short in capturing structural relations between a meta-model and task-specific models.

5.2.2 *Few-Shot Learning, Long-Tailed Classification, and Meta-Attack*

Since the "lazy" teacher does not change the innermost loop of gradient-based meta-learning — it instead "leaps" over the chain of updates to the task-specific model ϕ, we can apply it to different algorithms. We evaluate it on few-shot learning, long-tailed classification, and meta-attack, in which meta-learning based methods have led to state-of-the-art results.

Few-shot learning in this paper concerns an N-way-K-shot classification problem. To customize Algorithm 4 for this problem, we randomly select N classes for each task $\mathcal{T}_i$ and then draw from each class $K+1$ examples with labels, K of which are assigned to the training set $\mathcal{D}_{tr}$ and one is to the validation set $\mathcal{D}_{val}$. Besides, we choose the hyper-parameter γ_i by using the task-specific model's classification accuracy on the validation set, instead of the loss in L10, Algorithm 4.

There is an interesting "trap" in few-shot learning, identified as over-fitting by memorization [180]. The tasks $\{\mathcal{T}_i\}$ drawn from a distribution $P_{\mathcal{T}}$ are supposed to be i.i.d., but they could be correlated in the following scenario. Suppose there exists a global order of all classes. If we maintain this order among the N classes in each task, the meta-model could over-fit the tasks seen during meta-training by memorizing the functions that solve these tasks, and it would fail to generalize to new tasks. Hence, it is important to randomly shuffle the N classes every time we sample them for a task (e.g., "dogs" and "cats" are respectively labeled as 0 and 1 in a two-way classification task, and yet they are shuffled to 1 and 0 in another two-way task).

We will empirically show that our approach is less prone to over-fitting than MAML even without class shuffling. A possible reason is that we use longer chains of updates ($\phi_0, \cdots, \phi_k, k > 10$) to learn the functions that solve the individual tasks, making them harder to memorize.

Long-tailed classification emerges as an inevitable challenge as object recognition makes progress toward large-scale, fine-grained classes [1, 174], which often exhibit a long-tailed distribution. To

uplift infrequent classes, [73] propose to weigh each training example by two components, a fixed component w_y to balance different classes [25] and a trainable component ϵ_i. We improve their learning method by a "lazy" teacher, as described in Algorithm 5. It alternatively optimizes the per-example weight ϵ_i (using a balanced validation set) and a recognition network θ (using the long-tailed training set), in the same spirit as meta learning (cf. Algorithm 4 vs. L5-12 in Algorithm 5). We insert a "lazy" teacher model to L6, let it take a "leap" in L12, and then backpropagate the gradient with respect to the per-example weight ϵ_i through the "leap".

Algorithm 5 "Lazy" Two-Component Weighting for Long-Tailed Recognition

Require: A training set $\mathcal{D}_{tr}$ whose class frequency is long-tailed, a balanced validation set $\mathcal{D}_{val}$

Require: Class-wise weights $\{w_y\}$ estimated by using [**?**]

Require: Learning rates η, τ, pre-training steps t_1, fine-tuning steps t_2

1: Train a recognition network, parameterized by θ, for t_1 steps by a standard cross-entropy loss

2: **for** $t = t_1 + 1, \cdots, t_1 + t_2$ **do**

3: Sample a mini-batch B from the training set $\mathcal{D}_{tr}$

4: Set $\epsilon_i \leftarrow 0, \forall i \in B$, and denote by $\epsilon := \{\epsilon_i, i \in B\}$

5: Compute $\mathcal{L}_B(\theta, \epsilon) := \frac{1}{|B|} \sum_{i \in B} (w_{y_i} + \epsilon_i) \mathcal{L}_i(\theta)$ //$\mathcal{L}_i$ is a cross-entropy over the i-th input

6: Update $\tilde{\theta}(\epsilon) \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_B(\theta, \epsilon)$ // The "lazy" teacher, which depends on ϵ

7: Initialize a student model by setting $\phi_0 \leftarrow \tilde{\theta}(\epsilon)$

8: **for** $j = 1, 2, ..., k$ **do**

9: Update the student model by gradient descent $\phi_j \leftarrow \phi_{j-1} - \eta \nabla_\phi \mathcal{L}_B(\phi_{j-1}, \epsilon)$

10: **end for**

11: Grid search for γ s.t. the teacher's "leap", $\gamma \tilde{\theta}(\epsilon) + (1 - \gamma) \phi_k$, yields high accuracy on $\mathcal{D}_{val}$

12: Update $\epsilon \leftarrow \epsilon - \tau \nabla_\epsilon \mathcal{L}_{\mathcal{D}_{val}}(\gamma \tilde{\theta}(\epsilon) + (1 - \gamma) \phi_k)$

13: Compute $\mathcal{L}_B(\theta, \epsilon)$ (cf. Line 5) and update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_B(\theta, \epsilon)$

14: **end for**

Meta-attack [34] is a query-efficient blackbox attack algorithm on deep neural networks. Recent work [80, 29, 91] has shown that one can manipulate an image recognition network's predictions by adding very small perturbations to benign inputs. However, if the network's architecture and weights are unknown (blackbox), it takes a large number of queries into the network to find a valid adversarial example. To improve the query efficiency, [34] propose to learn a meta-model from many whitebox neural networks and then generalize it to blackbox attacks. They train this meta-model by using the same meta-learning framework as Algorithm 4. Therefore, it is straightforward to improve their inner loop by our "lazy" teacher. The inputs to the meta-attacker are images, and the desired outputs are their gradients — during meta-training, the gradients are generated from different classification models. Instead of the cross-entropy loss, meta-attack adopts a mean-squared error (MSE) loss in the inner loop, i.e.,

$$\mathcal{L}_{\mathcal{D}_{tr}}^{\mathcal{T}_i} = ||\phi(\mathcal{X}_{ij}) - \mathcal{G}_{ij}||_2^2 \tag{5.6}$$

where the task $\mathcal{T}_i$ is to find adversarial examples for the inputs to the i-th pre-trained classification network, $\mathcal{X}_{ij}$ is an image sampled for the task, $\mathcal{G}_{ij}$ are the gradients of the classification network with respect to (w.r.t.) the image, and $\phi(\cdot)$ is a task-specific model whose output is to approximate the gradients $\mathcal{G}_{ij}$. This model is useful because, given a blackbox classification network, we can use the task-specific model to predict the gradients of this network w.r.t. an image, followed by gradient ascent towards an adversarial example (cf. Algorithm 7).

Algorithm 6 presents how to train this meta-attacker by applying our "lazy" teacher to Reptile, and we then follow Algorithm 7 for attacking blackbox networks [34].

Algorithm 6 Training algorithm of meta-attack using "lazy" Reptile

Require: A distribution over tasks $P_{\mathcal{T}}$

Require: Input Images $\mathcal{X}$, gradients $\mathcal{G}_i$ generated from a classification network serving task $\mathcal{T}_i$

Require: Learning rates α, β

Ensure: The meta attacker θ

Randomly initialize the meta-attacker θ
while not done **do**
    Sample a batch of tasks $\{\mathcal{T}_i \sim P_{\mathcal{T}}\}$
    for all $\{\mathcal{T}_i\}$ **do**
        Sample data $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ for $\mathcal{T}_i$ // in the form of $\{\mathcal{X}_{ij}, \mathcal{G}_{ij}\}$
        $\phi_{i,0} \leftarrow \theta$
        for $j = 1, 2, \cdots, k$ **do**
            $\phi_{i,j} \leftarrow \phi_{i,j-1} - \alpha \nabla_{\phi} \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{tr}}(\phi_{i,j-1})$
        end for
        $\gamma_i \leftarrow \arg\min_{\gamma} \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}(\gamma\theta + (1-\gamma)\phi_{i,k})$
        $\phi_i(\theta) \leftarrow \gamma_i \theta + (1-\gamma_i)\phi_{i,k}$
    end for
    $\theta \leftarrow \theta - \beta \sum_i (\theta - \phi_i(\theta))$
end while

Algorithm 7 Adversarial Meta-Attack

Require: Test image x_o with label t, meta-attacker f_θ, target model $\mathcal{M}_{tar}$, iteration interval j, selected top-q coordinates

for $t = 0, 1, 2, \cdots$ **do**
  if $(t+1) \mod j = 0$ **then**
    Perform zeroth-order gradient estimation [79] on top-q coordinates, denoted as I_t and
    obtain g_t.
    Fine-tune meta-attacker f_θ with (x_t,g_t) on I_t by $\mathcal{L} = |[f_\theta(x_t]_{I_t} - [g_t]_{I_t}|_2^2$.
  else
    Generate the gradient map g_t directly from meta-attacker f_θ with x_t,
    select coordinates I_t.
  end if
  Update $[x']_{I_t} = [x_t]_{I_t} + \lambda[g_t]_{I_t}$.
  if $\mathcal{M}_{tar}(x') \neq t$ **then**
    $x_{adv} = x'$
    break
  else
    $x_{t+1} = x'$
  end if
end for

Ensure: adversarial example x_{adv}.

5.3 Experiments

We evaluate the "lazy", long-horizon meta-learning approach by plugging it into different algorithms with applications to few-shot learning, long-tailed recognition, and meta-attack.

5.3.1 Few-Shot Learning

We experiment with four datasets for few-shot learning: Omniglot [85], MiniImageNet [166], TieredImageNet [131], and CIFAR-FS [10]. The experiment protocols and implementation details largely follow MAML [41] and Reptile [122]. The Omniglot dataset consists of handwritten characters from 50 different alphabets and 1623 characters. There are 20 handwritten examples of each character. MiniImageNet contains 100 classes form ImageNet [27], which are split to 64, 16, and 20 classes for meta-training, meta-validation, and meta-test, respectively. TieredImagNet has 608 classes from ImageNet, which are grouped into 34 higher-level categories following the ImageNet taxonomy. They are split into 20 meta-training categories, 6 meta-validation categories, and 8 meta-test categories. Due to this partition scheme, the meta-test classes are less similar to the meta-training classes in TieredImageNet than in other datasets. CIFAR-FS re-purposes CIFAR-100 [82], splitting its 100 classes into 64, 16, and 20 classes for meta-training, meta-validation, and meta-test, respectively.

5.3.1.1 Experiment protocols and hyper-parameters

Our experiment protocols and implementation details largely follow MAML [41] and Reptile [122]. In particular, we use a convolutional neural network that comprises four modules in all the experiments. Each module has 3x3 convolutions, a batch-normalization layer, 2x2 max-pooling, and the ReLU activation, and every convolutional layer contains 64 filters for the experiments on Om-

niglot and 32 filters for other datasets. For fair comparison, we also re-implement some of the existing methods using this network architecture. We report more details for the "Lazy" Reptile in Table 5.1. For "Lazy" MAML, we set k = 10 for 1-shot and 5-shot tasks, respectively, on both MiniImageNet and TieredImageNet.

Table 5.1: Hyper-parameter details for few-shot learning in ours (Reptile). The "Eval inner batch" row shows the numbers for both 1-shot and 5-shot settings.

Hyper-parameter	Omniglot	CIFAR-FS	Mini-ImageNet	TieredImageNet
Inner learning rate (η)	0.001	0.001	0.001	0.001
Inner iterations (k)	5	8	8	8
Inner batch size	10	10	10	10
Training shots	10	15	15	15
Outer step-size (β)	1.0	1.0	1.0	1.0
Total outer-iterations	100k	120k	120k	130k
Meta batch size	20	20	20	20
Eval. inner iterations	50	50	50	50
Eval. inner batch	5/15	5/15	5/15	5/15

Table 5.2: Our approach applied to MAML and Reptile for five-way few-shot classification on MiniImageNet (Accuracy $\pm$ 95% confidence interval over 2000 runs)

Method	MiniImageNet	
	1-shot	5-shot
MAML [41]	48.70 ± 1.84	63.11 ± 0.92
"Lazy" MAML ($b = 1$)	48.26 ± 1.78	64.13 ± 1.90
"Lazy" MAML ($b = 3$)	48.17 ± 1.84	63.73 ± 1.10
Reptile [122]	49.97 ± 0.32	65.99 ± 0.58
"Lazy" Reptile ($b = 1$)	51.50 ± 1.00	67.22 ± 0.97
"Lazy" Reptile ($b = 3$)	52.67 ± 1.01	68.77 ± 0.98

Our approach permits long-horizon inner updates and involves a convex hull of the last few checkpoints. In Table 5.2, we first experiment with the last b=3 and b=1 checkpoints. We test them

with two representative meta-learning algorithms: MAML (cf. Algorithm 4) and Reptile (replacing Line 13 (L13) in Algorithm 4 with $\theta \leftarrow \theta - \beta \sum_i (\theta - \phi_i(\theta))$). The intervals are 0.05 in the grid search (L10), and the search range for the learning rate γ is between 0.75 and 0.95.

Table 5.2 shows that there is no significant difference between $b = 3$ and $b = 1$, so we shall employ $b = 1$ for the remaining experiments. Moreover, the "lazy" variation improves the vanilla Reptile, but not MAML, probably because the five-way one/five-shot learning is too simple for MAML to take advantage of the long-horizon inner updates. We next study many-way few-shot learning tasks, which are arguably more complex.

5.3.1.2 MAML vs. "Lazy" MAML for many-way few-shot learning

We switch to the TieredImageNet dataset since there are only 20 classes in MiniImageNet's meta-test set. The left panel of Figure 5.2 shows the results of MAML, FOMAML and "Lazy" MAML for N-way-five-shot learning, where N varies in $\{5, 20, 30, 50\}$, and the student runs for $k = 10, 15, 20, 20$ inner steps, respectively. We set the inner learning rate (η) to 0.005 and the outer learning rate (β) to 0.001 for all the settings. The "lazy" variation is on par with MAML for the five-way classification, and it outperforms MAML, and FOMAML for 20-way, 30-way, and 50-way five-shot classifications. This trend indicates that the many-way few-shot learning problems desire more inner updates to the task-specific models, amplifying the benefit of the "lazy" teacher.

5.3.1.3 Reptile vs. "Lazy" Reptile for many-way few-shot learning

The left panel of Figure 5.3 compares the results of Reptile and "Lazy" Reptile for N-way-five-shot learning on TieredImageNet where N varies in $\{5, 20, 30\}$. Our approach outperforms Reptile. We emphasize that not all meta-learning algorithms can be approximated by a first-order version; for

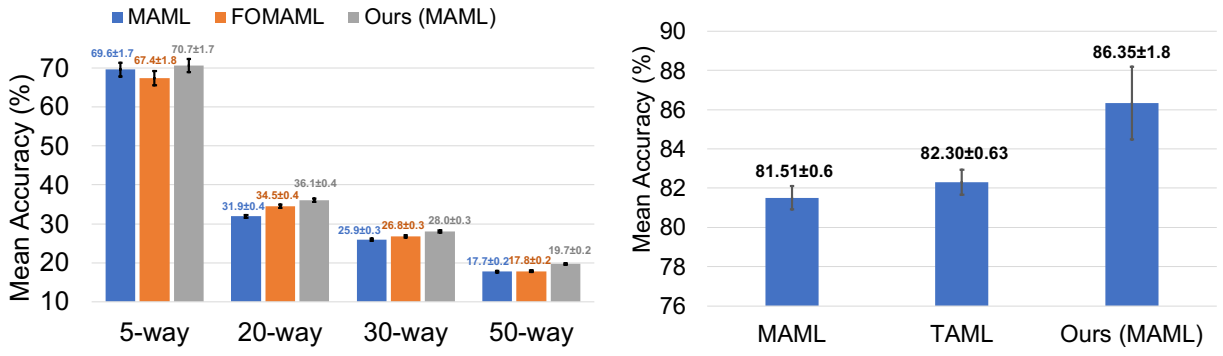


Figure 5.2: Left: Mean Accuracy (%) for N-way-five-shot classification on TieredImageNet. Right: Mean Accuracy (%) for 20-way-one-shot non-i.i.d. [180] classification tasks on Omniglot.

example, it is not immediately clear how to do it for Algorithm 5, the two-component weighting method for long-tailed classification. The right panel of Figure 5.3 shows some 20-way-5-shot results on MiniImageNet. We can see that our lazy strategy boosts both MAML and Reptile by a significant margin. It again indicates that more training data needs more steps of exploration for a task-specific model and hence magnifies the benefit of our teacher-student scheme introduced to both MAML and Reptile.

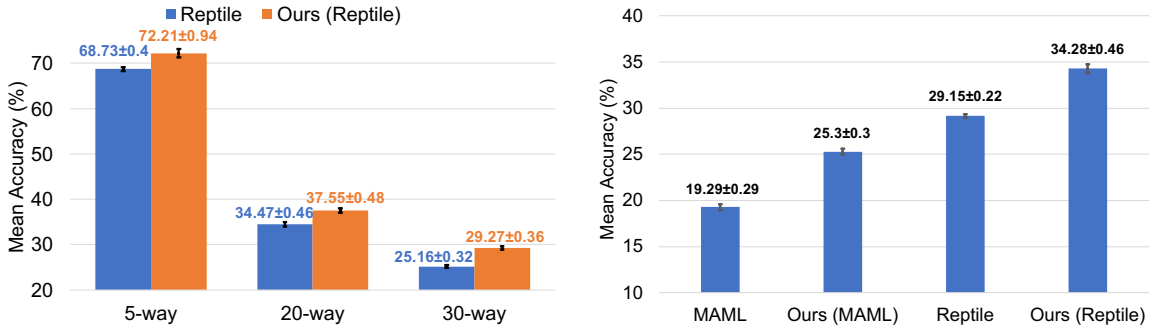


Figure 5.3: Left: Mean Accuracy (%) for N-way-five-shot classification on TieredImageNet. b). Mean Accuracy (%) for 20-way-5-shot classification on MiniImageNet.

5.3.1.4 *Many-shot Classification*

The Figure 5.4 shows the results of MAML and "Lazy" MAML for five-way-K-shot learning on MiniImageNet. We vary K in $\{1, 5, 20, 50\}$ and let the student run for $k = 10, 15, 15, 20$ steps, respectively. Under the 1-shot and 5-shot settings, our approach is comparable to MAML, but it significantly outperforms MAML for 20-shot and 50-shot classifications. This trend indicates that more training data desires more steps of exploration for a task-specific model and hence magnifies the benefit of our teacher-student scheme introduced to MAML.

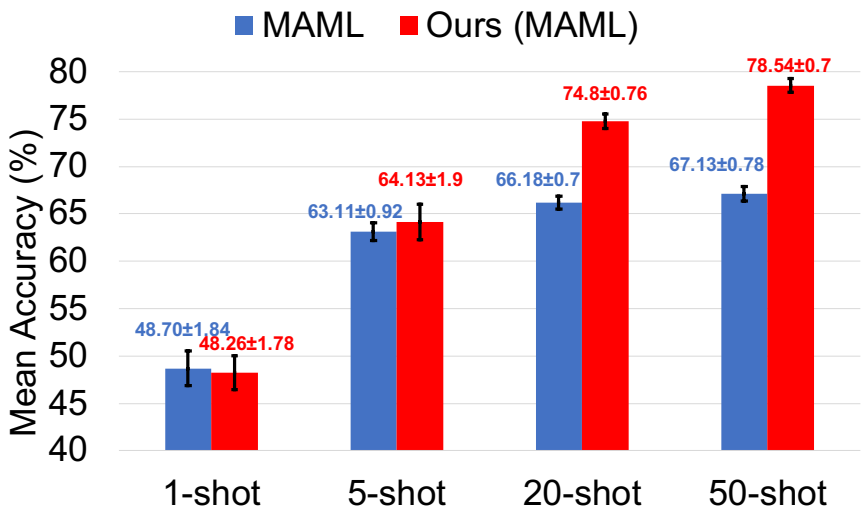


Figure 5.4: Mean Accuracy (%) for five-way K-shot classification on MiniImageNet.

5.3.1.5 *"Lazy" MAML is less prone to over-fitting by memorization than MAML*

The right panel of Figure 5.2 shows some 20-way-one-shot classification results on Omniglot when we learn from non-i.id. tasks, i.e., by maintaining a global order of all training classes. This global order creates a shortcut for meta-learning methods; they may memorize the order from the meta-training tasks and fail to generalize to meta-test tasks [180]. We can see that the "lazy" teacher boosts MAML by a large margin and outperforms TAML [71], and FOMAML indicating that it is

less prone to over-fitting by memorization. A plausible reason is that the $k = 15$ steps taken by the exploratory student make it harder to memorize than the one-step update in MAML or TAML.

5.3.1.6 Five-way-few-shot learning

We compare our approach with state-of-the-art meta-learning methods for five-way few-shot learning problems on four datasets. The results are shown in Tables 5.3 for MiniImageNet and TieredImageNet . For our own approach, we study both the MAML-style update to the meta-model (ours (MAML), L13 in Algorithm 4) and the Reptile-style [122] update (ours (Reptile), L14 in Algorithm 4) for MiniImageNet and TieredImageNet. Batch normalization with test data yields about 2% improvement over the normalization with the training data only, and we report the results of both scenarios.

Table 5.3: Five-way few-shot classification accuracies (%) on MiniImageNet and TieredImageNet. The $\pm$ shows 95% confidence intervals computed over 2000 tasks.

Method	BN w/	Mini-ImageNet		TieredImageNet	
	Test	1-shot	5-shot	1-shot	5-shot
MAML [41]	✗	46.21 ± 1.76	61.12 ± 1.01	49.60 ± 1.83	66.58 ± 1.78
MAML [41]	✓	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	69.60 ± 1.73
Meta-Curvature [126]	✓	48.83 ± 1.80	62.63 ± 0.93	50.30 ± 1.99	66.14 ± 0.95
iMAML [127]	✓	49.30 ± 1.88	-	-	-
Ours (MAML)	✓	48.26 ± 1.78	64.13 ± 1.90	51.03 ± 1.70	70.67 ± 1.72
FOMAML [41]	✗	45.53 ± 1.58	61.02 ± 1.12	48.01 ± 1.74	64.07 ± 1.72
Reptile [122]	✗	47.07 ± 0.26	62.74 ± 0.37	49.12 ± 0.43	65.99 ± 0.42
Meta-MinibatchProx [193]	✗	47.81 ± 1.00	63.18 ± 1.00	49.97 ± 0.93	66.60 ± 0.91
Ours (Reptile)	✗	48.14 ± 0.94	64.64 ± 0.92	51.15 ± 0.95	68.84 ± 0.90
FOMAML [41]	✓	48.07 ± 1.75	63.15 ± 0.91	50.12 ± 1.82	67.43 ± 1.80
Reptile [122]	✓	49.97 ± 0.32	65.99 ± 0.58	51.34 ± 0.4	68.73 ± 0.40
Meta-MinibatchProx [193]	✓	50.08 ± 1.00	66.28 ± 0.98	53.71 ± 1.04	69.78 ± 0.95
Ours (Reptile)	✓	$\mathbf{51.50 \pm 1.00}$	$\mathbf{67.22 \pm 0.97}$	$\mathbf{54.41 \pm 1.00}$	$\mathbf{72.21 \pm 0.94}$

Table 5.4: Five-way few-shot classification accuracies (%) on Omniglot and CIFAR-FS. The $\pm$ shows 95% confidence intervals computed over 1000 tasks.

Method	BN w/ Test	Omniglot		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot
MAML [41]	✓	98.70 ± 0.40	$\mathbf{99.90 \pm 0.10}$	56.50 ± 1.90	70.50 ± 0.90
iMAML [127]	✓	$\mathbf{99.16 \pm 0.35}$	99.67 ± 0.12	-	-
Reptile [122]	✗	95.39 ± 0.09	98.90 ± 0.10	53.12 ± 1.34	69.40 ± 1.30
Ours (Reptile)	✗	95.44 ± 0.57	98.92 ± 0.29	54.64 ± 1.30	70.56 ± 1.20
FOMAML [41]	✓	98.30 ± 0.50	99.20 ± 0.20	55.6 ± 1.88	69.52 ± 0.91
Reptile [122]	✓	97.68 ± 0.04	99.48 ± 0.06	57.50 ± 0.45	71.88 ± 0.42
Ours (Reptile)	✓	98.20 ± 0.38	99.70 ± 0.16	$\mathbf{59.36 \pm 1.44}$	$\mathbf{74.90 \pm 1.28}$

It can be seen that our results are better than or comparable with those of the competing methods. In general, the improvements by our teacher-student scheme are more significant on 5-shot settings than on 1-shot settings, verifying the trend in Section 5.3.1.2 that more training data can better leverage the exploratory student in our method. Besides, ours (Reptile) outperforms ours (MAML) probably for two reasons. One is that ours (Reptile) uses more than k shots of training examples per class for a k-shot learning problem during meta-training, following the experiment setup of Reptile [122]. The other is that the second-order gradients in ours (MAML) make the training procedure less stable than Reptile. We hypothesize that a many-shot setting would be less sensitive to both factors. Indeed, we verified this hypothesis by another five-way-50-shot learning experiment with ours (Reptile), which yields $76.17 \pm 0.32\%$ on MiniImageNet and is lower than 78.54 ± 0.70 by ours (MAML).

The results on Omniglot and CIFAR-FS are reported in Table 5.4. We only report ours (Reptile) due to its low computation cost. It can be seen that our results are better than or comparable with those of the competing methods.

5.3.1.7 *Computational Analysis*

In our evaluation, we also want to answer the following question empirically. How does the memory requirements of "Lazy" MAML compare with MAML?. Figure 5.5 shows the memory trade-off for "Lazy" MAML and MAML on 5-shot 20-way MiniImageNet. "Lazy" MAML decouples the dependency of inner and outer loop by teacher-student scheme which allows it to define a very lightweight computation graph. The figure shows that the memory of the "Lazy" MAML doesn't exceed beyond 5 GB for many inner gradient steps while on the other hand, MAML reaches the capacity of 12 GB after 5 inner steps.

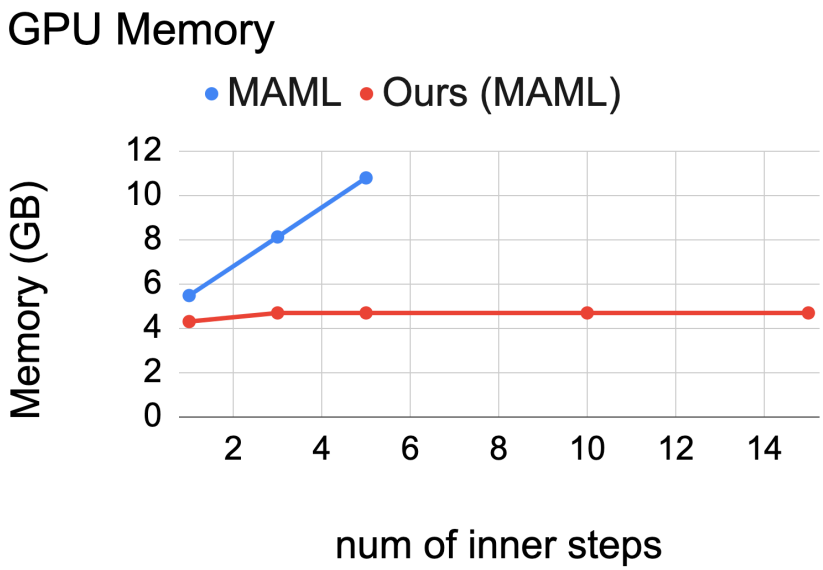


Figure 5.5: Memory trade-offs with 4 layer CNN on 20-way-5-shot MiniImageNet task.

5.3.2 *Long-Tailed Classification*

Following the experiment setup in [25] and [73], we use the CIFAR-LT-100 dataset [25] to compare our Algorithm 5 with several long-tailed recognition methods. [25] created multiple long-tailed

datasets by removing training examples from CIFAR-100 [82] according to different power law distributions. In each version, we compute an imbalance factor as the ratio between the sizes of the head class and the tail class. We run $k = 5$ steps in the innermost loop of Algorithm 5.

Table 5.5: Test top-1 errors (%) of ResNet-32 on CIFAR-LT-100 under different imbalance settings.

Method ↓ Imbalance factor →	200	100	50	20
Standard cross-entropy training	65.16	61.68	56.15	48.86
Class-balanced cross-entropy training [25]	64.30	61.44	55.45	48.47
Class-balanced fine-tuning [24]	61.78	58.17	53.60	47.89
Learning to reweight [132]	67.00	61.10	56.83	49.25
Meta-weight [153]	63.38	58.39	54.34	46.96
Two-component weighting [73]	60.69	56.65	51.47	44.38
Lazy two-component weighting (ours)	**58.67**	**53.46**	**48.24**	**43.68**

Table 5.5 shows the test errors (%) under different imbalance factors. We can see that our teacher-student scheme boosts the original two-component weighting approach [73] under all the imbalance factors. The results are especially interesting in that Algorithm 5 is not exactly a meta-learning method, though it shares the same framework as the gradient-based meta-learning due to the two nested optimization loops. Besides, compared with the other competing methods, our results establish a new state of the arts for the long-tailed object recognition.

5.3.3 Meta-Attack

We evaluate the "lazy" meta-attack on MNIST [86] and CIFAR-10 [82]. We follow [34] for the experiment setup and all training details, including the network architectures used to generate gradients for input images, the attack models, meta-attack models, and evaluation metrics for both the datasets, to name a few. The learning rates in the inner and outer loops are both 0.01. We let the

student run $k = 8$ and $k = 10$ steps in the innermost loop for MNIST and CIFAR-10, respectively.

Table 5.6: Untargeted adversarial attack results on MNIST and CIFAR10. We achieve comparable success rates and average ℓ_2 distortions with other methods by using a smaller number of queries.

Dataset / Target model	Method	Success Rate	Avg. ℓ_2	Avg. Queries
MNIST / Net4	Zoo [16]	1.00	1.61	21,760
	Decision boundary [2]	1.00	1.85	13,630
	Opt-attack [18]	1.00	1.85	12,925
	AutoZoom [163],	1.00	1.86	2,412
	Bandits [68]	0.73	1.99	3,771
	Meta-attack [34]	1.00	1.77	749
	Lazy meta-attack (ours)	1.00	1.65	**566**
CIFAR10 / Resnet18	Zoo [16]	1.00	0.30	8,192
	Decision boundary [2]	1.00	0.30	17,010
	Opt-attack [18]	1.00	0.33	20,407
	AutoZoom [163]	1.00	0.28	3,112
	Bandits [68]	0.91	0.33	4,491
	FW-black [15]	1.00	0.43	5,021
	Meta-attack [34]	0.94	0.34	1,583
	Lazy meta-attack (ours)	0.98	0.45	**1,061**

Table 5.6 shows the results of untargeted attack, namely, the attack is considered successful once it alters the recognition network's prediction to any incorrect class. In addition to the original meta-attack [34], Table 5.6 also presents several existing blackbox attack methods for comparison. We can see that meta-attack and our "lazy" meta-attack yield about the same success rates as the other blackbox attacks. The second-to-the-right column is about the average ℓ_2 distortion an attacker makes to an input, the lower the better. The rightmost column is about the number of queries an attacker makes into the recognition network, the lower the better. The "lazy" meta-attack is able to achieve comparable success rates and ℓ_2 distortion rates with the other methods yet by using a smaller number of queries. Both meta-attack and its "lazy" version significantly outperform the other methods in terms of the query efficiency, indicating the generalization capability of the meta-attack model from known whitebox neural networks to unknown blackbox networks. Similar to

untargeted attack, we achieve comparable results on success rate and average ℓ_2 distortion using a smaller number of queries on a targeted attack as shown in Table 5.7.

Table 5.7: Comparison of several methods under targeted attack on MNIST and CIFAR-10. Similar to the untargeted attack, we reduce the number of queries for meta attack.

Dataset / Target model	Method	Success Rate	Avg. L_2	Avg. Queries
MNIST / Net4	Zoo [16]	1.00	2.63	23,552
	Decision Boundary [2]	0.64	2.71	19,951
	AutoZoom [163]	0.95	2.52	6,174
	Opt-attack [18]	1.00	2.33	99,661
	Meta attack [34]	1.00	2.66	1,299
	Lazy meta-attack (ours)	1.00	2.63	**1,108**
CIFAR10 / Resnet18	Zoo [16]	1.00	0.55	66,400
	Decision Boundary [2]	0.58	0.53	16,250
	AutoZoom [163]	1.00	0.51	9,082
	Opt-attack [18]	1.00	0.50	121,810
	FW-black [15]	0.90	0.73	6,987
	Meta attack [34]	0.93	0.77	3,667
	Lazy meta-attack (ours)	0.92	0.69	**3,092**

CHAPTER 6: FACE DETECTOR ADAPTATION UNDER UNSUPERVISED AND SEMI-SUPERVISED SETTING

Face detection is often the very first step in analyzing faces. Recent literatures [102, 121, 188, 179] demonstrate the effectiveness of deep learning for face detection. However, as a massively data-driven method, the deep learning based face detectors are inevitably biased accordingly to the training data distribution. Collecting a comprehensive dataset for training can be highly expensive, if not impossible. Besides, considering the limited computational budget in real-world applications, arguably, there is no single face detector that fits all scenarios.

To address the discrepancy between the data distribution in training and the deployment of the face detector, it is highly desirable to have some adaptation mechanism built for the face detectors. When there are labeled *or unlabeled* images available from a particular target domain, one can adapt the detectors to achieve better performance in the target domain than the original ones do.

In this chapter, we propose a novel face detector adaptation approach [72] [1]that is applicable whenever the target domain supplies many representative images, no matter they are labeled or not. It entails some very interesting properties which we contend are missing or not explicitly discussed in the previous works of adapting face detectors [70, 170, 87].

First of all, our approach is designed to *avoid negative transfer*, i.e., the adapted detector is supposed to perform better than or at least on par with the original one in the target domain. It is worth noting that the negative transfer frequently occurs in domain adaptation [21, 53, 107], being a notoriously hard problem to solve. Moreover, this problem is likely more severe in the face detector

[1]This chapter's material has been accepted in 2018 in Conference on Computer Vision and Pattern Recognition (CVPR 2018), "Deep Face Detector Adaptation without Negative Transfer or Catastrophic Forgetting" authored by Muhammad Abdullah Jamal, Haoxiang Li and Boqing Gong.

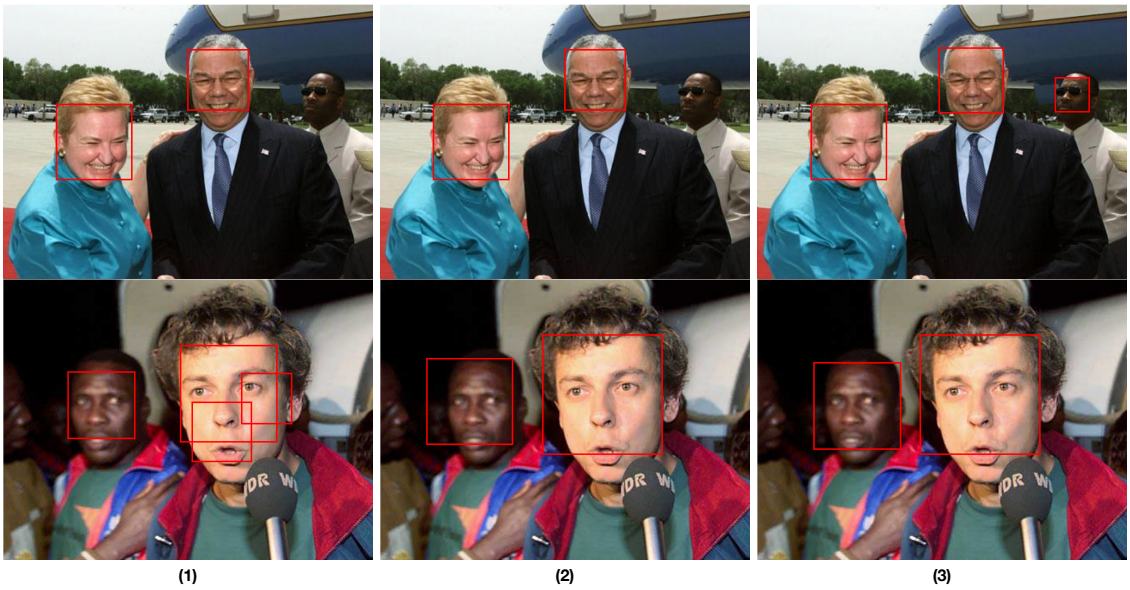


Figure 6.1: From left to right are face detection results on the FDDB dataset with **a state-of-the-art face detector (1)** [133, 74], the same detector but adapted by our method to the target domain (FDDB) **with no data annotation (2)**, and **with some data annotations (3)**.

adaptation since the room to improve the state-of-the-art face detectors is actually very small — for the same reason, we argue that it is vital for a face detector adaptation algorithm to explicitly take account of the negative transfer caveat.

Besides, we do not rely on the source data to conduct the adaptation, in a sharp contrast to most domain adaptation methods for generic visual recognition [56, 138, 54]. Indeed, the face detector adaptation is supposed to be done *without accessing the source data* because the source datasets are often extremely large and contain sensitive identity information. We note that some existing works on face detector adaptation [87] actually follow this protocol.

At last but not the least, we strive to *prevent our approach from catastrophic forgetting* or the so called interference [55, 114, 113] with the source domain. In this sense, our method is analogous to the well-known language model interpolation [76] where one extends the old language model

by interpolating it with the one trained for a new domain such that, in expectation, the resulting model performs well on all old domains as well the new domain. As such, our approach may also open an alternative direction for training the face detectors, namely, one can progressively improve the face detectors by growing the number of new domains without the need of keeping the images of the old domains.

We adapt a deep learning based face detector by fine-tuning [139, 182] it using both labeled and unlabeled images of the target domain. In order to avoid the negative transfer, we devise a loss function to approximate the expected performance improvement from the old detector to the new one. Since the hypothesis space — the set of networks specified by the weights — is the same for the two detectors, to minimize the loss does not change the old detector unless it finds another network that is expected to perform better than the old one in the target domain. While the expected performance gain of a network is mainly estimated by labeled data, we also augment it by deriving a closed form of the network's worst possible performance degradation that can be estimated by the unlabeled images of the target domain.

Our approach shares some spirits with AdaBoost [46] and residual learning [64] in the sense that the cost function of interest is a residual with respect to the source detector. Arguably, the residual loss is best captured by a residual detection score. Hence, we construct the target detector by an offset to the source one. Jointly, the residual loss and the offset detection score alleviate the urge of updating the weights of the old detector, effectively reducing the effect of catastrophic forgetting about the source domain.

The main contributions of this paper include both the novel adaptation approach and the three key properties of our method (cf. above) which we contend are missing from the previous works and yet are supposed to be possessed by a good face detector adaptation algorithm.

6.1 Approach

A face detector usually consists of two components: proposing candidate face regions from an image and classifying or scoring the regions. In this work, we adapt deep convolutional neural networks based face detectors to a given target domain by calibrating the second component, i.e., the classifiers. For simplicity, we express a deep face detector (e.g., [74]) as $\sigma(\mathbf{w}^T F(\mathbf{x};\theta))$, where $\sigma(z) = (1+\exp(-z))^{-1}$ is the sigmoid function indicating how likely the region proposal $\mathbf{x}$ out of an image is a face. The feature representations $F(\mathbf{x};\theta)$ of this region is extracted by a convolutional neural network, where θ collects all the network parameters except the classifier weights $\mathbf{w}$. Given such a detector pre-trained in the source domain, our goal is to adapt it to the target domain without using any source data and that the adapted face detector $\sigma(\widetilde{\mathbf{w}}^T F(\mathbf{x};\widetilde{\theta}))$ is not hurt by negative transfer or catastrophic forgetting.

In order to facilitate the adaptation to the target domain, we need the access to some representative images of that domain. We envision that a real use case of the face detector adaptation entails many unlabeled target images and yet only a small number or even none of labeled ones. Our approach takes account of both scenarios.

6.1.1 Unsupervised Face Detector Adaptation

We first consider the unsupervised face detector adaption in which we have access to the proposed regions $\{\mathbf{x}_t\}_{t=1}^T$ of the target domain but not their labels — the labels $\{y_t \in \{0,1\}\}$ are unknown. The objective is to obtain a high-quality face detector $\sigma(\widetilde{\mathbf{w}}^T F(\mathbf{x};\widetilde{\theta}))$ for the target domain using the pre-trained face detector $\sigma(\mathbf{w}^T F(\mathbf{x};\theta))$ and the unlabeled images of the target domain.

Our approach is originally motivated by the works on safe semi-supervised learning [95, 94, 108], where the idea is to trust the classifier pre-trained from the labeled data as much as possible and to

improve upon it only *relatively*. In our context, the relative performance change for any data point $(\mathbf{x}_t, y_t)$, $y_t \in \{0, 1\}$, of the target domain is

$$\begin{aligned}\textsc{res}_t(\widetilde{\mathbf{w}}, \widetilde{\theta}) := & \mathcal{C}\big(y_t, \sigma(\widetilde{\mathbf{w}}^T F(\mathbf{x}_t; \widetilde{\theta}))\big) \\ & - \mathcal{C}\big(y_t, \sigma(\mathbf{w}^T F(\mathbf{x}_t; \theta))\big),\end{aligned} \tag{6.1}$$

where $\mathcal{C}(y, \hat{y})$ is a performance measure, which is implemented as the multi-class classification accuracy in [94], top-k precision, F-score, and area under the ROC curve in [95], and log-likelihood in [108]. We instead use the cross-entropy $\mathcal{C}(y, \hat{y}) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$ in this paper. This choice seamlessly integrates it with the stochastic training procedure for deep neural networks.

When there are no labels available in the target domain, we find a robust target face detector that improves upon the source one under the worst case scenario,

$$\min_{\mathbf{u},\, \widetilde{\theta}} \frac{\lambda}{2} \|\mathbf{u}\|_2^2 + \mathbb{E}_t \max_{y_t \in \{0,1\}} \textsc{res}_t(\mathbf{w} + \mathbf{u}, \widetilde{\theta}), \tag{6.2}$$

where $\mathbb{E}_t$ denotes the mean average $\frac{1}{T}\sum_{t=1}^{T}$. We introduce this notation to stress the fact that the expected performance change from the old face detector to the adapted one can be unbiasedly estimated by the mean average over the target examples. We overload the notation y_t a little and use the fact that the groundtruth labels are binary. We also decompose the classifier of the target detector by $\mathbf{w} + \mathbf{u}$, where $\mathbf{w}$ are the parameters of the source detector's classifier. This decomposition is mainly for two reasons. First, we can interpret Eq. (6.1) as the residual between the performances of the two face detectors. Arguably, this quantity is accordingly best captured by the residual detection score between the two detectors. Hence, we re-parameterize the binary classifier of the target face detector as $\widetilde{\mathbf{w}} = \mathbf{w} + \mathbf{u}$. Second, notice that the ℓ_2 regularization over the offset weights $\mathbf{u}$ effectively constrains the classifier ($\widetilde{\mathbf{w}}$) of the target face detector around that ($\mathbf{w}$) of the source detector. This prevents the classifier from shifting around, taxing less than otherwise over

the network weights $\widetilde{\theta}$ for the overall target face detector to generate right predictions. Accordingly, the resultant representations $F(\mathbf{x};\widetilde{\theta})$ do not significantly deviate from the original representations $F(\mathbf{x};\theta)$ for the region proposal $\mathbf{x}$ of either source or target domain. In other words, the network *does not catastrophically forget* the knowledge extracted from the source domain.

To fit problem (6.2) to the existing deep learning tools (e.g., Tensorflow), we first note that there is an analytical solution to the inner maximization. Denote by $a_t = \sigma((\mathbf{w}+\mathbf{u})^T F(\mathbf{x}_t;\widetilde{\theta}))$, $\bar{a}_t = 1-a_t$, $b_t = \sigma(\mathbf{w}^T F(\mathbf{x}_t;\theta))$, $\bar{b}_t = 1-b_t$. We have the following,

$$\max_{y_t\in\{0,1\}} \quad \text{RES}_t(\mathbf{w}+\mathbf{u},\widetilde{\theta}), \quad \forall t \tag{6.3}$$

$$\begin{aligned} \Leftrightarrow \max_{y_t\in\{0,1\}} \quad & -y_t \log a_t - (1-y_t)\log \bar{a}_t \\ & +y_t \log b_t + (1-y_t)\log \bar{b}_t \end{aligned} \tag{6.4}$$

$$\begin{aligned} \Rightarrow \; y_t = 1 \text{ if } \log a_t + \log \bar{b}_t - \log \bar{a}_t - \log b_t < 0 \\ \text{and } y_t = 0 \text{ otherwise.} \end{aligned} \tag{6.5}$$

Next, we substitute the above back to Eq. (6.2) which then reduces to the canonical minimization problem and can be conveniently solved by programming the cost function using some off-shelf deep learning tools.

Eq. (6.2) is interesting in a few ways. The residual term indicates the relative loss by the target face detector with respect to the source detector. If, for the ease of discussion, we assume the adapted face detector performs about the same on all the target examples, then the residual is large only when the source face detector does a good job and correctly classifies the data point $(\mathbf{x}_t, y_t)$

— incurring small cross-entropy loss. The data points with small cross-entropy loss values by the source detector would be penalized more, because of their relative large residuals, than the other data in the optimization process. As a result, the new face detector is enforced to imitate the source detector: if a data point is correctly classified by the source detector's classifier, so should it be by the target detector.

In our experiments, we initialize the weights of the target face detector $(\widetilde{\theta}, \mathbf{w}, \mathbf{u})$ by the source detector $(\theta, \mathbf{w}, \mathbf{0})$. Hence, after solving Eq. (6.2), the new detector gives rise to no higher loss than the source face detector; the residuals are either negative or zero. As a result, there is *no negative transfer* to the target domain in expectation. Moreover, since we seek to minimize the residual loss for the worst possible label assignments (cf. $\max_{y_t}$ in Eq. (6.2)), the obtained detector is not worse than the source one (i.e., no negative transfer) for *any* label assignments to the region proposals $\{\mathbf{x}_t\}$.

We note that the search space of the possible label assignments in Eq. (6.2) could be reduced by imposing similar assumptions as in [70, 170, 87]. In particular, for the region proposals whose prediction scores are high (low) by the source face detector, we may assign 1's (0's) to them. The worst case label assignment would then be applied only to the regions of which the source detector is unsure. We leave this to the future work.

6.1.2 Supervised Face Detector Adaptation

In the supervised face detector adaptation, we are given a small set of labeled face images of the target domain $\{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ which is by itself insufficient for training a high-quality face detector. Following Eq. (6.2), it is now natural to write out the objective function under the supervised

setting as below,

$$\min_{\mathbf{u},\,\widetilde{\theta}} \quad \frac{\lambda}{2}\|\mathbf{u}\|_2^2 + \mathbb{E}_t\,\mathrm{RES}_t(\mathbf{w}+\mathbf{u},\widetilde{\theta}). \tag{6.6}$$

Note that the second cross-entropy term of Eq. (6.1) has no actual effect in the problem (6.6) — the minima of $(\mathbf{u},\widetilde{\theta})$ remain the same if we remove that term from Eq. (6.6). However, we keep it there for the ease of presentation.

6.1.3 Semi-supervised Face Detector Adaptation

Recall that we aim to adapt a pre-trained deep neural network based face detector to the target domain that supplies many unlabeled images and possibly some labeled ones. Indeed, a real use case of the face detector adaptation likely falls under this semi-supervised regime. In this case, we initialize the target detector by copying the weights from the source detector, and then alternate between the supervised and unsupervised adaptations in our training. In particular, we update the target face detector twice in each iteration by the gradients of eq. (6.6) and eq. (6.2), respectively.

6.2 Experiments

Our approach is model-agnostic, in the sense that it is readily applicable to different types of face detectors. In this section, we report extensive experimental results on two massively benchmarked deep face detectors.

6.2.1 Face Detectors and Source Domains

We experiment with two deep learning based face detectors: CascadeCNN [88] and Faster-RCNN [133, 74]. The CascadeCNN face detector is fast but extracts relatively weaker features while the Faster-RCNN model runs slower due to its use of a bigger network and more discriminative features.

In particular, CascadeCNN is trained by 25,000 faces from the AFLW dataset [112]. The Faster-RCNN face detector is trained using the training set of WIDER FACE dataset [179], which provides 32,203 images and 393,703 labeled faces with a high degree of variability in scale, pose, occlusion, etc. Per the comparison experiments in [74], the open-sourced Faster-RCNN face detector model is superior over 11 other top-performing detectors, all of which are published after 2015. Finally, it is interesting to note that both AFLW and WIDER FACE strive to cover a wide spectrum of face appearance variations, making them effective sources to adapt from.

6.2.2 Target Domain

The FDDB [69] dataset is a popular face detection benchmark. It contains 2,854 images and a total of 5,171 labeled faces. The images are randomly partitioned into 10 folds, of which we use the first six as our training set, the seventh for validation, and the remaining three for testing. We also evaluate our method on Caltech Occluded Faces in the Wild (COFW) dataset [13].

We claim that this choice — WIDER FACE or AFLW as the source domain and FDDB as the target domain — well represents the real application scenarios of face detector adaptation. On the one hand, there is a large training set in the source domain for us to learn a generic face detector that performs very well on different testing sets. WIDER FACE relies on diverse data sources since it employs Google and Bing to acquire the images and AFLW is a large-scale dataset collected from Flicker. On the other hand, the target domain of FDDB images are relatively homogeneous,

all sampled from the Yahoo! news website. They are mostly professional photos sharing some common idiosyncrasies.

6.2.3 Evaluation Metrics

Both WIDER FACE and FDDB datasets have defined and released the code for standard evaluation metrics. The Precision-Recall curve is used by WIDER FACE. FDDB employs the ROC curves of discrete and continuous scores computed from a bipartite graph. We use their code to evaluate our results in order to have direct comparison with existing methods.

6.2.4 Competing Methods

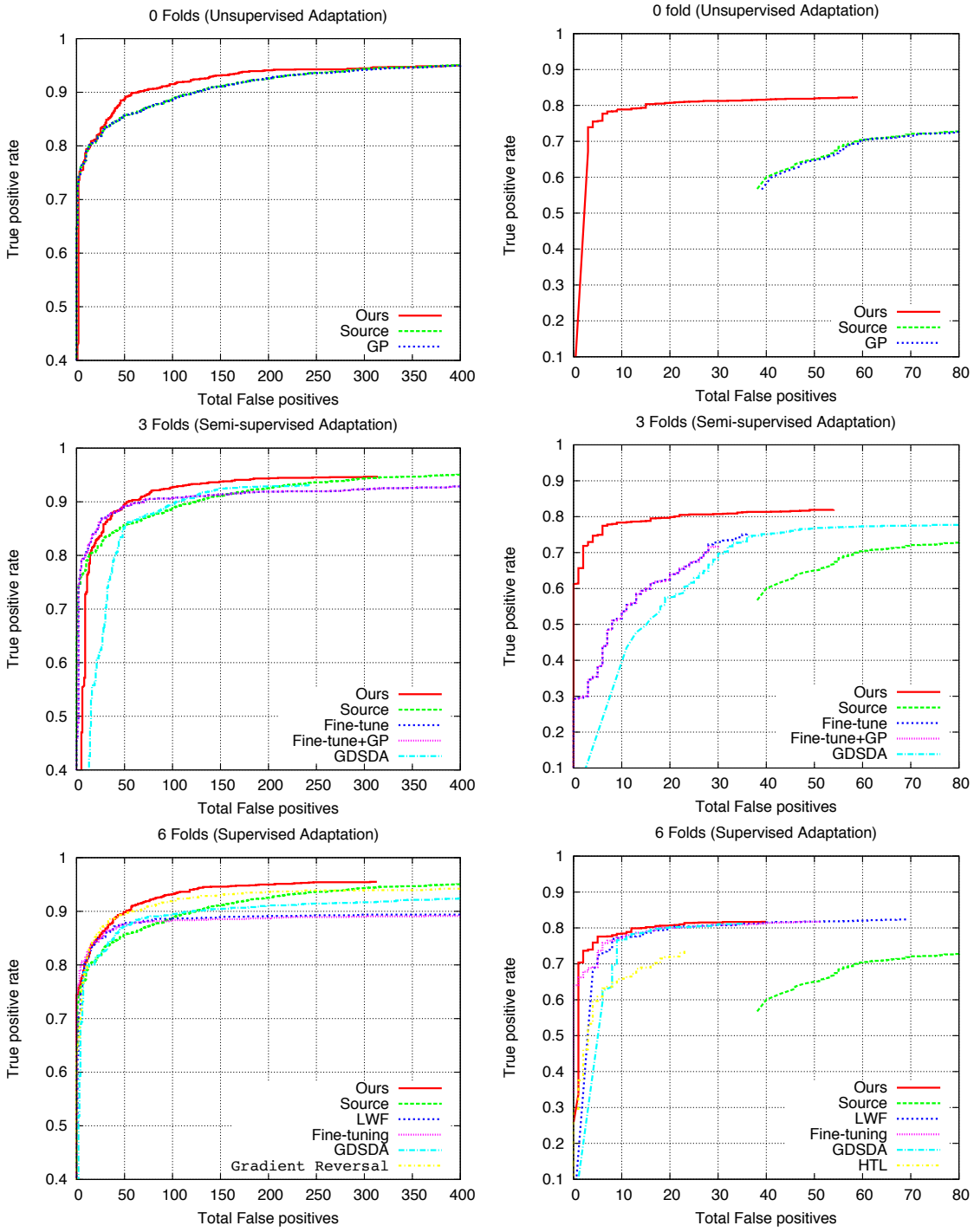
We compare our approach to the following competing baselines

- **Source** refers to the detectors trained from the original training data and is the starting point for our method to fine-tune the neural network parameters.
- **Fine-tuning [139]** simply fine-tunes the models using the labeled data of the target domain, if they are available, following the same way the detectors are trained in their source domains yet with smaller learning rates.
- **GP [70]** is a Gaussian process based unsupervised face detector adaptation method which uses the regions of high detection confidence — far from $p = 0.5$ — to update the detection scores of the other regions.
- **LWF [96]** is a recent learning without forgetting (LWF) method that augments the conventional cross-entropy loss with the knowledge distillation loss [65] such that the adapted face detector preserves the response characteristics learned from the source domain.

- **GDSDA [5]** introduces the generalized distillation [109] into semi-supervised domain adaptation.
- **HTL [84]** is a representative hypothesis transfer method that transfers knowledge from the source domain to the target by augmenting the feature representations of the target domain.
- *Gradient Reversal* [48] is an effective method for the domain adaptation of deep neural networks. The main idea is to learn representations to fail the classifier that predicts from which domain a data point comes. Since it has to access the source domain data, it is actually not fair to compare this method with the other baselines or ours. Nonetheless, we still include its results in the FDDB experiment for reference.

6.2.5 Implementation Details

We freeze the first eight convolutional layers of the Faster-RCNN model for all the experiments. We fine-tune all parameters of the last 48-net detection net in the CascadeCNN model. The validation set of the target domain is used to determine the hyper-parameters of all the methods. For Faster-RCNN, we use $\lambda = 1e$-3 and the base learning rates $1e$-4 and $5e$-4 for the supervised and unsupervised settings, respectively. Early stopping happens at the 5,000th iteration for the supervised experiment and the 6,000th for the unsupervised. For CascadeCNN, we set $\lambda = 2$ and the base learning rate $1e$-4 for both supervised and unsupervised settings. For the supervised case, we fine-tune the model for 8,000 iterations with the base learning rate and another 4,000 iterations with the learning rate of $1e$-5. For the unsupervised, we fine-tune the model for 10,000 iterations and divide the base learning rate by 10 at the 7,000th iteration. For all competing methods, the validation sets are used to determine all the free parameters.

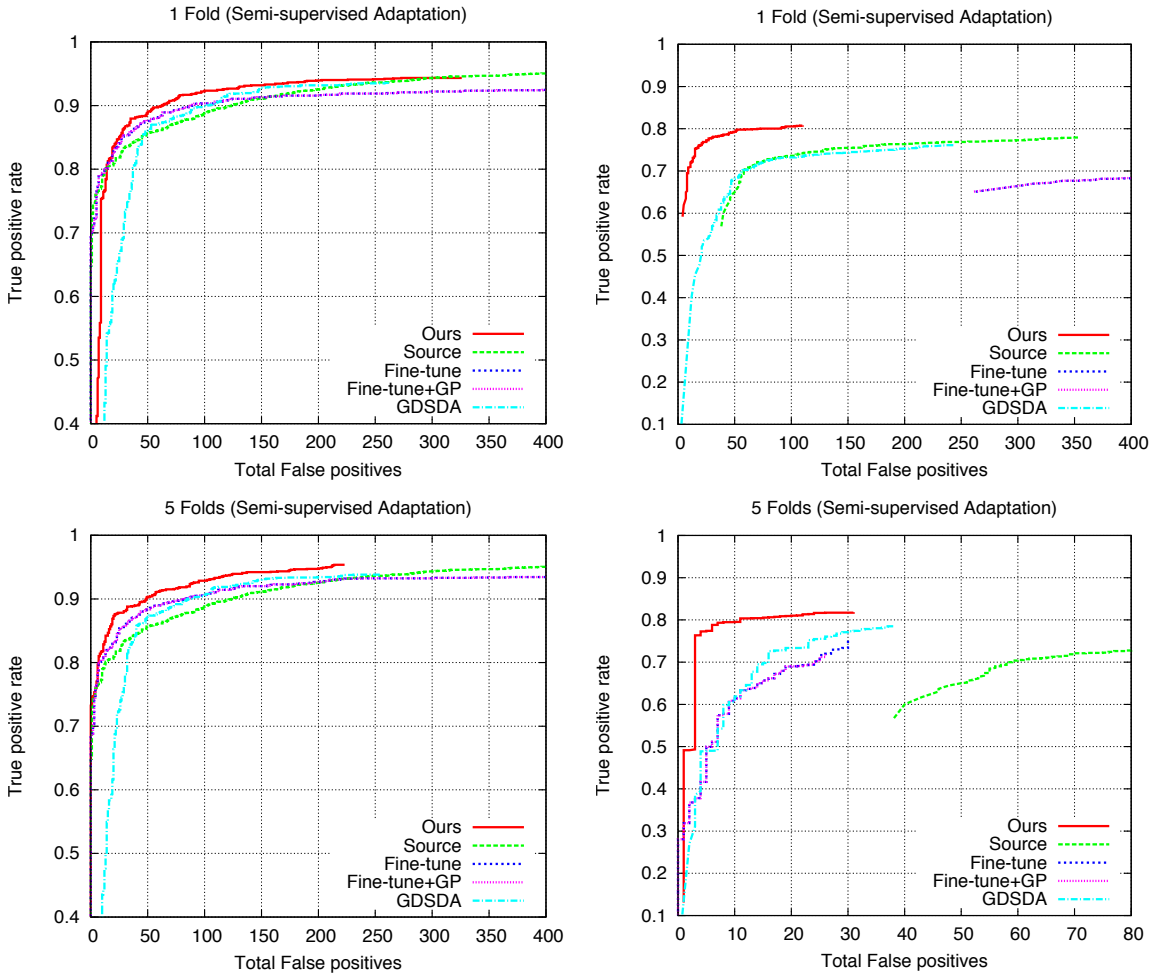


(a) FasterRCNN Detector Adaptation

(b) CascadeCNN Detector Adaptation

Figure 6.2: Detection results comparison on FDDB under unsupervised (0 out of 6 folds labeled), semi-supervised (3 out of 6 folds labeled), and supervised settings: our method generally outperforms all competing methods and does not suffer from negative transfer.

- **Fine-tuning [139]:** For Faster-RCNN, we run the experiments for 7,000 iterations in total. The base learning rate is $1e$-4 in the first 4,000 iterations and then reduced to $1e$-5 for the remaining 3,000 iterations. For CascadeCNN, we fine-tune the model with the learning rate of $1e$-4 for 10,000 iterations and another 5,000 iterations with the learning rate of $1e$-5.



(a) FasterRCNN Detector Adaptation (b) CascadeCNN Detector Adaptation

Figure 6.3: More detection results under semi-supervised settings with $N = \{1, 5\}$ out of 6 folds training images annotated. Combined with Figure 6.2, our method can generally bring additional performance gains from additional annotated data.

- **LWF [96]:** For Faster-RCNN, we train the model for 8,000 iterations. The base learning rate is set to $1e$-4 for the first 6,000 iterations and then reduced to $1e$-5 for the next 2000

iterations. For CascadeCNN, we train the model for 15,000 iterations with the base learning rate of $1e$-4 and reduce it to $1e$-5 at the 10,000th iteration. In both the experiments, the learning rate for the last layer is 10 times the base learning rate of the other layers.

- **GDSDA [5]:** We train the Faster-RCNN, and CascadeCNN with a learning rate of $1e$-4 for 8,000 iterations, and 12,000 iterations respectively.
- **HTL [84]:** We train the CascadeCNN using the learning rate $1e$-4 for 10,000 iterations and another 3,000 iterations using the learning rate $1e$-5.
- *Gradient Reversal [48]:* We train the Faster-RCNN using the base learning rate of $1e$-4 for 20,000 iterations and then reduce it to $1e$-5 for the next 10,000 iterations. Since the training sets of the source domain and the target domain are highly unbalanced, we alternatively take one image from either set to train the *Gradient Reversal*.

6.2.6 Comparison Results

We compare our algorithm with other competing methods in this section. We evaluate the effectiveness of all the methods by varying the number of labeled data from the target domain. More specifically, all the methods have access to the 6 folds of training images for the adaptation, while only N folds out of the 6 are labeled, $N \in \{0, 1, 3, 5, 6\}$. It is a fully unsupervised setting when $N = 0$, a semi-supervised adaptation setting when $1 \leq N \leq 5$, and a supervised adaptation setting when $N = 6$. Note that not all the baseline methods can handle all the settings.

Figure 6.2 and Figure 6.3 together show the ROC curves of the discrete scores on FDDB for the (a) CascadeCNN detector and (b) Faster-RCNN detector. When $N = 0$ (unsupervised adaptation), most of the above-mentioned competing methods are not applicable any more. As shown in Figure 6.2, in this challenging setting, we observe **GP** cannot improve the pre-trained high-quality

face detectors while our method still brings extra gains.

When $N = 6$, all the training images of the target domain are labeled (supervised adaptation), we outperform all the competing methods when adapting the CascadeCNN detector. Even for the high-quality FasterRCNN detector, our method gives rise to the largest improvement among all the methods, including *Gradient Reversal* which takes advantage of the extra training data in the source domain.

Under the semi-supervised setting, which is more realistic, our method achieves significant and consistent improvement for both face detectors over the original **Source** detectors. With the additional results shown in Figure 6.3[2], varying N from 0 to 6, our method generally performs better and better as more annotated data become available.

Overall, compared with **Source** models, our method does not cause negative transfer, while all the other competing methods suffer from negative transfer to some extent excluding *Gradient Reversal*.

Figure 6.8 and Figure 6.9 show the ROC curves of the continuous scores on FDDB for the Faster-RCNN and CascadeCNN. The left panels exhibit the curves of the Faster-RCNN and the right panels show the curves for the CascadeCNN. For CasacadeCNN, our approach outperforms all the competing methods in all the settings. For Faster-RCNN, our method can still boost the performance under the supervised setting and under semi-supervised setting when $N = 5$. More importantly, our approach does not incur negative transfer, i.e., the results are either better than or about the same as the source detectors. It is actually worth pointing out that the annotations are inconsistent between the FDDB dataset and the source where the detectors are trained. As a result, the FDDB under-evaluates the adaptation methods.

[2]The scale of the horizontal axis of the top-right panel differs from the other panels of CascadeCNN. If we used the same scale as the others instead, the fine-tuning results would be left out.

In addition to the FDDB dataset, we additionally consider COFW [13] as the target domain here. COFW provides 1,345 training faces and 507 testing faces and includes heavy occlusion and large shape variations.

Figure 6.4 and Figure 6.5 show the ROC curves of both discrete and continuous scores on COFW for both the Faster-RCNN and CascadeCNN face detectors under the supervised setting. We can draw the same observation as on the FDDB dataset, that our method shows no negative transfer as compared to other competing methods for both the Faster-RCNN and CascadeCNN detectors. GDSDA is an exception among the competing methods and leads to no negative transfer for CascadeCNN (but not for Faster-RCNN).

6.2.7 Ablation Study

We investigate our proposed method by examining its ablated versions. Recall that our approach is two-pronged. On the one hand, it uses the residuals in the cost function to explicitly prevent negative transfer in terms of the cross-entropy loss. On the other hand, it re-parameterizes the classifier of the target detector by $\widetilde{\mathbf{w}} = \mathbf{w} + \mathbf{u}$, where $\mathbf{w}$ is the classifier weights of the source detector.

Figure 6.6 shows that both components contribute to the performance improvement in our method. The ROC curve of the source detector is included for reference. Clearly, we observe that the two components mutually complement. Besides, removing the residual loss (Ours w/o residual loss) hurts our method more than directly optimizing the classifier weights $\widetilde{\mathbf{w}}$ without re-parameterization (Ours w/o residual score).

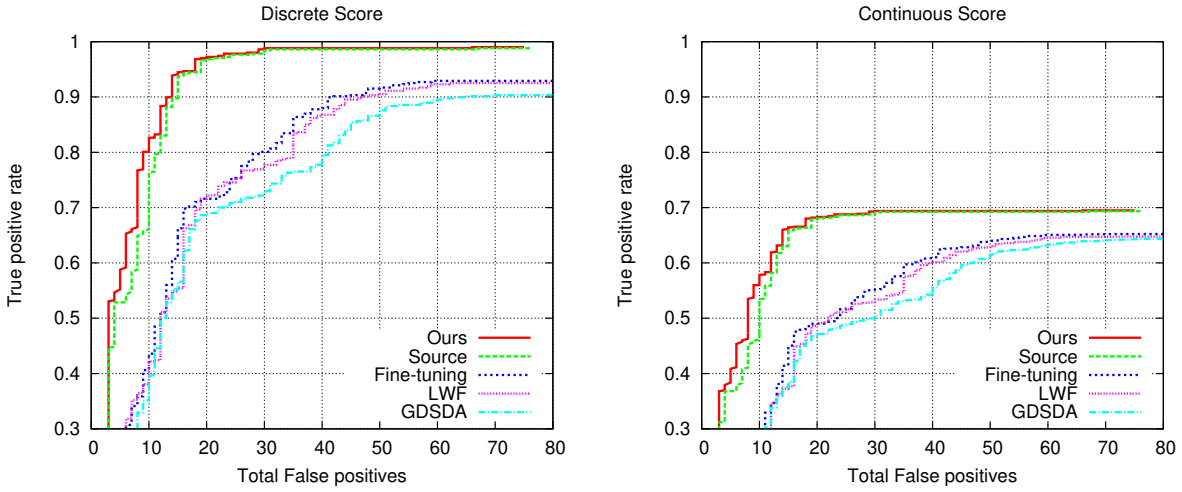


Figure 6.4: ROC Curves on COFW using **FasterRCNN** (supervised adaptation).

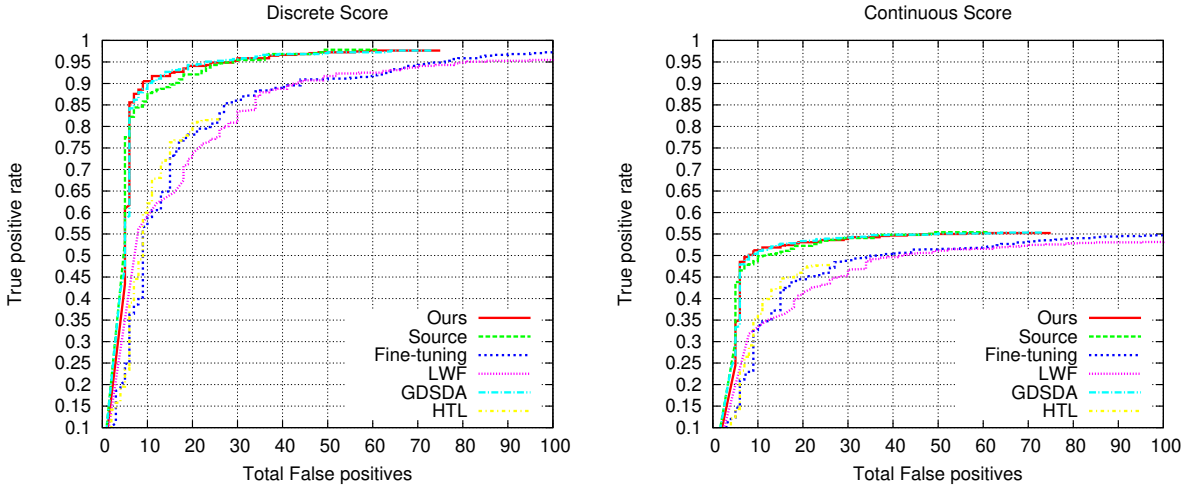
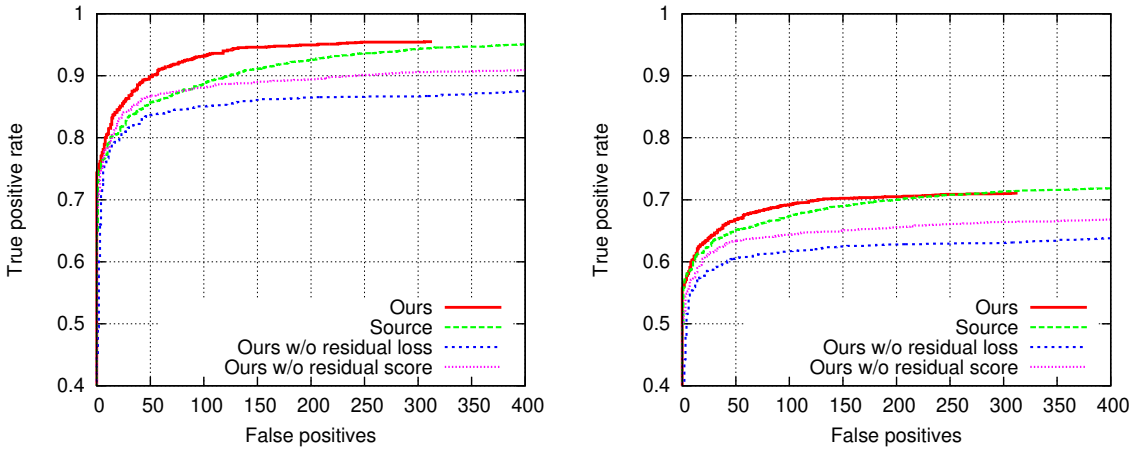


Figure 6.5: ROC Curves on COFW using **CascadeCNN** (supervised adaptation).

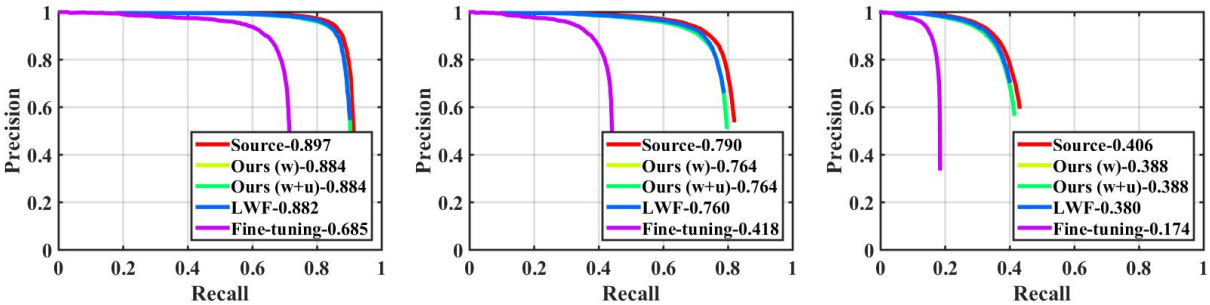
6.2.8 No Catastrophic Forgetting

Finally, we evaluate the catastrophic forgetting in the domain adaptation context. After adapting all competing methods to the target domain (FDDB), we evaluate their performance back to the



(a) Discrete Scores (b) Continuous Scores

Figure 6.6: Ablation Studies about our approach on FDDB (supervised adaptation)



(a) Easy Set (b) Medium Set (c) Hard Set

Figure 6.7: Evaluation of catastrophic forgetting on source domain after supervised adaptation to target domain: detection results on validation set of WIDER FACE (Easy, Medium and Hard sets).

source domain (WIDER Face). We test on the validation set of the WIDER Face in our experiment. **Source** refers to the one without adaptation and is thus with no forgetting at all.

As shown in Figure 6.10, it is not surprising to see that fine-tuning leads to severe forgetting about the source domain. This observation is well-aligned with prior arts. After all, domain adaptation can be seen as a special case of the sequential multi-task learning, under which previous studies have shown that fine-tuning causes catastrophic forgetting [55, 97]. Both **LWF** and our methods

maintain a reasonably good performance in the source domain compared with the **Source** detector. **LWF** prevents forgetting about the source domain using a knowledge distillation loss, while we do so by the residual loss coupled with the residual detection score. Thanks to the ℓ_2 regularization over the offset vector $\mathbf{u}$ in the classifier of the adapted detector, there is no noticeable difference between the new classifier ($\mathbf{w} + \mathbf{u}$) and that ($\mathbf{w}$) of the source face detector. We test both classifiers stacked over the network of the adapted detector and find that their corresponding curves almost overlap, as shown in Figure 6.10. We also evaluate the catastrophic forgetting when the detectors are adapted to COFW. Figure 6.10 shows the performance on the validation set of the WIDER Face for Faster-RCNN. Our approach maintains a good performance in the source domain compared with the original source detector.

6.2.9 More Qualitative Results

We show more qualitative results in Figure 6.11 and Figure 6.12. Our method is able to discard some of the false positives from both the source detectors. We can also observe that our method is able to detect the true positives that have not been detected by the source detectors.

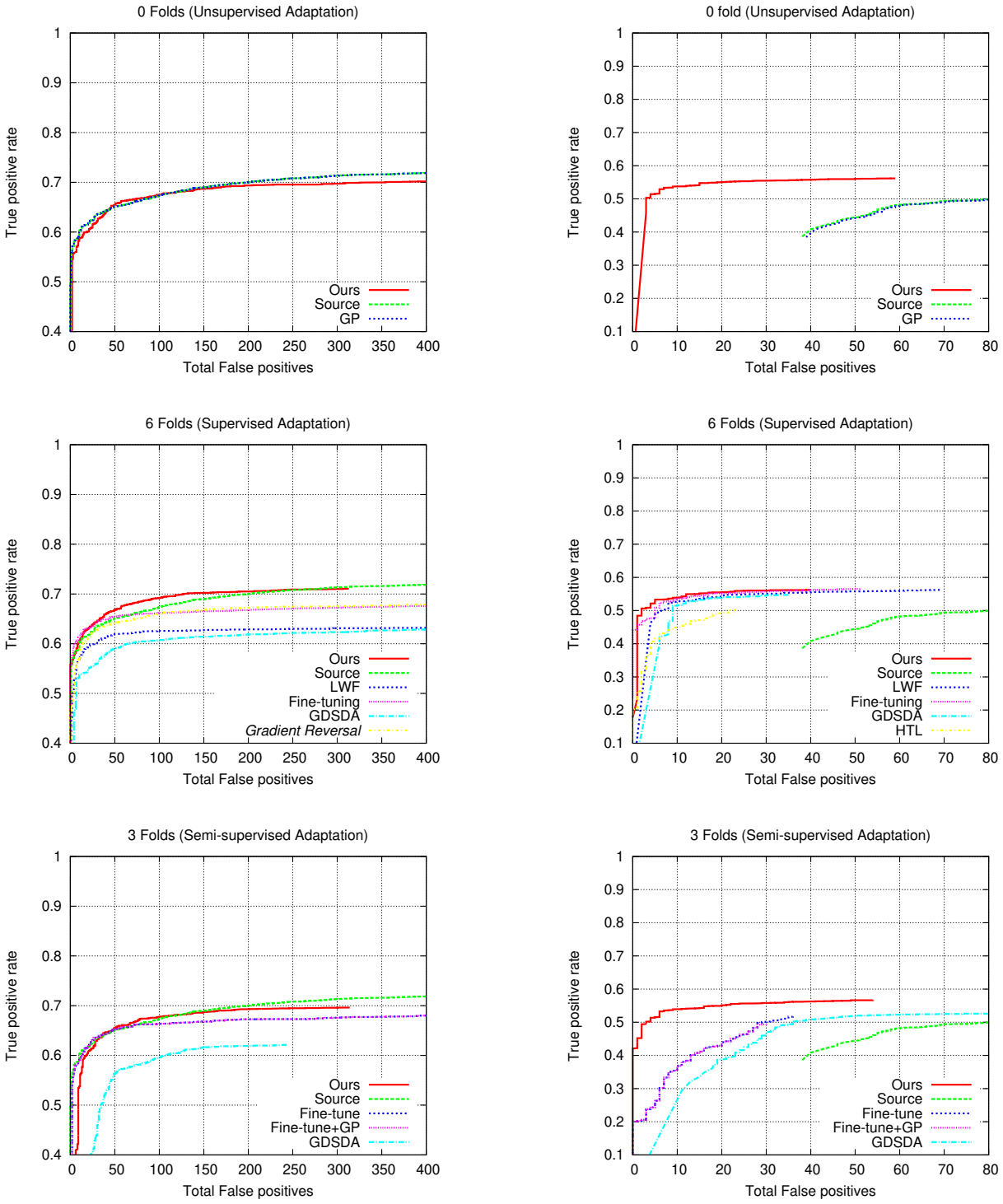


Figure 6.8: Continuous score results on the FDDB under unsupervised, supervised, and semi-supervised settings (3 out of 6 folds of training images annotated). (Left: Faster-RCNN, Right: CascadeCNN)

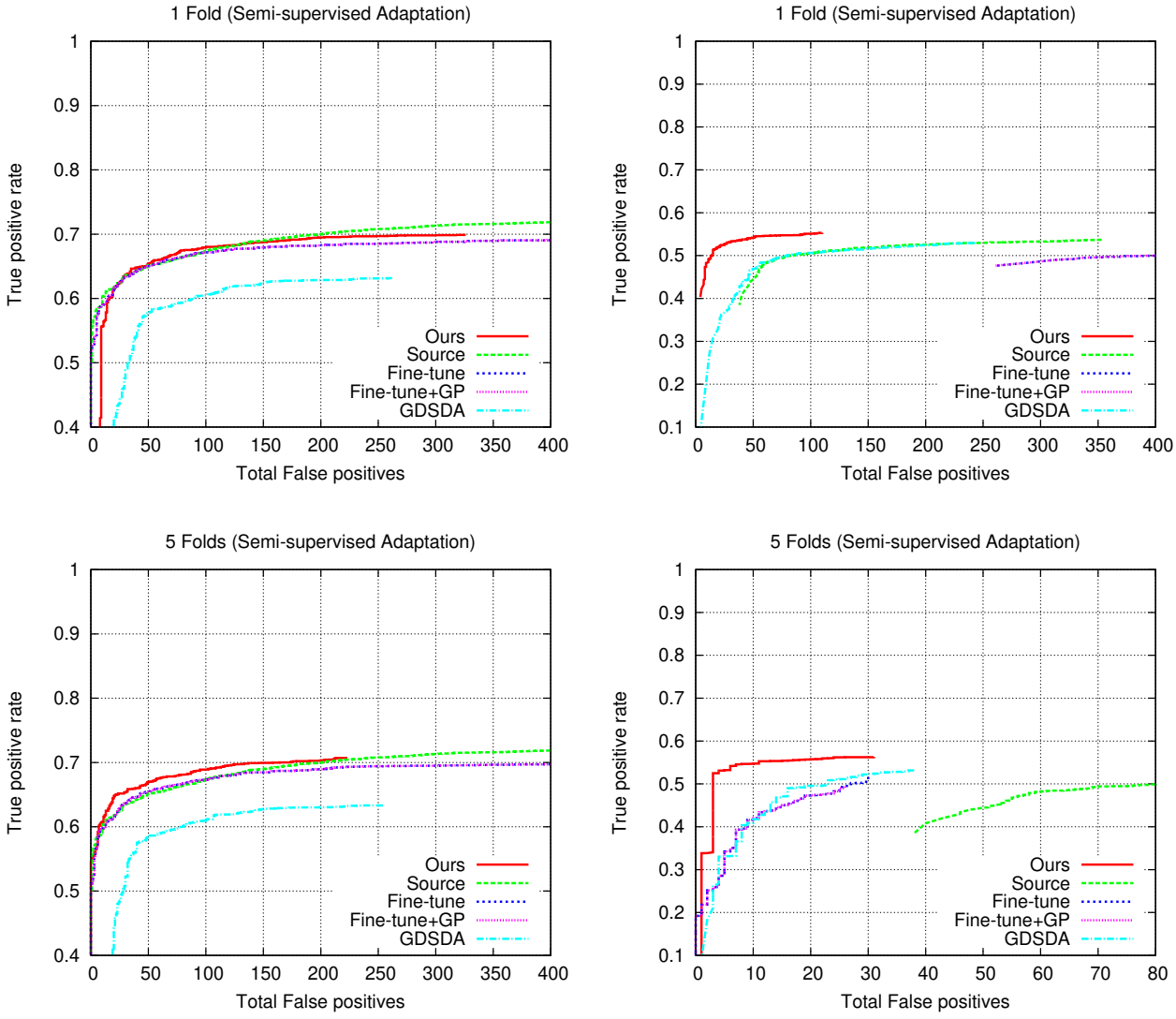
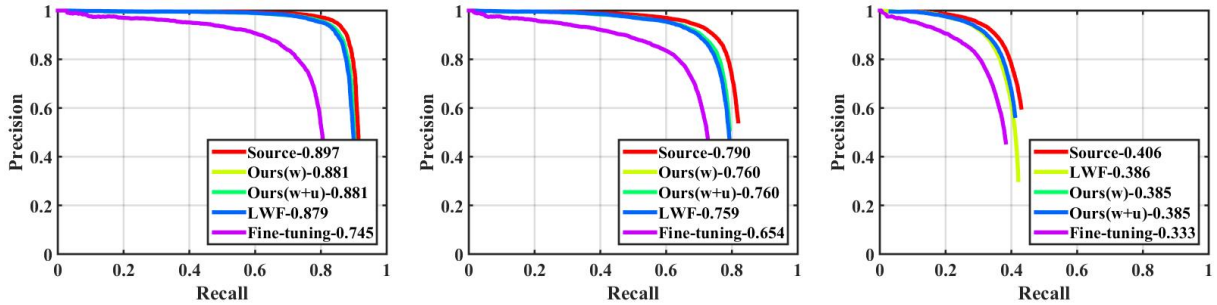


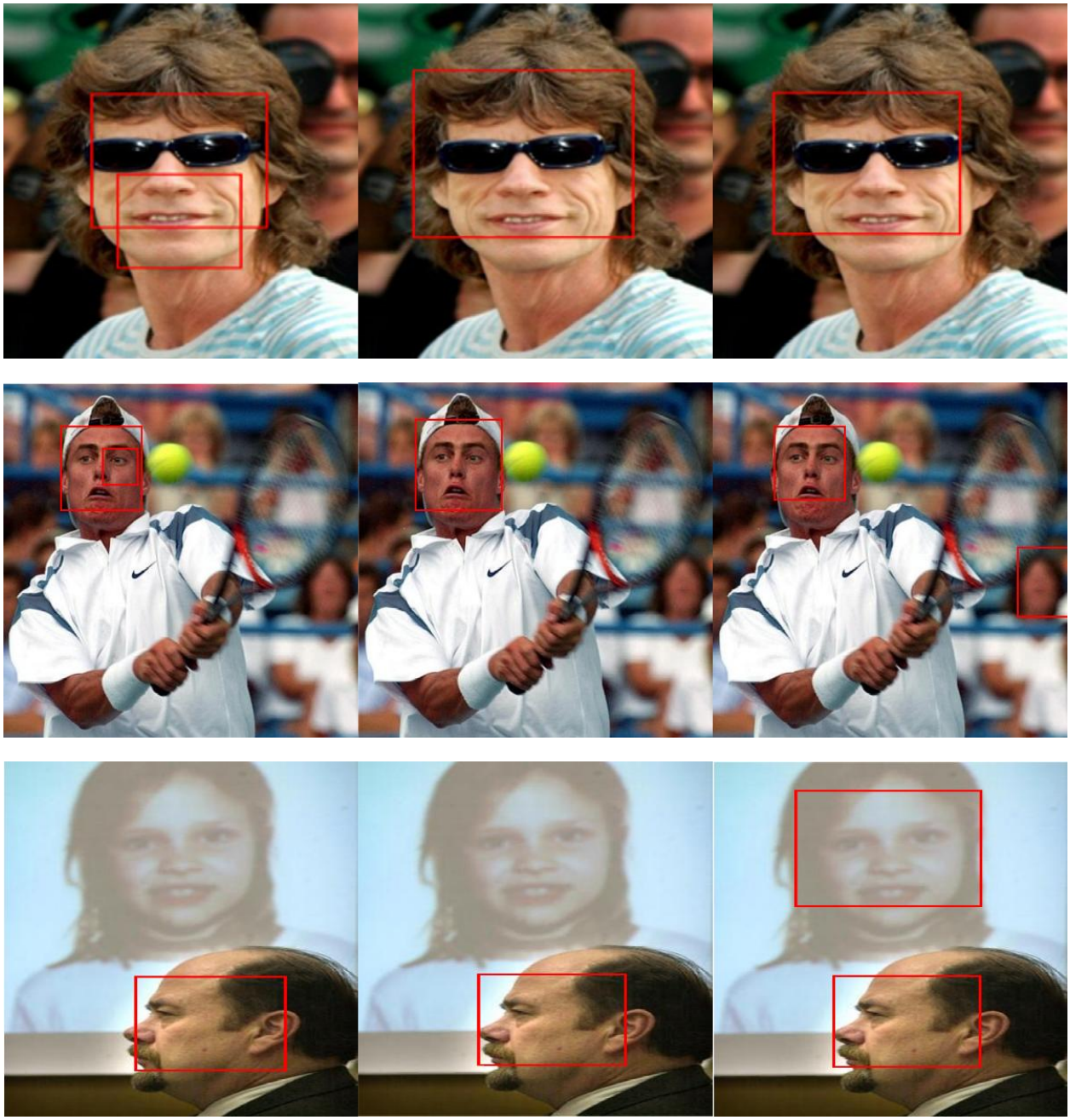
Figure 6.9: Continuous score results under the semi-supervised settings with $N = \{1, 5\}$ out of 6 folds training images annotated on the FDDB dataset. (Left: Faster-RCNN, Right: CascadeCNN)



(a) Easy Set (b) Medium Set (c) Hard Set

Figure 6.10: Evaluation of **catastrophic forgetting** on source domain after supervised adaptation to target domain (COFW): detection results on the validation set of WIDER FACE (Easy, Medium and Hard sets).

Figure 6.11: Qualitative results of adapting **Faster-RCNN**. The image on the left of each pair shows the detection results by the **source model** and the right image shows **our** method in the supervised adaptation setting.



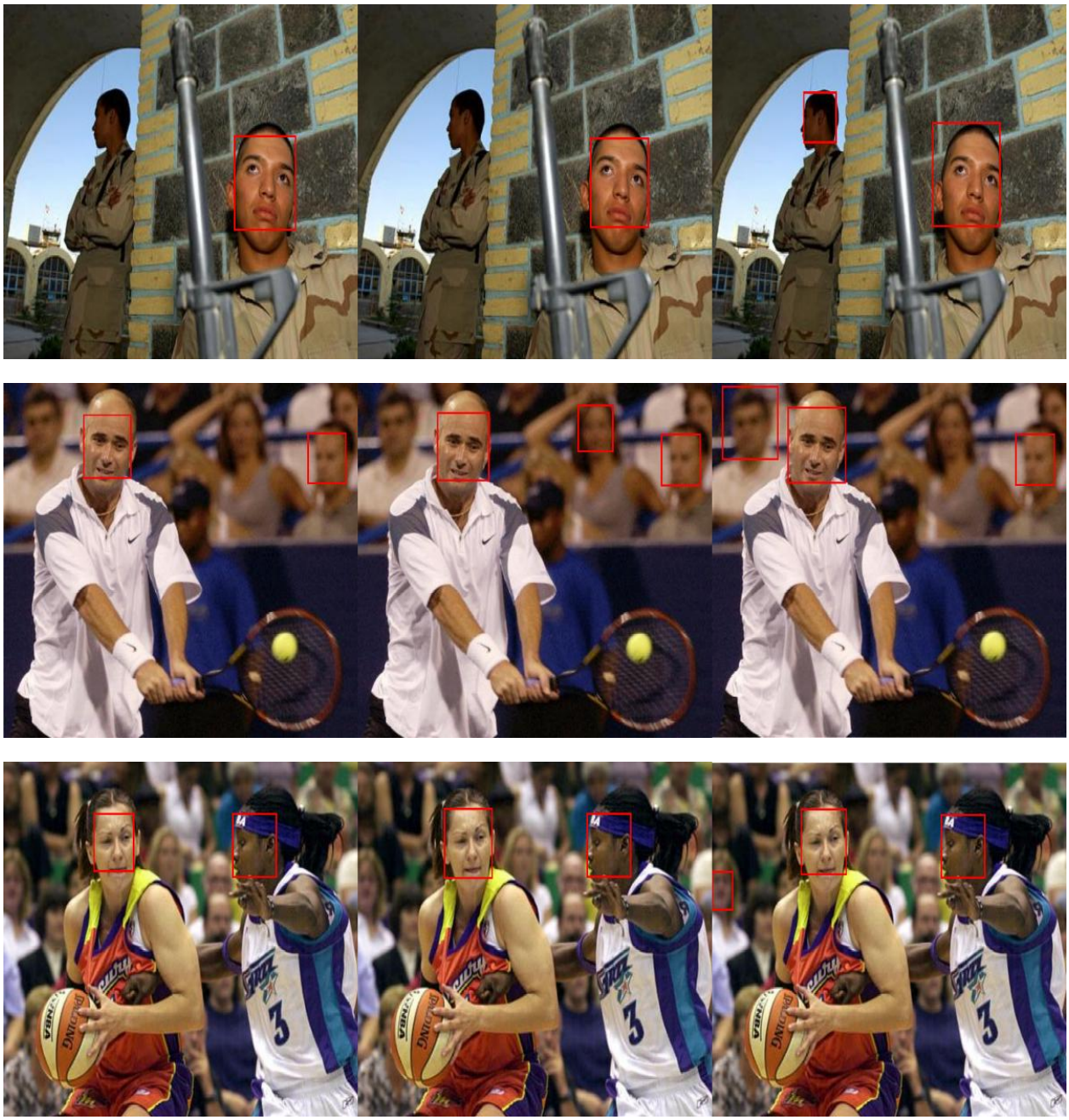


Figure 6.12: More qualitative results, from left to right are face detection results on FDDB dataset with **a CascadeCNN (1)**, the same detector but adapted by our method to the target domain (FDDB) **with no data annotation (2)**, and **with some data annotation (3)**.

CHAPTER 7: CONCLUSION AND FUTURE WORK

The success in computer vision tasks are largely dependent on large scale human curated datasets. But there are learning regimes where data collection and annotating them are difficult. In this dissertation, we studied and proposed different algorithms to learn beyond human curated datasets.

In Chapter 3, we proposed a novel paradigm of Task Agnostic Meta-Learning (TAML) algorithms for few-shot learning to train a meta-learner unbiased towards a variety of tasks before its initial model is adapted to unseen tasks. Both an entropy-based TAML and a general inequality-minimization TAML applicable to more ubiquitous scenarios are presented. We argue that the meta-learner with unbiased task-agnostic prior could be more generalizable to handle new tasks compared with the conventional meta-learning algorithms. The experiment results also demonstrate the TAML could consistently outperform existing meta-learning algorithms on both few-shot classification and reinforcement learning tasks.

In Chapter 4, we made two major contributions to the long-tailed visual recognition. One is the novel domain adaptation perspective for analyzing the mismatch problem in long-tailed classification. While the training set of real-world objects is often long-tailed with a few classes that dominate, we expect the learned classifier to perform equally well in all classes. By decomposing this mismatch into class-wise differences and the discrepancy between class-conditioned distributions, we uncover the implicit assumption behind existing class-balanced methods, that the training and test sets share the same class-conditioned distribution. Our second contribution is to relax this assumption to explicitly model the ratio between two class conditioned distributions.

Following these, in Chapter 5, we proposed a teacher-student scheme for the gradient-based meta-learning algorithms to allow them run more steps of inner updates to task-specific models while being immune to the risk of vanishing or exploding gradients. The student explores the tasks-

specific model's feasible space up to many steps, and the "lazy" teacher takes a one-step "leap" towards the region explored by the student. As a result, the teacher defines a lightweight computation graph and yet it takes advantage of the adequately explored checkpoints by the student. This approach is generic; we apply it to different problems, include few-shot learning, long-tail recognition, and meta-attack and various meta-learning methods.

Finally, in Chapter 6, we revisited the face detector adaptation problem under the new context of deep learning based face detectors. The approach we proposed offers three key properties which we contend are missing or not explicitly discussed in the existing face detector adaptation works. In short, the adaptation of face detectors is supposed to be executed in the absence of the source domain's data, with little negative transfer, and incurring no catastrophic forgetting about the source domain. Our approach explicitly accounts for all the requirements by two residuals: a residual loss to avoid negative transfer and a residual classifier to alleviate catastrophic forgetting.

7.1 Future Work

I'm interested in developing models that don't generally rely on human annotated large datasets, but can also learn representations without human supervision. There are three areas that are critical to this goal and natural next steps: first, improving the general meta-learning algorithms; second, improving self-supervised learning in few-shot learning in downstream tasks, and third, incorporating meta-learning in general domain adaptation problem.

7.1.1 Self-Supervised Learning

To avoid the excessive cost of human annotation, and collecting large-scale datasets, a promising sub-class of unsupervised visual learning called self-supervised learning [92, 30] has been

proposed recently. It requires unlabeled data to define a *pretext* task from which it can encode high-level semantic representation that are useful for downstream tasks. One can think of self-supervision as a prior for downstream tasks in fine-tuning stage. However, the fine-tuning becomes in-efficient when there are few labeled examples. On the other hand, meta-learning has been particularly successful in learning better prior or initialization for future fine-tuning. I'm interested in understanding self-supervised learning in this setting and wants to answer couple of questions. 1) Can we train a model in a self-supervised manner via meta-learning? 2) Meta-learning usually requires many training tasks for generalization. Can we generate these tasks via self-supervised learning from unlabeled data? I believe this is an important direction towards making the machine less reliable on labeled data.

7.1.2 Domain Adaptation

DA is an adaptation problem in which a goal is to learn a model from labeled source domain that can perform well on unlabeled or labeled data in target domain. The whole adaptation scenario can be analyzed from the meta-learning perspective. We can formulate meta-learning as a way of adaptation from seen classes in training set to unseen classes in test set where unseen classes have few examples. It has been successfully useful for few-shot learning recently where it learns an initialization for unseen tasks for faster adaptation. The question that I'm interested in is; can meta-learning be used as initialization for general domain adaptation? But, there are two problems that we must find solution for 1) Meta-learning computational graph is intractable for domain adaptation 2) In unsupervised domain adaptation, there is no access to the labels for the target domain whereas, meta-learning needs some supervision to define a learning loss for minimizing the meta-objective.

LIST OF REFERENCES

[1] iNaturalist 2018 competition dataset. https://github.com/visipedia/inat_comp/tree/master/2018, 2018.

[2] W. B. *, J. R. *, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[3] P. D. Allison. Measures of inequality. *American Sociological Review*, 1978. ISSN 00031224.

[4] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. *CoRR*, abs/1606.04474, 2016. URL http://arxiv.org/abs/1606.04474.

[5] S. Ao, X. Li, and C. X. Ling. Fast generalized distillation for semi-supervised domain adaptation. In *AAAI*, 2017.

[6] A. B. Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 1970.

[7] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the optimization of a synaptic learning rule. In *Optimality in Biological and Artificial Networks*. 1995.

[8] Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, 1991.

[9] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in Neural Information Processing Systems*, pages 181–189, 2010.

[10] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *CoRR*, 2018.

[11] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep), 2009.

[12] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *arXiv:1106.1813*, 2011.

[13] X. P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1513–1520, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.191. URL `http://dx.doi.org/10.1109/ICCV.2013.191`.

[14] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pages 1565–1576, 2019.

[15] J. Chen, D. Zhou, J. Yi, and Q. Gu. A frank-wolfe framework for efficient and effective adversarial attacks, 2018.

[16] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, 2017.

[17] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1431–1439, 2015.

[18] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference*

on Learning Representations, 2019. URL https://openreview.net/forum?id=rJlk6iRqKX.

[19] B. Chidlovskii, S. Clinchant, and G. Csurka. Domain adaptation in the absence of source domain data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 451–460. ACM, 2016.

[20] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[21] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In *International conference on algorithmic learning theory*, 2008.

[22] F. A. Cowell. Generalized entropy and the measurement of distributional change. *European Economic Review*, 1980.

[23] G. Csurka. *Domain adaptation in computer vision applications*. Springer, 2017.

[24] Y. Cui, Y. Song, C. Sun, A. Howard, and S. J. Belongie. Large scale fine-grained categorization and domain-specific transfer learning. *arXiv:1806.06193*, 2018.

[25] Y. Cui, M. Jia, T. Lin, Y. Song, and S. J. Belongie. Class-balanced loss based on effective number of samples. *arXiv:1901.05555*, 2019.

[26] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf. Fidelity-weighted learning. In *ICLR*, 2018.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[28] Y. Ding, L. Wang, D. Fan, and B. Gong. A semi-supervised two-stage approach to learning from noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1215–1224. IEEE, 2018.

[29] Y. Ding, L. Wang, H. Zhang, J. Yi, D. Fan, and B. Gong. Defending against adversarial attacks using random forest. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[30] Y. Ding, Y. Xu, S.-X. Zhang, Y. Cong, and L. Wang. Self-supervised learning for audio-visual speaker diarization. In *45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020)*, 2020.

[31] Y. Ding, L. Wang, and B. Gong. Analyzing deep neural network's transferability via frechet distance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3932–3941, 2021.

[32] Q. Dong, S. Gong, and X. Zhu. Class rectification hard mining for imbalanced deep learning. *arXiv:1712.03162*, 2017.

[33] C. Drummond and R. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats oversampling. *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets*, 2003.

[34] J. Du, H. Zhang, J. T. Zhou, Y. Yang, and J. Feng. Query-efficient meta attack to deep neural networks. *arXiv preprint arXiv:1906.02398*, 2019.

[35] L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9): 1667–1680, 2012.

[36] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016. URL http://arxiv.org/abs/1604.06778.

[37] H. Edwards and A. Storkey. *Towards a Neural Statistician*. 2 2017.

[38] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.

[39] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2009.

[40] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[41] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

[42] S. Flennerhag, P. G. Moreno, N. D. Lawrence, and A. Damianou. Transferring knowledge across learning processes. *arXiv preprint arXiv:1812.01054*, 2018.

[43] S. Flennerhag, A. A. Rusu, R. Pascanu, H. Yin, and R. Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019.

[44] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[45] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, Aug. 1997. ISSN 0022-0000. doi: 10.1006/jcss.1997.1504. URL http://dx.doi.org/10.1006/jcss.1997.1504.

[46] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.

[47] C. Gan, T. Yang, and B. Gong. Learning attributes equals multi-source domain generalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 87–97, 2016.

[48] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[49] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

[50] L. Ge, J. Gao, H. Ngo, K. Li, and A. Zhang. On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Statistical Analysis and Data Mining*, 7 (4):254–271, 2014.

[51] B. Gong, F. Sha, and K. Grauman. Overcoming dataset bias: An unsupervised domain adaptation approach. In *NIPS Workshop on Large Scale Visual Recognition and Retrieval*, volume 3. Citeseer, 2012.

[52] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.

[53] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, 2013.

[54] B. Gong, K. Grauman, and F. Sha. Learning kernels for unsupervised domain adaptation

with applications to visual object recognition. *International Journal of Computer Vision*, 109(1-2):3–27, 2014.

[55] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[56] R. Gopalan, R. Li, V. M. Patel, R. Chellappa, et al. Domain adaptation for visual recognition. *Foundations and Trends® in Computer Graphics and Vision*, 8(4):285–378, 2015.

[57] J. Gordon, W. Bruinsma, A. Y. Foong, J. Requeima, Y. Dubois, and R. E. Turner. Convolutional conditional neural processes. In *ICLR*, 2020.

[58] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJ_UL-k0b`.

[59] Y. Guo, Y. Li, R. Feris, L. Wang, and T. Rosing. Depthwise convolution is all you need for learning multiple visual domains. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[60] Y. Guo, Y. Li, L. Wang, and T. Rosing. Adafilter: Adaptive filter fine-tuning for deep transfer learning. In *34th AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[61] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[62] M. Hayat, S. H. Khan, W. Zamir, J. Shen, and L. Shao. Max-margin class imbalanced learning with gaussian affinity. *arXiv:1901.07711*, 2019.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[64] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[65] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[66] C. Huang, Y. Li, C. C. Loy, and X. Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.

[67] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *NeurIPS*, 2007.

[68] A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019.

[69] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, 2010.

[70] V. Jain and E. G. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. 2011.

[71] M. A. Jamal and G.-J. Qi. Task agnostic meta-learning for few-shot learning. In *CVPR*, 2019.

[72] M. A. Jamal, H. Li, and B. Gong. Deep face detector adaptation without negative transfer or catastrophic forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] M. A. Jamal, M. Brown, M.-H. Yang, L. Wang, and B. Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective, 2020.

[74] H. Jiang and E. G. Learned-Miller. Face detection with the faster R-CNN. *CoRR*, abs/1606.03473, 2016. URL `http://arxiv.org/abs/1606.03473`.

[75] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[76] D. Jurafsky and J. H. Martin. *Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2017. URL `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf`.

[77] L. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. *CoRR*, abs/1703.03129, 2017. URL `http://arxiv.org/abs/1703.03129`.

[78] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv:1910.09217*, 2019.

[79] E. Kazemi and L. Wang. A proximal zeroth-order algorithm for nonconvex nonsmooth problems. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 64–71. IEEE, 2018.

[80] E. Kazemi, T. Kerdreux, and L. Wang. Trace-norm adversarial examples. *arXiv preprint arXiv:2007.01855*, 2020.

[81] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[82] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[84] I. Kuzborskij and F. Orabona. Stability and hypothesis transfer learning. In *ICML (3)*, pages 942–950, 2013.

[85] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[86] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[87] H. Li, G. Hua, Z. Lin, J. Brandt, and J. Yang. Probabilistic elastic part model for unsupervised face detector adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[88] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, pages 5325–5334. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0. URL `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#LiLSBH15`.

[89] K. Li and J. Malik. Learning to optimize. *CoRR*, abs/1606.01885, 2016. URL `http://arxiv.org/abs/1606.01885`.

[90] R. Li and T. Zickler. Discriminative virtual views for cross-view action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2855–2862. IEEE, 2012.

[91] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[92] Y. Li, D. Huang, D. Qin, L. Wang, and B. Gong. Improving object detection with selective self-supervised self-training. In *European Conference on Computer Vision*, pages 589–607. Springer, Cham, 2020.

[93] Y. Li, X. Jia, R. Sang, Y. Zhu, B. Green, L. Wang, and B. Gong. Ranking neural checkpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2663–2673, 2021.

[94] Y.-F. Li and Z.-H. Zhou. Towards making unlabeled data never hurt. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1081–1088, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

[95] Y.-F. Li, J. T. Kwok, and Z.-H. Zhou. Towards safe semi-supervised learning for multivariate performance measures. In *AAAI*, pages 1816–1822, 2016.

[96] Z. Li and D. Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016. URL `http://arxiv.org/abs/1606.09282`.

[97] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[98] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017. URL `http://arxiv.org/abs/1707.09835`.

[99] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv:1707.09835*, 2017.

[100] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[101] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *TPAMI*, 2018.

[102] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang. Recurrent scale approximation for object detection in cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[103] Z. Liu, H. Zhang, B. Rao, and L. Wang. A reinforcement learning based resource management approach for time-critical workloads in distributed computing environment. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 252–261. IEEE, 2018.

[104] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019.

[105] Z. Liu, L. Wang, and G. Quan. Deep reinforcement learning based elasticity-compatible heterogeneous resource management for time-critical computing. In *49th International Conference on Parallel Processing-ICPP*, pages 1–11, 2020.

[106] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[107] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015.

[108] M. Loog. Contrastive pessimistic likelihood estimation for semi-supervised classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(3):462–475, 2016.

[109] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[110] D. Mahajan, R. B. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. *arXiv:1805.00932*, 2018.

[111] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[112] P. M. R. Martin Koestinger, Paul Wohlhart and H. Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[113] J. McClelland. A connectionist perspective on knowledge and development. 1995.

[114] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

[115] L. Metz, N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-dickstein. Learned optimizers that outperform on wall-clock and validation loss, 2019.

[116] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *arXiv:1310.4546*, 2013.

[117] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1DmUzWAW`.

[118] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

[119] T. Munkhdalai and H. Yu. Meta networks. *CoRR*, abs/1703.00837, 2017. URL `http://arxiv.org/abs/1703.00837`.

[120] D. K. Naik and R. J. Mammone. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 437–442 vol.1, Jun 1992. doi: 10.1109/IJCNN.1992.287172.

[121] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis. Ssh: Single stage headless face detector. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[122] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL `http://arxiv.org/abs/1803.02999`.

[123] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *CVPR*, 2017.

[124] E. A. Ok and J. Foster. Lorenz Dominance and the Variance of Logarithms. Technical report, C.V. Starr Center for Applied Economics, New York University, 1997.

[125] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning, 2018.

[126] E. Park and J. B. Oliva. Meta-curvature. In *Advances in Neural Information Processing Systems*, pages 3309–3319, 2019.

[127] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[128] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285–308, 1990.

[129] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

[130] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, 2015.

[131] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[132] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. *arXiv:1803.09050*, 2018.

[133] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[134] M. Riemer, E. Khabiri, and R. Goodwin. Representation stability as a regularizer for improved text analytics transfer learning. *arXiv preprint arXiv:1704.03617*, 2017.

[135] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, volume 898, 2005.

[136] P. M. Roth, S. Sternig, H. Grabner, and H. Bischof. Classifier grids for robust adaptive object detection. In *cvpr*, 2009.

[137] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[138] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

[139] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[140] E. Sangineto. Statistical and spatial consensus collection for detector adaptation. In *European Conference on Computer Vision*, pages 456–471. Springer, 2014.

[141] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. One-shot learning with memory-augmented neural networks. *CoRR*, abs/1605.06065, 2016. URL `http://arxiv.org/abs/1605.06065`.

[142] V. G. Satorras and J. B. Estrach. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJj6qGbRW`.

[143] J. 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`.

[144] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. 1992.

[145] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

[146] C.-W. Seah, Y.-S. Ong, and I. W. Tsang. Combating negative transfer from predictive distribution differences. *IEEE transactions on cybernetics*, 43(4):1153–1165, 2013.

[147] H. Shao, B. Tong, and E. Suzuki. Compact coding for hyperplane classifiers in heterogeneous environment. *Machine Learning and Knowledge Discovery in Databases*, pages 207–222, 2011.

[148] P. Sharma and R. Nevatia. Efficient detector adaptation for object detection in a video. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[149] H. Sheikh and L. Bölöni. Emergence of scenario-appropriate collaborative behaviors for teams of robotic bodyguards. In *to be presented at Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2019)*, May 2019.

[150] H. U. Sheikh and L. Bölöni. Reducing overestimation bias by increasing representation dissimilarity in ensemble based deep q-learning. *CoRR*, abs/2006.13823, 2020. URL `https://arxiv.org/abs/2006.13823`.

[151] H. U. Sheikh and L. Bölöni. Multi-agent reinforcement learning for problems with combined individual and team reward. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9206879.

[152] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[153] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

[154] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[155] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[156] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.

[157] M. Sugiyama, M. Krauledat, and K.-R. MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 2007.

[158] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. *CoRR*, abs/1711.06025, 2017. URL `http://arxiv.org/abs/1711.06025`.

[159] K. Tang, V. Ramanathan, L. Fei-Fei, and D. Koller. Shifting weights: Adapting object detectors from image to video. In *Advances in Neural Information Processing Systems*, pages 638–646, 2012.

[160] H. Theil. *Economics and information theory*. Studies in mathematical and managerial economics. North-Holland Pub. Co., 1967.

[161] S. Thrun and L. Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-8047-9.

[162] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[163] C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks, 2018.

[164] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The iNaturalist species classification and detection dataset. In *CVPR*, 2018.

[165] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

[166] O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016. URL `http://arxiv.org/abs/1606.04080`.

[167] D. Wang, Y. Li, L. Wang, and B. Gong. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[168] D. Wang, S. Zhang, and L. Wang. Deep epidemiological modeling by black-box knowledge distillation: An accurate deep learning model for covid-19. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[169] M. Wang and X. Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3401–3408. IEEE, 2011.

[170] X. Wang, G. Hua, and T. Han. Detection by detections: Non-parametric detector adaptation for a video. 2012.

[171] Y. Wang, A. Kucukelbir, and D. M. Blei. Robust probabilistic modeling with bayesian data reweighting. In *ICML*, 2017.

[172] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *NeurIPS*, 2017.

[173] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NeurIPS*, 2006.

[174] T. Weyand, A. Araujo, B. Cao, and J. Sim. Google landmarks dataset v2–a large-scale benchmark for instance-level recognition and retrieval. *arXiv preprint arXiv:2004.01804*, 2020.

[175] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein. Learned optimizers that scale and generalize, 2017.

[176] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

[177] W. Xing, Y. Yang, S. Zhang, Q. Yu, and L. Wang. Noisyotnet: A robust real-time vehicle tracking model for traffic surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2021. doi: 10.1109/TCSVT.2021.3086104.

[178] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *CVPR*, 2017.

[179] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[180] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

[181] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker. Feature transfer learning for face recognition with under-represented data. In *CVPR*, 2019.

[182] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[183] C. You, C. Li, D. P. Robinson, and R. Vidal. A scalable exemplar-based subspace clustering algorithm for class-imbalanced data. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *ECCV*, 2018.

[184] R. Yu, Y. Wang, Z. Zou, and L. Wang. Convolutional neural networks with refined loss functions for the real-time crash risk analysis. *Transportation research part C: emerging technologies*, 119:102740, 2020.

[185] H. Zhang, H. Zhang, C. Wang, and J. Xie. Co-occurrent features in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[186] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *ICML*, 2013.

[187] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9593–9604, 2019.

[188] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S3fd: Single shot scale-invariant face detector. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[189] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tailed training data. In *ICCV*, pages 5419–5428, 2017.

[190] Y. Zhang, P. David, H. Foroosh, and B. Gong. A curriculum domain adaptation approach to the semantic segmentation of urban scenes. *TPAMI*, 2019.

[191] Y. Zhang, P. David, H. Foroosh, and B. Gong. A curriculum domain adaptation approach to the semantic segmentation of urban scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):1823–1841, 2020.

[192] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2018.

[193] P. Zhou, X. Yuan, H. Xu, S. Yan, and J. Feng. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, pages 1532–1542, 2019.

[194] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

[195] Y. Zou, Z. Yu, B. V. Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.