Attribute transfer and prototype-based transductive methods for few-shot learning in visual object classification

by

Majed Alsadhan

B.S., Kansas State University, 2011

M.S., Kansas State University, 2014

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Abstract

Humans are capable of learning a specific task from few observations and examples because we have the ability of learning to learn. Modern artificial intelligence (AI) systems have achieved great performance in many settings and benchmarks, but this performance is achieved through large quantities of annotated samples such as labeled images. While it might be feasible to acquire such large quantities of labeled images in some application, it remains difficult to build such image corpora in others. This is due to the fact that labeled images may be unavailable and expensive for domain experts to annotate. Few shot learning (FSL) has been proposed to tackle the challenge of training an AI system with such a human-like capability. FSL aims to learn a classifier from a set of base classes with many available labeled data, and generalize to a set of novel classes with few available labeled data. Meta-learning frameworks, which follow the key idea of learning to learn, have been proposed for FSL. In such frameworks, FSL classification tasks are sampled from base categories, and a model is optimized to perform well in those tasks which then is used to classify tasks sampled from novel categories.

This dissertation introduces two contributions in the field of few-shot learning. One of these is a transductive framework that enables state-of-the-art FSL models to use intra-class attributes from within the model itself. The other contribution is a transductive metric-based iterative approach that achieves state-of-the-art accuracy in the field of few-shot learning.

The first contribution of this work addresses the problem of transfer learning between few-shot source and target domains, using synthetic attributes in addition to convolutional neural networks that are pre-trained on larger image corpora where no labeled instances of the target domains are present, though they may contain instances of their superclasses. Using a probabilistic transfer learning approach from predicted classes and inferred attributes, I develop a meta-learning ensemble method. I apply this method to examine the inference

approach: specifically, how it can extend and improve upon existing deep learning models for FSL, and how related probabilistic learning architectures can be adapted to use state-of-the-field deep learning components in this framework.

The second contribution of this work falls under the metric-based category of FSL. *Metric-based* approaches use a distance measure over embeddings extracted from convolutional neural networks. While there are many such proposed approaches, a simple one is to calculate similarities between embeddings of the query set and the mean embeddings of the support set (prototypes) using cosine similarity. My second contribution leverages the query set in a supervised learning setting to enhance those prototypes using convolutional neural networks that are pre-trained on base classes and fine-tuned using my iterative prototype-enhancement approach. Beyond the supervised learning setting, this second aspect of the work also studies the effect of leveraging an extra unlabeled set in a semi-supervised learning setting. Through extensive experiments, I show the positive effect of my iterative prototype-enhancement approach in terms of accuracy.

Attribute transfer and prototype-based transductive methods for few-shot learning in visual object classification

by

Majed Alsadhan

B.S., Kansas State University, 2011

M.S., Kansas State University, 2014

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Approved by:

Major Professor
William H. Hsu

Copyright

Abstract

Humans are capable of learning a specific task from few observations and examples because we have the ability of learning to learn. Modern artificial intelligence (AI) systems have achieved great performance in many settings and benchmarks, but this performance is achieved through large quantities of annotated samples such as labeled images. While it might be feasible to acquire such large quantities of labeled images in some application, it remains difficult to build such image corpora in others. This is due to the fact that labeled images may be unavailable and expensive for domain experts to annotate. Few shot learning (FSL) has been proposed to tackle the challenge of training an AI system with such a human-like capability. FSL aims to learn a classifier from a set of base classes with many available labeled data, and generalize to a set of novel classes with few available labeled data. Meta-learning frameworks, which follow the key idea of learning to learn, have been proposed for FSL. In such frameworks, FSL classification tasks are sampled from base categories, and a model is optimized to perform well in those tasks which then is used to classify tasks sampled from novel categories.

This dissertation introduces two contributions in the field of few-shot learning. One of these is a transductive framework that enables state-of-the-art FSL models to use intra-class attributes from within the model itself. The other contribution is a transductive metric-based iterative approach that achieves state-of-the-art accuracy in the field of few-shot learning.

The first contribution of this work addresses the problem of transfer learning between few-shot source and target domains, using synthetic attributes in addition to convolutional neural networks that are pre-trained on larger image corpora where no labeled instances of the target domains are present, though they may contain instances of their superclasses. Using a probabilistic transfer learning approach from predicted classes and inferred attributes, I develop a meta-learning ensemble method. I apply this method to examine the inference

approach: specifically, how it can extend and improve upon existing deep learning models for FSL, and how related probabilistic learning architectures can be adapted to use state-of-the-field deep learning components in this framework.

The second contribution of this work falls under the metric-based category of FSL. *Metric-based* approaches use a distance measure over embeddings extracted from convolutional neural networks. While there are many such proposed approaches, a simple one is to calculate similarities between embeddings of the query set and the mean embeddings of the support set (prototypes) using cosine similarity. My second contribution leverages the query set in a supervised learning setting to enhance those prototypes using convolutional neural networks that are pre-trained on base classes and fine-tuned using my iterative prototype-enhancement approach. Beyond the supervised learning setting, this second aspect of the work also studies the effect of leveraging an extra unlabeled set in a semi-supervised learning setting. Through extensive experiments, I show the positive effect of my iterative prototype-enhancement approach in terms of accuracy.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my family for their support during my time here at Kansas State University. Not only for their support during the time I spent in the PhD program, but for their support during the time I spent in the masters and bachelors program.

I also would like to thank my major professor Dr. William Hsu for his support and for his help refining this dissertation work. I've learned so many things from him and I am looking forward to learning more while I continue my research work with him for years to come.

Finally, I would like to thank all my committee members, Dr. Torben Amtoft, Dr. Mitchell Neilsen, and Dr. Caterina Scoglio for accepting to be members of my PhD committee. I would like to thank them for their valuable comments on my work and their valuable time.

Dedication

To the most loving person in the world, to my late sister.

Preface

While this dissertation work has been my own work and my own research, some of its content have been submitted for two different conference publications[10].

Chapter 1

Introduction

In this dissertation, I offer two contributions to the field of few-shot learning. One addresses the problem of transfer learning between few-shot sources and target domains, using synthetic attributes in addition to convolutional neural networks pre-trained on larger image corpora. In such scenarios, the original image corpora do not contain labeled instances of the target domains though they may contain instances of their superclasses. I developed a meta-learning ensemble method using a probabilistic inference approach from predicted classes and inferred attributes. This dissertation explains the inference approach, how it can extend and improve upon existing deep learning models for few-shot learning, and how to adapt related probabilistic learning architectures to use state-of-the-field deep learning components in this framework.

The second contribution that this work offers is a transductive supervised and semi-supervised approach that enhances the prototypes of a very popular and well-known metric-based meta-learning framework, the cosine-based prototypical networks that was proposed by Oreshkin et al. (2018)[11]. I developed a meta-learning approach that can boost the performance of cosine-based prototypical networks using supervised and semi-supervised learning in a transductive setting.

In this chapter, I briefly introduce FSL in Section 1.1. I then discuss meta-learning in Section 1.2. Section 1.3 provides a brief overview of transfer-learning, and Section 1.4

discusses the significance of FSL. The objective of this work is in Section 1.5, and the chapter concludes with a roadmap in Section 1.6.

1.1 Few Shot Learning

Humans can learn a specific task from a few observations because we can learn to learn[5]. This ability of learning-to-learn is unique to humans: we see a few pictures of an animal or an object that we have never seen before and can then identify the animal or an object the next time we see it. Modern artificial intelligence (AI) systems have achieved great performance in many settings and benchmarks. These AI systems, however, can only achieve this performance using large numbers of annotated samples like labeled images[12]. In some application domains, large numbers of labeled images might be easy to obtain, but obtaining so many labeled images is very difficult or almost impossible in other domains[12]. Labeled images might not be available, or expert annotation might be too expensive.

In order to be able to train an AI system with the human-like capacity to identify objects from only a few examples, few shot learning (FSL) has been proposed[4]. FSL attempts to learn a classifier from a set of base classes with many available labeled data, then generalizes to a set of unseen novel classes with few labeled data[12], usually FSL learning tasks consists of one or five images per class.

If FSL uses one image, it is called one-shot learning, and if it uses k images, it is called a k-shot learning. An example of a five-shot learning task is the following: given only a total of 15 labeled images of three different animals (five images per animal), and then given an unlabeled image of an animal that belongs to one of the 3-animal categories, Can you tell which category it belongs to? Such a task is easy for humans because humans can easily learn from only a few examples, but conventional AI systems would fail such a task unless trained in a special way because directly learning a large number of parameters with few samples might lead to overfitting[4].

1.2 Meta-learning

Meta-learning frameworks, which follow the key idea of learning to learn, have been proposed for FSL[4]. In the meta-learning framework, FSL classification tasks are sampled from base categories (this set is referred to as the meta-training set). Those tasks are then used to train a model that performs well, generalizing to tasks that are sampled from novel categories (this set is referred to as the meta-testing set). In such frameworks, a task T comprises K-way and N-shot (implying K classes and N support samples per K), and Q query samples for every class[4]. For semi-supervised learning, an extra set of U unlabeled samples is also sampled from every class. The ultimate goal is to use only the $N * K$ support samples (and the $U * K$ unlabeled samples in the case of semi-supervised learning) to classify all $Q * K$ samples into K classes[4]. Typically, small values of N are used to evaluate FSL techniques where $N \in \{1, 5\}$[5]. In subsection 1.2.1, I provide the problem definition of meta-learning. In subsection 1.2.2, I provide an example of what an FSL model consists of and how it is trained.

1.2.1 Problem Definition

In this subsection, I describe the problem of meta-learning for FSL using the same notations as Lee et al. (2019)[5] for FSL in supervised learning, but I modify it to generalize to semi-supervised learning as well. Given the following four sets:

1. Training set: $D^{train} = \{(x_t, y_t)\}_{t=1}^{S}$,
2. Testing set: $D^{test} = \{(x_t, y_t)\}_{t=1}^{Q}$,
3. Unlabeled set: $D^{Unlabeled} = \{(x_t, y_t)\}_{t=1}^{U}$
4. *Meta-training* set of tasks: $T = \{(D^{train}, D^{test}, D^{unlabeled}\}_{i=1}^{I}$

where D^{train}, D^{test}, and $D^{Unlabeled}$ are sampled from the same distribution. The goal of a base learner A is to estimate parameters θ of the predictor $y = f(x; \theta)$, so it generalizes well

to the unseen test set[5]. Meta-learning uses episodic sampling when given the following three sets:

1. the set of training classes (base): C^{train}
2. the set of validation classes: C^{val}
3. the set of testing classes (novel): C^{test}

An episodic sampling of task $T_i = (D_i^{train}, D_i^{test}, D_i^{Unlabeled})$, with samples (instances) each belonging to the *meta-training* set, is sampled as follows:

1. Using sampling without replacement, sample K of C^i categories from C^{train};
2. Using the set from 1, sample N images per category to obtain the support set $D_i^{train} = \{(x_n, y_n)|n = 1, ..., N * K, y_n \in C_i\}$;
3. Using the set from 1, sample Q images per category to obtain the query set $D_i^{test} = \{(x_q, y_q)|q = 1, ..., Q * K, y_q \in C_i\}$;
4. Using the set from 1, sample U images per category to obtain the unlabeled set $D_i^{unlabeled} = \{(x_u, y_u)|u = 1, ..., U * K, y_u \in C_i\}$,

where $D_i^{train} \cap D_i^{test} \cap D_i^{Unlabeled} = \varnothing$.

The same process is repeated to sample *meta-validation* from C^{Val} and *meta-testing* sets from C^{test}[5]. Then an FSL model can be trained using *meta-training*, validated using *meta-validation*, and tested using *meta-testing*.

1.2.2 Meta-Learning Example

Most state-of-the-art FSL models use a feature extractor or an encoder to extract embeddings. A feature extractor in vision deep-learning can be defined as a convolutional neural network (CNN) that parses the samples or images and converts them into a feature space or embeddings where each sample or image has its own unique embedding that represented

the image. This CNN can be a simple encoder that comprises only a few layers, or it can have an architecture that follows one of the state-of-the-art image classification models like VGG[13], AlexNet[14], and ResNet[15]. Most researchers use the first few layers of a ResNet as a feature extractor or an encoder because ResNet has outstanding performance.

From here, different state-of-the-art FSL models do different things with the embeddings of images. Some models that are considered to be metric-based models use a distance measure to compare images in the query set with images in the support set; then they use that measure to decide which image in the query set belongs to which class. Other models that are considered to be optimization-based models go further by using another model with trainable parameters and use that model's embeddings to determine classes for the query set.

My first approach in this dissertation work could be applied to both types of models, enabling these models to use attributes. My second approach focused on making the distance measure of a metric-based model more accurate by taking advantage of the query set and the unlabeled samples.

For example, in training a model using the meta-learning framework, a 5-shot/5-way scenario with the mini-ImageNet dataset could be used, and at each iteration, the model gives a batch size of 100-1000, where each batch consists of eight episodes or tasks. Each of the eight episodes comprises 25 images for the support set (5 * 5 = 25), 75 images for the query set (15 * 5 = 75), and approximately 275 images for the unlabeled set (55 * 5 = 275) for semi-supervised training, adding up to 3000 images (8 * [25+75+275]=3000). The limitation for the unlabeled set is determined by the data availability (a total of 600 images per class in case of mini-ImageNet, which means a total of 3000 images for five classes) or resource availability (approximately four 48GB-GPU's to fit eight episodes of this size). To an extent, the more episodes in a batch, the better the training for the model; usually eight episodes in a batch is the optimal number of episodes. At every iteration, the output of the query set (logits) are then calculated. These logits are used and fed into a loss function that calculate an error. This error is then back-propagated through the entire model, which is then optimized using an optimizer.

This process is repeated for a number of iterations where at the end of each iteration, the model is then validated using the meta-validation set which usually consists of 100-1000 batches and each batch consisting of one episode. The accuracy of the validation set is calculated, and the weights of the model at every iteration are saved. This concludes the training process.

When testing the model, weights of the iteration with the highest validation accuracy are loaded and used with the model. The model is tested on 1000-10,000 batches where each batch consists of one episode sampled from the novel categories or classes, which are different sets of classes were never present during training. After testing the model on the 1000-10,000 batches, the average accuracy is considered the final accuracy of the model.

1.3 Transfer Learning

In machine learning, transfer learning is the act of using knowledge learned from one particular situation and then applying the knowledge to another situation[16]. One example of transfer learning is transferring knowledge from a classifier trained to classify bees to another classifier that is then used to classify insects. While transfer learning can be represented and applied in many forms and ways in machine learning, one form that related to my work was transferring knowledge obtained from meta-training to meta-testing, as discussed in the previous section. Attribute transfer is another important form of transfer learning that relates to my work. In attribute transfer, a model is also given attributes to enrich its knowledge for the sole purpose of better classifying objects. These attributes are usually in textual form that represents visual features of the images. In most cases, every image would have more than one attribute.

1.4 Significance

In machine learning, FSL is an important problem for a number of reasons. One is learning for rare cases like drug discovery[17]. Another is to reduce data collection efforts. Less

data is needed to train an FSL model, so data collection and labeling costs can be reduced. Training models that can successfully perform in FSL is challenging, so transferring knowledge obtained from the meta-training step to the meta-testing step has succeeded using a meta-learning framework. We can further improve this knowledge transfer by obtaining supplemental knowledge from different sources like textual attributes that describe visual features of images. Another improvement is to use unlabeled samples to enrich the model's knowledge.

1.5 Objectives

In this section, I present my two objectives with their alternative and null hypotheses in subsections 1.5.1 and 1.5.2. I then discuss two novel contributions that this dissertation work includes in subsection 1.5.3. I conclude this section with the research questions in subsection 1.5.4.

1.5.1 First Objective

In this research, I hypothesized that when applying my attribute-transfer approach to current state-of-the-art FSL models to classify the mini-ImageNet and tiered-ImageNet datasets in a meta-learning framework, the performance of these models will improve because my proposed attribute-transfer approach is more generally useful. My alternative hypothesis is the following:

$H_A : Accuracy(baseline + my\ transfer) > Accuracy(baseline)$,

and my null hypothesis is the following:

$H_0 : Accuracy(baseline + my\ transfer) \leq Accuracy(baseline)$

1.5.2 Second Objective

In my research, I also hypothesized that applying my prototype enhancement approach to cosine-based prototypical network (CBPN) to classify the mini-ImageNet and tiered-

ImageNet datasets in a meta-learning framework will improve the performance of cosine-based prototypical networks because my proposed prototype enhancement approach enhances those prototypes so that the final prediction accuracy improves. Hence, my alternative hypothesis is the following:

$$H_A : Accuracy(CBPN + my\ prototype\ enhancement) > Accuracy(CBPN),$$

and my null hypothesis is the following:

$$H_0 : Accuracy(CBPN + my\ prototype\ enhancement) \leq Accuracy(CBPN)$$

1.5.3 Novel Contributions

There are two novel contributions that this research offers. These novel continuations are the following:

1. **I succeeded in showing that my technique of attribute-transfer derived from zero-shot learning can be applied to few-shot learning when used with a meta-learning framework.**

   Lampert et al. (2009)[9] proposed an attribute-transfer technique for zero-shot learning in 2009. This technique does not work out of the box when applied to state-of-the-art FSL models with mini-ImageNet and tiered-ImageNet in a meta-learning framework. I succeeded in developing a technique for attribute transfer that is derived from Lampert et al. (2009)[9] and adapted to FSL by using pre-training[4], probabilistic fusion[8], and metric scaling[11].

2. **I succeeded in developing a technique for enhancing prototypes of cosine-based prototypical networks using an enhancement loop in supervised and semi-supervised learning.**

   Inspired by the technique developed by Liu et al. (2020)[3], I developed a prototype enhancement technique that uses pre-training[4] and my own proposed enhancement loop and achieved better accuracy measures than in Liu et al. (2020)[3] when used with mini-ImageNet and tiered-ImageNet datasets in a meta-learning framework. My

technique can be used in both supervised and semi-supervised learning as I show in this work.

1.5.4 Research Questions

I answered several primary questions in detail in the results chapter of this dissertation. The following are the primary research questions:

1. How does applying attribute-transfer to existing state-of-the-art FSL models affect their performance?
2. How does varying the number of transferred attributes affect my attribute-transfer approach?
3. How does applying prototype enhancement to cosine-based prototypical network affect its performance in supervised learning?
4. Does using unlabeled samples with prototype enhancement improve the performance of cosine-based prototypical network?
5. For supervised prototype enhancement, how does varying the number of the query set affect the performance?
6. For semi-supervised prototype enhancement, what is the best ratio of query and unlabeled sets?

1.6 Roadmap

This dissertation is organized as follows: I first review prior research in few-shot learning as related to my work in Chapter 2 and highlight the challenges revealed by published research and how my two approaches overcome those challenges. I then discuss related research on attribute transfer, prototypes rectification, and probabilities fusion in Chapter 3. My methodology explains my first and second approaches and provides detail in Chapter

4. I explain my experimental design in Chapter 5, including the baselines I used in this research, the datasets and how they were sampled, and my experimental set up, which details everything needed for reproducible results and evaluation metrics. I then discuss the results and analysis from my experiments in Chapter 6. Finally, I present my conclusions and a summary of the main contributions of this research as well as remarks on future directions of research in Chapter 7.

Chapter 2

Related Work on Few Shot Learning

2.1 Few Shot Learning

Modern artificial intelligence (AI) systems have been highly successful in applications with large quantities of data. However, and due to overfitting, the very same systems might fail when using small data sets[12]. Recently, Few-Shot Learning (FSL) was created to tackle this problem[4]. FSL can generalize to new tasks with only few supervised samples by using prior knowledge[12]. Research into FSL can be categorized into 2 settings: inductive and transductive[3]. In the inductive setting[1;2;4;18–21], no knowledge is assumed about the query set, and each sample from that set is treated individually[3]. On the other hand, the transductive setting[3;22–29] allows us to assume and use the query set in a way that contributes to better accuracy of the model[3]. My research as reported in this dissertation uses the transductive setting because of its effectiveness. Related research in FSL as it relates to my research can be broadly placed into three approaches: metric-based[1;2;4;11], optimization-based[5;30], and semantics-based[6–8;31–33].

In the subsections 2.1.1, 2.1.2, and 2.1.3, I discuss each approach using examples from the literature.

2.1.1 Metric-based approaches

The metric-based approach uses the embeddings obtained from a feature extractor, and a model learns a distance-based prediction rule[5]. Matching networks[1] is early inductive research that uses the metric-based approach. This model was based on the idea of attention and memory using an LSTM network[1]. Figure 2.1 shows an illustration from Vinyals et al. (2016) demonstrating the technique[1].

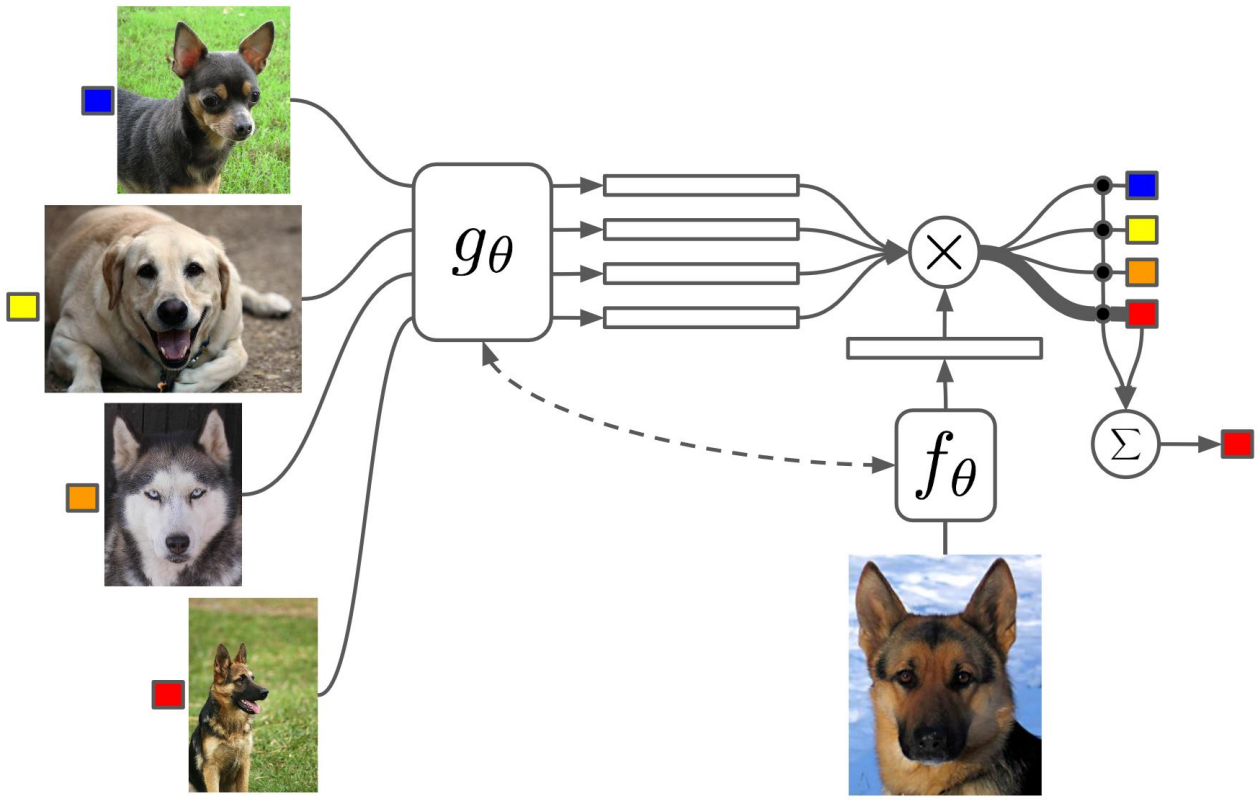

Figure 2.1: *Overview of the Vinyals et al. (2016)[1] approach.*

A prototypical network[2] is another inductive approach also considered metric-based. In a prototypical network, the support set is first used to calculate prototypes that represent the mean embedding of each class. Then the similarities between the query set and those prototypes are calculated using Euclidean distance. Figure 2.2 shows the illustration that Snell et al. (2017)[2] used in their paper to illustrate five support samples per class, three prototypes and one query sample.

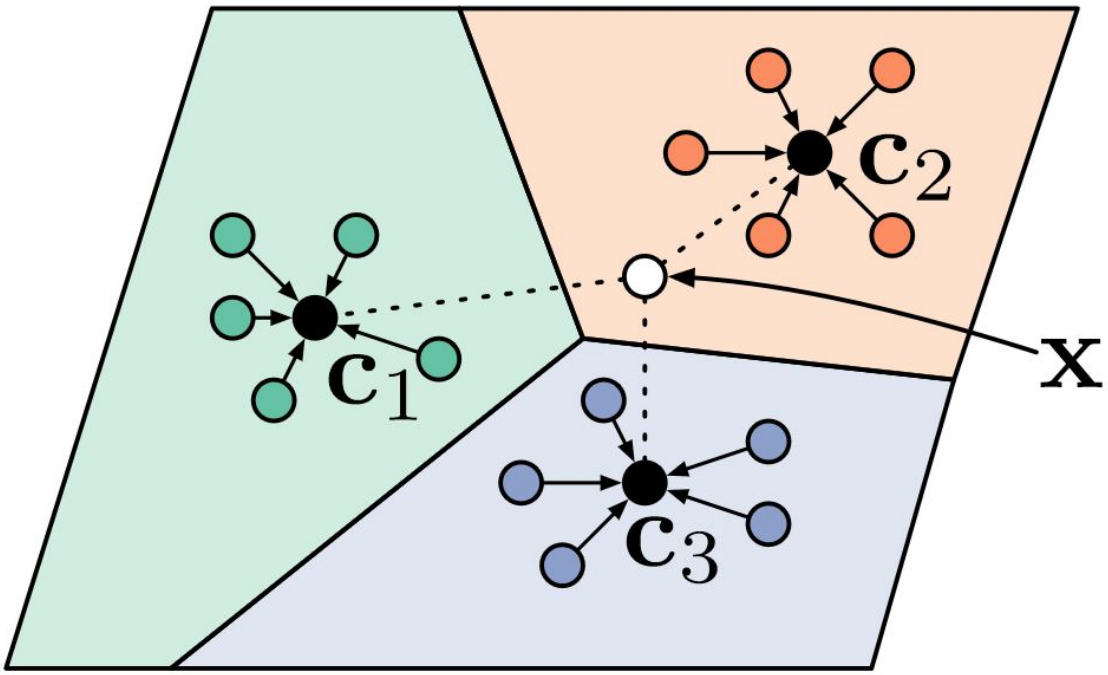

Figure 2.2: *Overview of the Snell et al. (2017)[2] representation of prototypes.*

Snell et al. (2017)[2] reported that they used Euclidean distance instead of cosine similarity because it more accurately trained their model[2]. Oreshkin et al. (2018)[11] have showed that scaling the cosine similarity can improve the performance of cosine-based prototypical network over the performance of Euclidean-based prototypical network. In my second approach, I used the cosine-based prototypical network that was proposed by Oreshkin et al. (2018)[11].

Liu et al. (2020)[3] proposed prototype rectification, a transductive metric-based approach closely related to my second approach. They used a cosine-based prototypical network like the one proposed by Oreshkin et al. (2018)[11]. In their research, Liu et al. (2020)[3] identified two biases: the cross-class bias and the intra-class bias, both of which boost the accuracy of their model. Then, they proposed rectifying the prototypes to diminish these two biases. To diminish the cross-class bias, they shifted embeddings of the query set obtained from a feature extractor using a calculated shifting term. To diminish the intra-class bias, they

used label propagation[34] that predicted pseudo-labels for the query set using the original prototypes and then chose the top predicted samples and added them to the support set to recalculate the prototypes again (rectification). They then used reactivated prototypes to calculate the final labels for the query set. Even though their approach was simple, it significantly increased the accuracy of the cosine-based prototypical network.

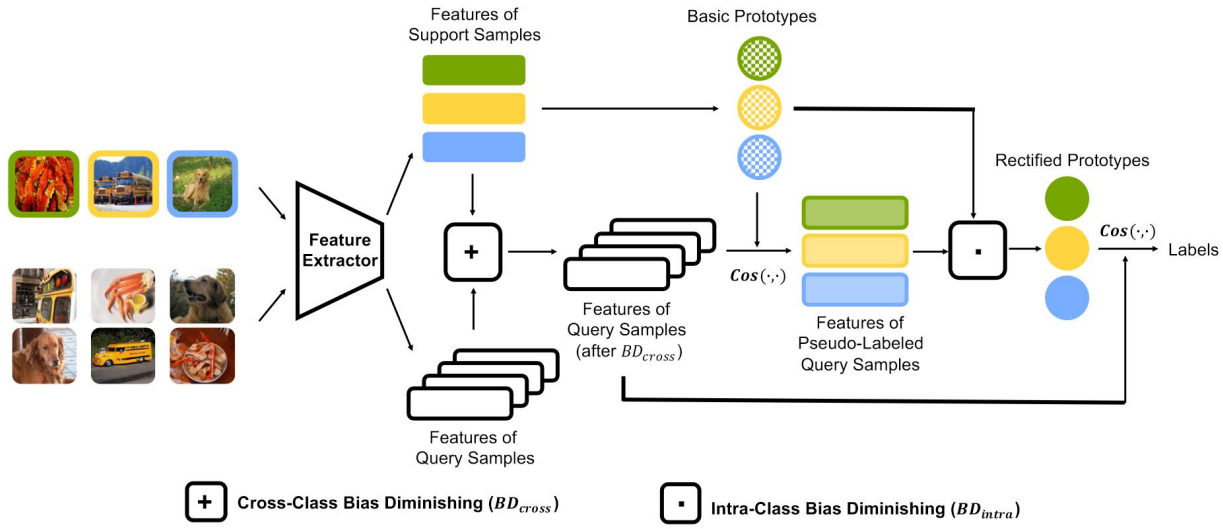

Figure 2.3: *Overview of the approach of Liu et al. (2020)[3].*

While researchers in the FSL community were busy engineering complicated models, Chen et al. (2020)[4] discovered a very simple, yet very effective, FSL model that has been overlooked by the research community. Because their model has set a new benchmark score by outperforming all existing state-of-art models at the time, the authors named it "new Meta-Baseline"[4]. The authors described their approach as a 2-step approach. First, they used the base classes to train a network with a classifier in an ordinary classification set up. Second, they used query and support sets in a meta-learning framework to fine-tune the pre-trained network. The network was then used as a feature extractor to extract embeddings that are then used with a scaled cosine similarity measure[4]. Figure 2.4 illustrates the approach that Chen et al. (2020) used[4]. The meta-baseline can be thought of as a cosine-

based prototypical network with a pre-trained feature extractor. Because of its simplicity and effectiveness, most researchers have adopted pre-training into their FSL approaches. I also adopted the pre-training stage into both of my approaches.

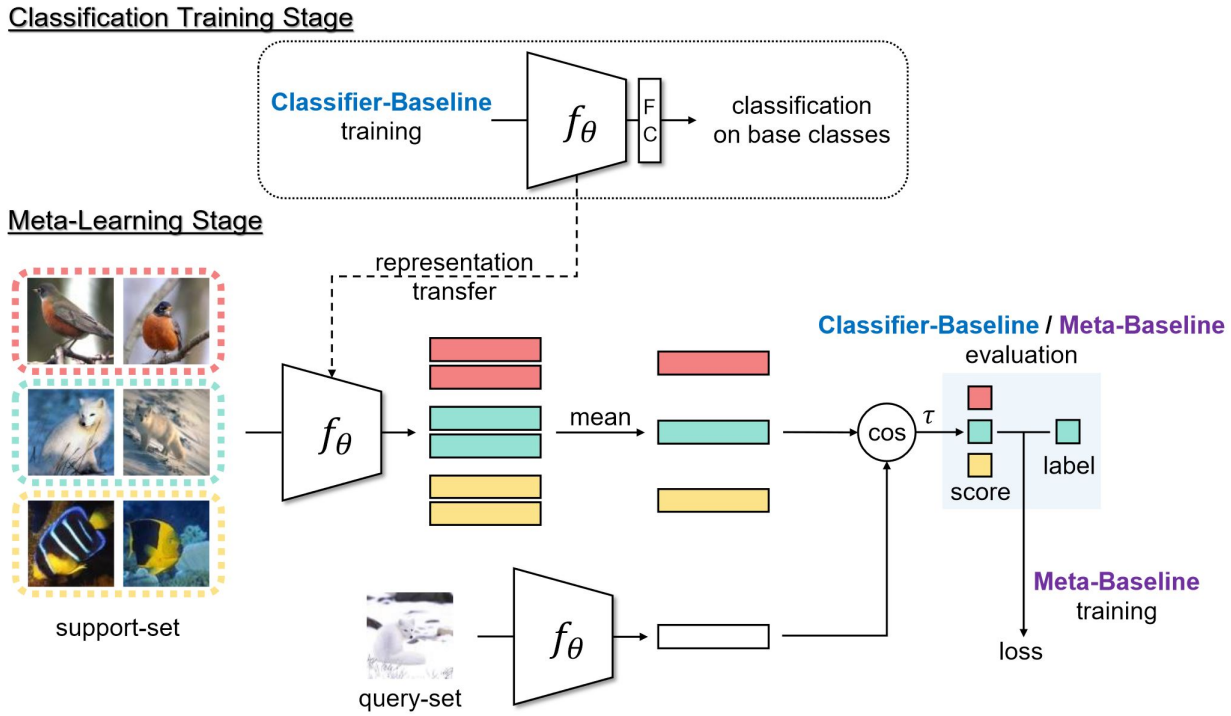

Figure 2.4: *Overview of the approach of Chen et al. (2020)*[4].

2.1.2 Optimization-based approaches

Recent models using the optimized-based approaches estimate model parameters by learning a parameterized predictor on top of a feature extractor[5;30]. Bertinetto et al. (2018)[30] reported on teaching a deep neural network to use standard machine learning tools. In their research, they used a deep neural network as a feature extractor, then applied ridge regression to the embeddings obtained from the feature extractor[30]. Their approach, at the time, achieved better performance than metric-based approaches[30].

Inspired by the work of Bertinetto et al. (2018)[30], the authors of the MetaOptNetSvm proposed using a differentiable GPU-based quadratic programming (QP) solver (proposed by Amos et al. (2017)[35]) and named OptNet to learn linear classifiers like ridge regression (RD)

and support-vector machine (SVM)[5]. In their research, dual QP equations were formulated and solved to implement RD and SVM on top of a deep neural network. Figure 2.5 shows the illustration that Lee et al. (2019) used in their research[5].

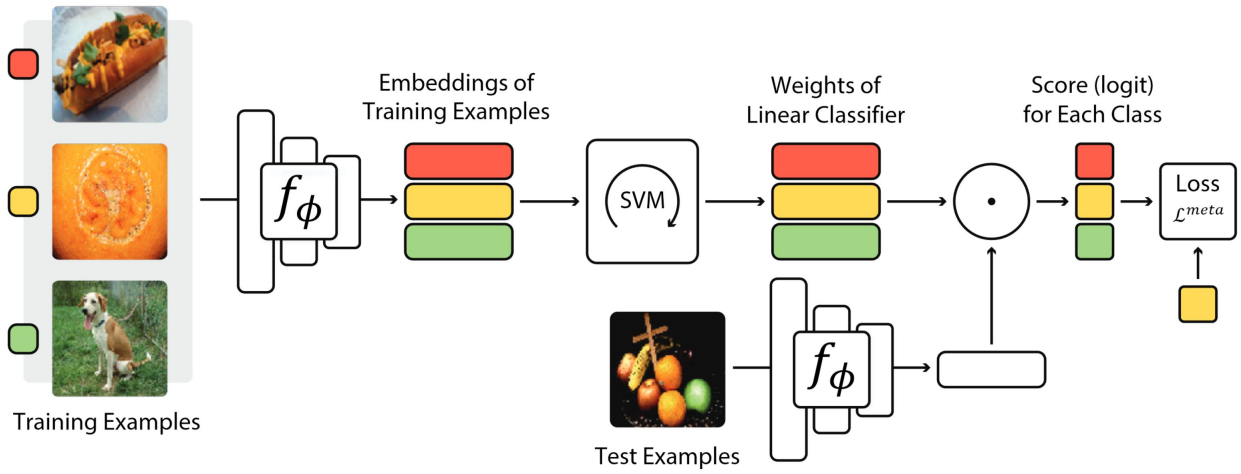

Figure 2.5: *Overview of the approach of Lee et al (2019)[5].*

2.1.3 Semantics-based approaches

My first approach fits this line of FSL research. In this approach, the performance of the model is improved by using textual semantic knowledge[6–8;31–33]. Chen et al. (2019)[6] created the Dual TriNet Network[6] where feature representations come from different layers of a ResNet-18. These representations are then treated as different levels of abstract semantic information. They then used an encoder to encode the semantic information into the semantic space. Using Semantic Gaussian and Semantic Neighborhood, they decoded the semantic space into feature augmentation[6]. Figure 2.6 illustrates this network from Chen et al. (2019)[6].

Li et al. (2020)[7] used the names of the base classes in the meta-training task as attributes. A word-embedding model was then given those attributes to extract semantic vectors for classes[7]. Then, they obtained margin penalty for each pair of classes by using an adaptive margin generator. Thus, their proposed adaptive margin loss was obtained by combining the margin penalty with the classification loss[7]. Figure 2.7 illustrates this process.

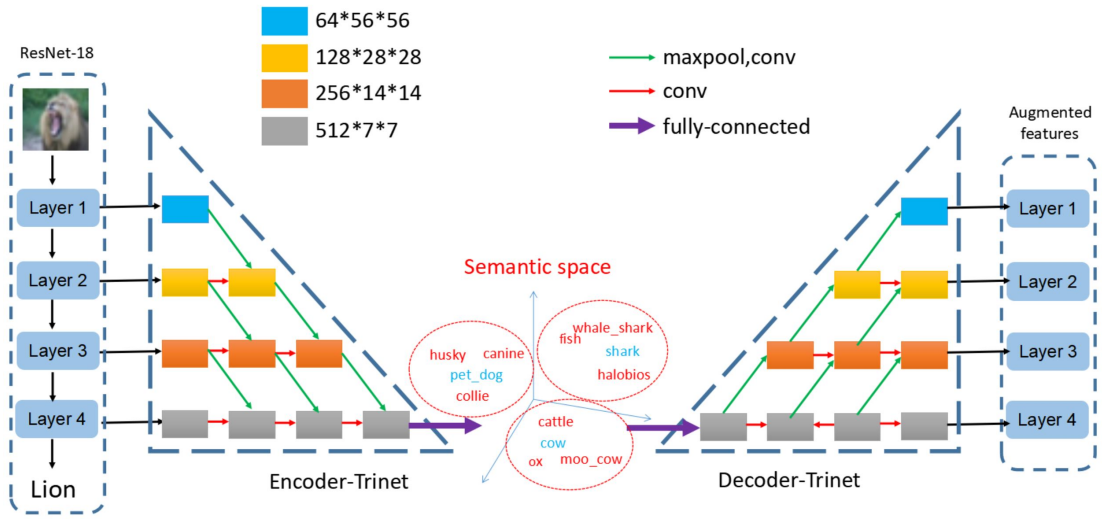

Figure 2.6: *Overview of the approach of Chen et al. (2019)*[6].

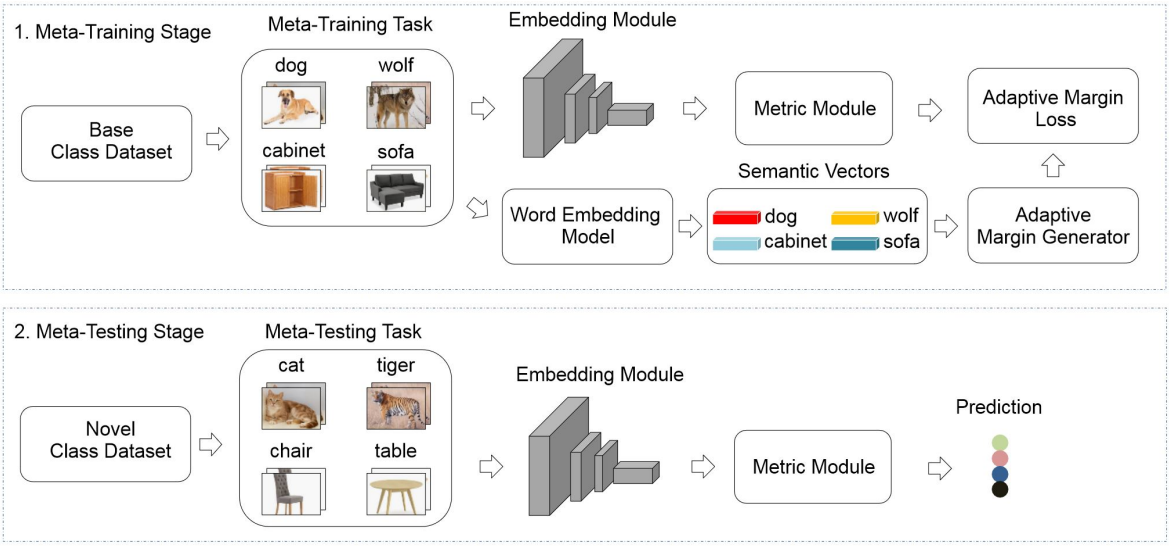

Figure 2.7: *Overview of the approach of Li et al. (2020)*[7] .

Zhang et al. (2021)[8] used attributes extracted from WordNet[36] as input to their proposed prototype-completion network, ProtoComNet. Their model works side-by-side with a feature extractor[8]. Their research can be viewed as two models in one, where one model is their ProtoComNet, using attributes to complete prototypes, and the second model is somewhat similar to the model in Chen et al. (2020)[4]. Then, and using their probability fusion strategy, they fused the outputs of the two models. They assumed that the estimated prototypes

followed the Multivariate Gaussian Distribution (MGD) and named their fusion strategy "GaussFusion"[8]. I used their fusing strategy in my approach as well. Figure 2.8 illustrates Zhang et al. (2021)[8].

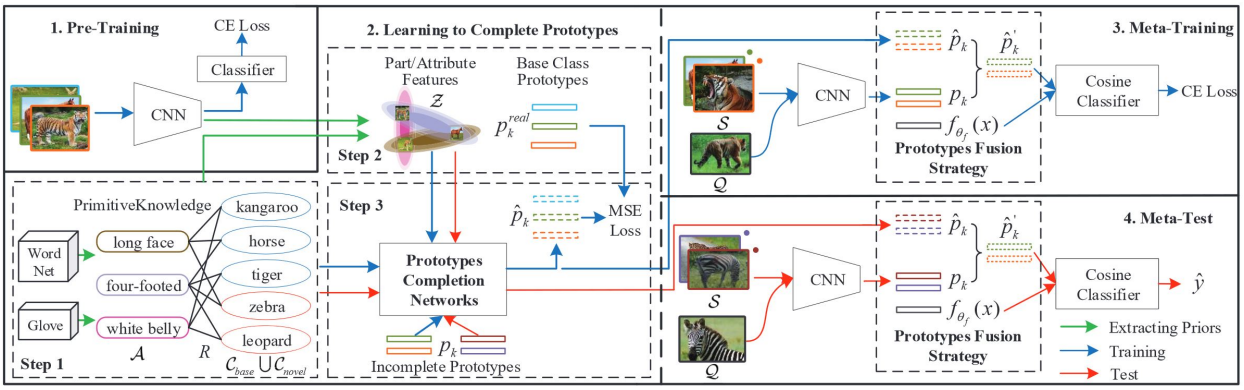

Figure 2.8: *Overview of the approach of Zhang et al. (2021)[8].*

2.2 Limitations of Current Work

I found a total of three major limitations with current FSL approaches. Two major limitations in the current semantic-based approaches, and one limitation in the work of Liu et al. (2020)[3]:

1. **Generalization**

   Most semantic-based models do not generalize to other models because they use specific embeddings or direct optimization instead of inter-class attribute transfer, which potentially improves multiple models.

2. **Complexity**

   Most semantic-based models are based on very complex architectures in order to be able to derive knowledge from attributes; hence, they require a significant amount of computational power.

3. **Optimization opportunities**

The prototypical rectification in Liu et al. (2020)[3] does not take advantage of fine-tuning, enhancement loop, or unlabeled samples, which could significantly boost its performance.

The first approach in my research generalizes easily and can apply on top of any existing FSL model, as I show in this dissertation. My first approach was very simple and straightforward, incurring negligible marginal runtime over the model. My second approach used a pre-trained feature extractor that it then fine-tuned. It also further improved the prototypes with an iterative process that takes advantage of unlabeled samples in the case of semi-supervised learning. This in turn boosts the prediction performance.

2.3 Summary

In this chapter, I introduced research on FSL as it relates to my work. FSL has both an inductive and transductive setting. In the inductive setting, no knowledge is assumed about the query set, and each sample from that set is treated individually. On the other hand, the transductive setting uses assumptions and uses the query set to contribute to higher model accuracy. FSL falls broadly into three categories: metric-based, optimization-based, and semantic-based. Metric-based approaches are models that obtain embeddings from a feature extractor over which a distance-based prediction rule is then learned; optimization-based approaches estimate model parameters by learning a parameterized predictor on top of a feature extractor; semantics-based approaches improve the performance of the model by using textual semantic knowledge. My research addresses problems with current FSLs, primarily complexity and generalization.

Three additional methods are directly related to my research. Transfer learning, probabilistic fusion, and prototype rectification are discussed in the next chapter.

Chapter 3

Related Research on Transfer Learning, Prototype Rectification, and Probabilistic Fusion

In this chapter, I discuss research in transfer learning, prototype rectification, and probabilities fusion as it relates to my research. My first approach relies on two major techniques: attribute transfer as reported in Lampert et al. (2009)[9] and probabilities fusion as reported in Zhang et al. (2021)[8] I explain the first in Section 3.1 and the second in Section 3.3. My second research approach was inspired by prototype rectification as reported in Liu et al. (2020)[3], which I explain in Section 3.2.

3.1 Attribute Transfer

Lampert et al. (2009)[9] formulated the problem of zero-shot learning as follows: Let training samples be $(x_1, l_1), ..., (x_n, l_n) \subset X \times Y$ where X is an arbitrary feature space and $Y = y_1, ..., y_K$, which consists of K discrete classes. The task of zero-shot learning is to learn a classifier $f : X \to Z$ for a label set $Z = z_1, ..., z_L$ that is disjoint from Y.

Lampert et al. (2009)[9] explained the zero-shot learning problem and why it is difficult

for machine learning systems. Lampert et al. (2009)[9] also explained how humans can identify unseen classes when high-level descriptions of these classes are available in a zero-shot classification task[9]. For example, if we provide a human a high-level description of a STOP sign such as: "red", "white letters", "eight-sided", and "traffic sign", then a human can easily identify stop signs in any language; another example from Lampert et al. (2009)[9] is the ability of a human to identify an elephant when provided with the following high-level description: "grey", "large animal", and "long trunk"[9].

Following this paradigm, the authors developed two approaches that use high-level attributes to classify unseen classes in zero-shot learning[9]. The first approach was the Direct Attribute Prediction (DAP) where a classifier trained to directly predict attributes was used in a zero-shot learning task to infer classes. Figure 3.1 from Lampert et al. (2009)[9] illustrates DAP.

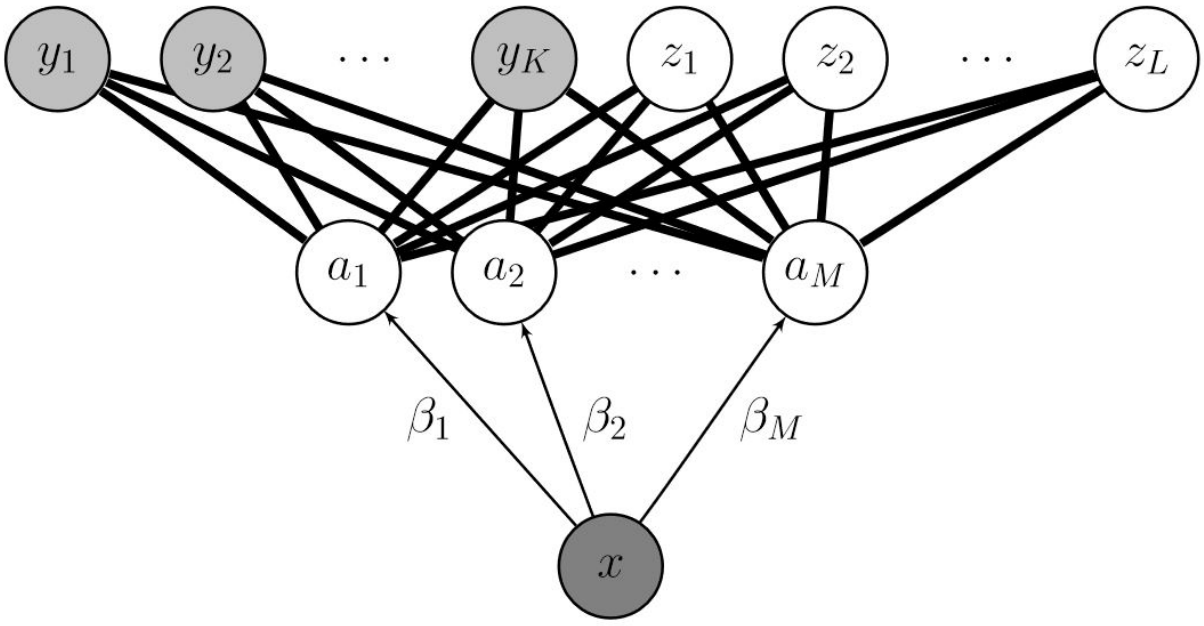

Figure 3.1: *Overview of the DAP approach in Lampert et al. (2009)[9].*

The second approach was the Indirect Attribute Prediction (IAP) where a classifier was trained to predict classes that are then used to induce or infer attributes[9]. I incorporated

their second approach into my own approach. Figure 3.2 from Lampert et al. (2009)[9] illustrates IAP.

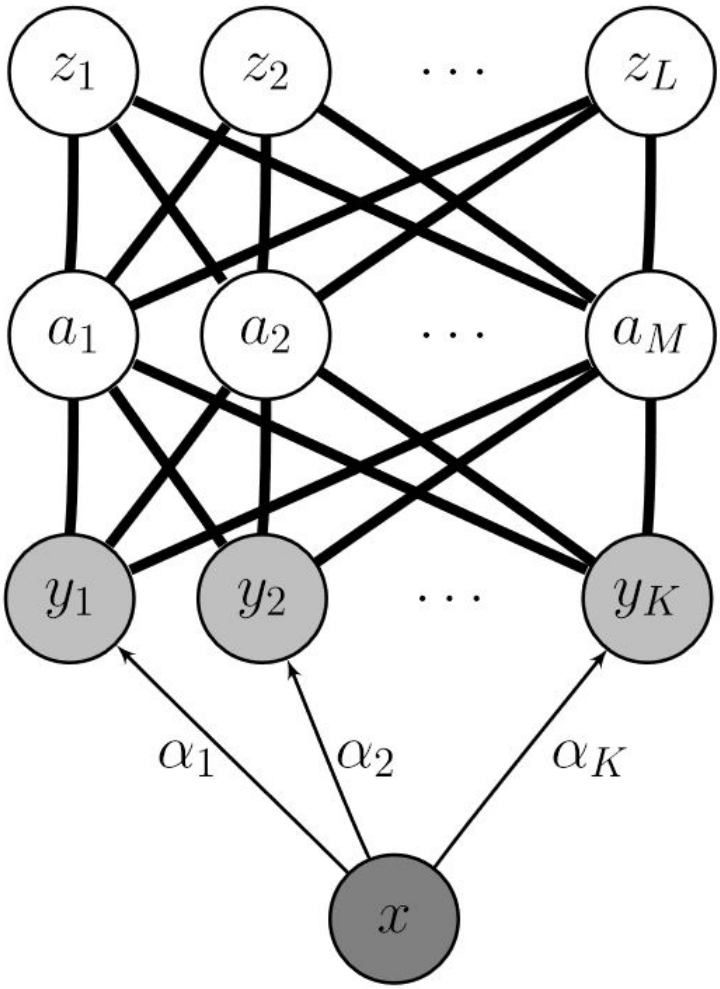

Figure 3.2: *Overview of the IAP approach in Lampert et al. (2009)[9].*

In a zero-shot learning set up, using IAP, known classes are first predicted and used to infer attributes, then the class labels of the unseen classes are inferred from the inferred attributes. In zero-shot learning, an IAP is trained using a multi-class classifier, either a regression or linear classifier, to predict known classes[9]. When using this classifier to predict

unseen classes, the first step is to infer attributes by using the following equation:

$$p(a_m|x) = \sum_{k=1}^{K} = p(a_m|y_k)p(y_k|x) \tag{3.1}$$

where $p(y_k|x)$ represents the estimates of the probabilistic multi-class classifier for all training classes $y_1, ..., y_K$. The authors assume a deterministic dependence between classes and attributes when setting $p(a_m|y)$. They also assume that all attributes have binary values so for any training class y, the attribute representation $a_y = (a_1^y, ..., a_m^y)$ are fixed-length binary vectors: $p(a_m|y) = [a_m = a_m^y]$. Then those attributes are transferred into labels using the following equation:

$$f(x) = \underset{l=1,...,L}{argmax} \prod_{m=1}^{M} = \frac{p(a_m^{z_l}|x)}{p(a_m^{z_l})} \tag{3.2}$$

where they assume a factorial distribution $p(a) = \prod_{m=1}^{M} p(a_m)$ for the factor $p(a_m)$, using the empirical means $p(a_m) = \frac{1}{k}\sum_{k=1}^{K} a_m^{yk}$ as attribute priors over the training classes. I used an approach similar to Lampert et al.'s IAP approach; this approach is explained in subsection 4.1.2.

3.2 Prototype Rectification

Liu et al. (2020)[3] proposed a cosine similarity based prototypical network, where the feature extractor is trained with a cosine classifier attached to it, using the base classes in a normal classification set up. Liu et al. then used the feature extractor (without fine-tuning) to predict the query set. They claimed that fine-tuning ("retraining" is the word they used) the feature extractor will likely cause overfitting[3].

As discussed in Section 2.1.1, Liu et al. (2020)[3] proposed prototype rectification through diminishing two biases: the cross-class bias and the intra-class bias. To diminish the cross-class bias, Liu et al. first calculated a shifting term. Then, they obtained the original embeddings of the query set from a feature extractor. Using the shifting term, they shifted the original embeddings of the query set. To diminish the intra-class bias, they used label

propagation[34] by predicting pseudo-labels for the query set using the original prototypes, then choosing the top predicted samples and adding them to the support set to recalculate the prototypes (rectification). They used the reactivated prototypes to calculate the final labels for the query set.

Liu et al. (2020)[3] first formalized the cross-class bias as follows:

$$B_{cross} = E_{X_s \sim PS}[X_s] - E_{X_q \sim PQ}[X_q] \tag{3.3}$$

where PS represents the distribution of the support set and PQ represents the distribution of the query set. They first obtained a shifting term, ξ , using the following[3]:

$$\xi = \frac{1}{|S|} \sum_{i=1}^{S} \overline{X}_{i,s} - \frac{1}{|Q|} \sum_{i=1}^{Q} \overline{X}_{i,q} \tag{3.4}$$

where $\overline{X}$ is the normalized embeddings obtained from a feature extractor. This shifting term was then used to shift the original embeddings of the query set to diminish the cross-class bias. The newly shifted embeddings of the query set were then used in the rest of the process.

Liu et al. (2020)[3] formalized the intra-class bias as follows:

$$B_{intra} = E_{X' \sim P'_X}[X'] - E_{X \sim P_X}[X] \tag{3.5}$$

where P'_X is the distribution of all data belonging to a certain class and P_X is the distribution of the available labeled data of this class. Liu et al. (2020)[3] adapted the pseudo-labeling strategy to reduce the intra-class bias. They predicted labels for the query set, then used their prediction confidence[37] to pick the top Z query samples that were confidently predicted.

To predict the query samples, Liu et al. (2020)[3] calculated original prototypes using the normalized embeddings of the support set as follows[3]:

$$P_n = \frac{1}{|K|} \sum_{i=1}^{K} \overline{X}_{i,n} \tag{3.6}$$

where n is the class and K is the number of shots. Liu et al. (2020)[3] then used the top Z

query samples, augmenting them into the support set S to create $S' = S \cup Q$. They used the newly augmented support set, S' to calculate new prototypes using the following equation[3]:

$$P'_n = \sum_{i=1}^{Z+K} w_{i,n}.\overline{X'}_{i,n} \tag{3.7}$$

where X' refers to the embeddings of the support samples and the embeddings of the top Z query samples together. Liu et al. (2020)[3] calculated $w_{i,n}$ using the following equation[3]:

$$P'_n = \frac{exp(\varepsilon.Cos(X'_{i,n}, P_n)}{\sum_{j=1}^{K+Z} exp(\varepsilon.Cos(X'_{j,n}, P_n)} \tag{3.8}$$

where ε is a scale factor. Liu et al. (2020)[3] used the newly calculated prototypes, P'_n to calculate final predictions for the query set (using the shifted embeddings). Their approach inspired my second approach to this research, which also predicts labels for the query set.

3.3 Probabilistic Fusion

As I discussed in Section 2.1.3, Zhang et al. (2021)[8] have two models fused through a probabilistic fusing strategy called "GaussFusion"[8]. They proposed this probabilistic fusing strategy because they observed the estimates (p_k) of one model were better with more shots and worse with fewer shots, but the estimates ($\hat{p}_k$) of their proposed ProtoComNet were better with fewer shots and worse with more shots. Hence, they developed the fusion strategy, so the two estimates from the two different models can be combined to complement each other[8]. I chose to incorporate this strategy into my approach.

Zhang et al. (2021)[8] applied the Bayesian estimation when fusing prototypes[8]. They assumed that a Multivariate Gaussian Distribution (MGD) had sampled both prototypes p_k and $\hat{p}_k$ with $N(\mu_k, diag(\sigma_k^2))$ and $N(\hat{\mu}_k, diag(\hat{\sigma}_k^2))$, where the means are μ_k and $\hat{\mu}_k$, and the diagonal covariances are $diag(\sigma_k^2)$ and $diag(\hat{\sigma}_k^2)$. The goal was to calculate $N(\hat{\mu'}_k, diag(\hat{\sigma'}^2_k))$, which is a fused representation with the MGD mean, $\hat{\mu'}_k$, and the diagonal covariance, $diag(\hat{\sigma'}^2_k)$[8].

Using the cosine similarity between the prototype p_k and sample x, Zhang et al. (2021)[8]

calculated $P(y = k|x)$, which is the probability of a sample x belonging to a K class. They calculated $\hat{P}(y = k|x)$ using $\hat{p}_k$ similarly[8]. After calculating the probabilities, they then calculated the mean, μ_k, using the following equation[8]:

$$\mu_k = \frac{1}{\sum\limits_{x \in S \cup Q} P(k|x)} \sum_{x \in S \cup Q} P(k|x) f_{\theta_f}(x) \tag{3.9}$$

where $f_{\theta_f}(x)$ is the extracted embedding, Q is the query set, and S is the support set. Zhang et al. (2021)[8] calculated $\hat{\mu}_k$ using $\hat{P}(y = k|x)$. They then calculated σ_k using the following equation:

$$\sigma_k = \sqrt{\frac{1}{\sum\limits_{x \in S \cup Q} P(k|x)} \sum_{x \in S \cup Q} P(k|x)(f_{\theta_f}(x) - \mu_k)^2} \tag{3.10}$$

Zhang et al. (2021)[8] then calculated $\hat{\sigma}_k$ using $\hat{\mu}_k$. After calculating μ_k, $\hat{\mu}_k$, σ_k, and $\hat{\sigma}_k$, they then calculated $\hat{\mu'}_k = \frac{\sigma_k \odot \hat{\mu}_k + \hat{\sigma}_k \odot \mu_k}{\hat{\sigma}_k \odot \sigma_k}$ and $diag(\hat{\sigma'}^2_k) = diag(\frac{\sigma_k^2 \odot \hat{\sigma}_k^2}{\hat{\sigma}_k^2 \odot \sigma_k^2})$. This provides a new set of prototypes (or estimates) $\hat{p'}_k$ that are then used to calculate the new probabilities of the query samples: $\hat{P'}(y = k|x)$. These new probabilities represent the fused probabilities[8].

3.4 Summary

This chapter covered attribute transfer, prototype rectification, and probabilistic fusion, three methods that apply directly to my research. High-level descriptions such as attributes can be very useful in improving classifier knowledge for classifying objects. I also discussed two approaches to attribute transfer: a direct approach and indirect approach. In the direct approach, attributes are directly predicted and then used to infer labels. In the indirect approach, attributes are inferred from labels that are then used to infer new labels. I adopted the indirect attribute transfer into my first approach in this research. Prototype rectification, to achieve better results, reduces two biases: cross-class bias through embedding shifting and inter-class bias through pseudo-labels. I adopted pseudo-labels in my second approach in this research. Probabilistic fusion can be very useful in fusing two outputs to improve a

model. This type of fusion is based on a MGD. I adopted this fusion strategy into my first approach.

The next chapter, Methodology, provides more information about attribute transfer, prototype rectification, and probabilistic fusion, and how I adopted them into my two approaches.

Chapter 4

Methodology

In this chapter, I explain two contributions in my research. The first contribution, between-class attribute transfer (BCAT), which successfully transfers attributes is detailed in Section 4.1. The second contribution, prototype enhancement (PE), where prototypes are enhanced, is detailed in Section 4.2.

4.1 New Attribute Transfer Framework: BCAT

In this section, I explain my between-class attribute transfer framework in detail. Figure 4.1 illustrates the framework. Subsection 4.1.1 explains the pre-training step I used before fine-tuning my model. Subsection 4.1.2 explains in detail my own version of the Indirect Attribute Prediction (IAP) that was proposed by Lampert et al. (2009)[9]. Specifically, the section focuses on how I incorporate IAP into my framework and adapt it to a deep learning setting. Subsection 4.1.3 explains how I use the fusion strategy. I then conclude with the scaling step in subsection 4.1.4.

4.1.1 Step 1: Pre-training

In section 2.1.1, I reviewed Chen et al. (2020)[4] and their simple yet effective few-shot learning (FSL) model. They first pre-trained a feature extractor using the base classes in

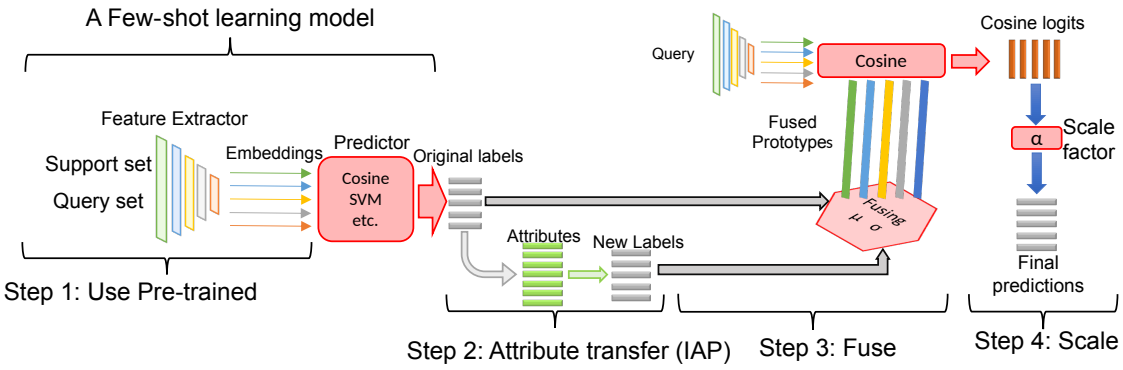

Figure 4.1: *Overview of my BCAT framework. First, I used base class labels to pre-train the feature extractor in a normal classification setting before I use it to fine-tune the model in a meta-learning setting. Second, I used the original probabilistic estimates to infer new attributes for the query set, which I then used to infer new probabilistic estimates for the query set. Third, I fused the original probabilistic estimates with the inferred ones to construct prototypes that I used in calculating cosine similarity for the query set. Finally, I used a learnable scaling factor to scale the similarities.*

a normal classification set up. They then adapted the the pre-trained feature extractor into their model, which they fine-tuned using a meta-learning set up[4]. In my framework, I adapted pre-training as a first step. I found that pre-training the feature extractor is an important step to boost the results in my BCAT framework. As I show in Section 5.1, incorporating pre-training in a model doesn't always boost the performance of that model, but incorporating pre-training into my framework as a first step does help boost the results. When I pre-trained, I used a Resnet-12 with a linear classifier. I used the base classes to train both Resnet-12 and the linear classifier in a normal classification set up. I then removed the linear classifier and used the ResNet-12 as a feature extractor and followed the rest of the steps which I explain in the following subsections.

4.1.2 Step 2: Indirect Attribute Transfer

Lampert et al. (2009)[9] proposed the IAP approach introduced in Section 3.1. Here, I will discuss my own, very similar approach for few-shot learning. I also provide my own implementation using the tensorized notation typically used in deep learning.

When I finished pre-training the feature extractor on the base classes in the previous step, I then began fine-tuning the model. First, attributes were inferred using the predicted estimates, $p(y_k|x)$, from the probabilistic multi-class classifier. The following equation was then used to infer attributes:

$$p(a_m|x) = \sum_{k=1}^{K} \frac{e(a_m|y_k)}{\frac{1}{k}\sum_{m=1}^{m} e(a_m|y_k)} p(y_k|x) \tag{4.1}$$

where $e(a_m|y_k)$ is the static encoding of attributes for every label. I discovered that converting the static encoding into a probability distribution gave better results. Therefore, I scaled the static encoding by dividing by $\frac{1}{k}\sum_{m=1}^{m} e(a_m|y_k)$. Using the scaled encoding of labels for every attribute, $p(y_k|a_m)$, the attribute priors $p(a_m)$ could then be calculated with the following equation:

$$p(a_m) = \frac{p(y_k|a_m)}{\sum_{k=1}^{k} p(y_k|a_m)}. \tag{4.2}$$

Inferring new probabilistic multi-class estimates from the inferred attributes was done using the following equation:

$$p(\hat{y}_k|x) = \sum_{m=1}^{m} p(a_m|x)p(a_m). \tag{4.3}$$

In the next step, I combined this estimator with the original estimator. I could then propose a very similar approach using my own implementation method using the tensorized notation typically used in deep learning:

Let $support^{E\times S\times F}$ and $query^{E\times Q\times F}$ be two tensors representing the features extracted (embeddings) for the support and query sets, where

E = *number of episodes (batch size)*,

$S = total\ number\ of\ samples\ in\ the\ support\ set$,

$Q = total\ number\ of\ samples\ in\ the\ query\ set$,

and $F = dimension\ size\ of\ the\ extracted\ features$.

Let $Y^{E\times S\times K}$ and $A^{E\times S\times M}$ be two tensors representing two hot-encoding of labels and attributes for the support set, where

$K = number\ of\ classes\ in\ an\ episode$,

and $M = number\ of\ attributes\ in\ an\ episode$.

Let $logits^{E\times Q\times K}$ be the output of the model to which I am applying this framework when given $support^{E\times S\times F}$ and $query^{E\times Q\times F}$ as inputs.

The hot-encoding of attributes can then be obtained for every label, $H^{E\times K\times M}$, by the following batch matrix multiplication:

$$H = Y^T A. \tag{4.4}$$

Scaling $H^{E\times K\times M}$ using the following equation gave better results because the scaling spans $H'^{E\times K\times M}$ over a probability distribution instated of a static encoding:

$$H' = \frac{H}{\frac{1}{K} * \underset{d=2}{Sum}(H)} \tag{4.5}$$

where $d = 2$ means applying the sum functions across the second dimension (M). Obtaining the predictions $P^{E\times Q\times K}$ of $query^{E\times Q\times F}$ can be done easily by passing the $logits^{E\times Q\times K}$ of the model to a soft-max according to the following equation:

$$P = \underset{d=2}{Softmax}(logits). \tag{4.6}$$

Having calculated $P^{E\times Q\times K}$ and $H'^{E\times K\times M}$, we can then infer a probabilistic distribution for attributes, $I^{E\times Q\times M}$ for the $query^{E\times Q\times F}$ set by the following batch matrix multiplication:

$$I = PH'. \tag{4.7}$$

The hot-encoding of labels for every attribute, $L^{E\times M\times K}$, can then be obtained using the

following batch matrix multiplication:

$$L = A^T Y. \tag{4.8}$$

Scaling $L^{E\times M\times K}$ using the following equation can also provide better results because it spans $L'^{E\times M\times K}$ over a probability distribution instated of a static encoding:

$$L' = \frac{L}{\frac{1}{K} * \underset{d=2}{Sum}(L)}. \tag{4.9}$$

$L'^{E\times M\times K}$ can then be used to obtain $L''^{E\times M\times K}$, which is the average number of labels for every attribute, using the following equation:

$$L'' = \frac{L'}{\underset{d=2}{Sum}(L')}. \tag{4.10}$$

Inferring a new probabilistic distribution of labels, $\hat{P}^{E\times Q\times K}$ from the inferred probabilistic distribution of attributes, $I^{E\times Q\times M}$ is straightforward using $L''^{E\times M\times K}$ in the following batch matrix multiplication:

$$\hat{P} = IL''. \tag{4.11}$$

$P^{E\times Q\times K}$ and $P'^{E\times Q\times K}$ can then be fused in Step 3, which is explained in Section 4.1.3.

4.1.3 Step 3: Fusing

Section 3.2 introduced probabilistic fusion from Zhang et al. (2009)[8]. In this step, I used the very same approach they implemented. Using the fusion strategy of Zhang et al. (2009)[8], the original probabilistic estimates $p(y_k|x)$ of the multi-class classifier were fused with the new probabilistic multi-class estimates $p(\hat{y}_k|x)$ inferred in the previous step. First, the multivariate Gaussian distribution means μ_k and $\hat{\mu}_k$ for $p(y_k|x)$ and $p(\hat{y}_k|x)$ were calculated using Equation 3.9. Next, the covariances σ_k^2 and $\hat{\sigma}_k^2$ were calculated using Equation 3.10. Finally, new prototypes were calculated; then, using a cosine function, the similarities logits of the

query set were calculated.

I also provide a listing of the implementation of Zhang et al.'s (2009)[8] fusion technique using the tensorized notation in the previous section.

Let $support^{E\times S\times F}$ and $query^{E\times Q\times F}$ be two tensors representing the features extracted for the support and query sets, where:

$E =$ *number of episodes (batch size)*,

$S =$ *total number of samples in the support set*,

$Q =$ *total number of samples in the query set*,

and $F =$ *dimension size of the extracted features*.

Let $Y^{E\times S\times K}$ be a tensor representing one hot encoding of labels for the support set, where

$K =$ *number of classes in an episode*,

and let $P^{E\times Q\times K}$ be the probabilistic output obtained from Equation 4.6 of the model to which my framework is being applied, and $\hat{P}^{E\times Q\times K}$ be the inferred probabilities obtained from Equation 4.11. I first calculated a concatenation $C_1^{E\times (S+Q)\times K}$ between $Y^{E\times S\times K}$ and $P^{E\times Q\times K}$ using the following equation:

$$C_1 = Cat(Y, P) \tag{4.12}$$

where Cat is the concatenation function that major deep-learning frameworks provide. A concatenation $C_2^{E\times (S+Q)\times K}$ was then calculated between $support^{E\times S\times F}$ and $query^{E\times Q\times F}$ using the following equation:

$$C_2 = Cat(support, query). \tag{4.13}$$

Using the calculated concatenations $C_1^{E\times (S+Q)\times K}$ and $C_2^{E\times (S+Q)\times K}$, I can calculate the mean, $M^{E\times K\times K}$, using the following batch matrix multiplication:

$$M = C_1^T C_2. \tag{4.14}$$

The difference, $D^{E\times K\times(S+Q)\times K}$, was then calculated by expanding the size of $C_2^{E\times(S+Q)\times K}$ to become $C_2^{E\times K\times(S+Q)\times K}$ and $M^{E\times K\times K}$ to become $M^{E\times K\times(S+Q)\times K}$, using them in the following equation:

$$D = (C_2 - M)^2. \tag{4.15}$$

The size of $C_1^{TE\times K\times(S+Q)}$ was then expanded to become $C_1^{TE\times K\times(S+Q)\times K}$. and using an element-wise multiplication. it was multiplied with $D^{E\times K\times(S+Q)\times K}$. I then took the summation across the third dimension with size $S+Q$ to obtain $G_1^{E\times K\times K}$:

$$G_1 = \underset{d=3}{Sum}(C_1^T * M). \tag{4.16}$$

Using the expanded $C_1^{TE\times K\times(S+Q)\times K}$ and $G_1^{E\times K\times K}$, J could be determined with the following equation:

$$J = \frac{G_1}{\underset{d=3}{Sum}(C_1^T)}. \tag{4.17}$$

Using $\hat{P}^{E\times Q\times K}$, I repeated the process to obtain $\hat{M}$ and $\hat{J}$. Finally, prototypes were produced that were the result of fusing using the following equation:

$$prototypes = \frac{M * \hat{J} + \hat{M} * J}{\hat{J} + J}. \tag{4.18}$$

I expanded the size of $prototypes^{E\times K\times K}$ and $query^{E\times Q\times F}$ to become $prototypes^{E\times Q\times K\times K}$ and $query^{E\times Q\times K\times F}$ and calculated the cosine similarity between them across the fourth dimension to find the final $logits^{E\times Q\times K}$:

$$logits = \underset{d=4}{Cos}(query, prototypes). \tag{4.19}$$

The final $logits^{E\times Q\times K}$ then was used in the final step of my framework, as described in the next section.

4.1.4 Step 4: Scaling

As mentioned in Section 2.1.1, Oreshkin et al. (2018)[11] proposed metric scaling to close the gap between the Euclidean distance and cosine similarity[11]. Because I finished the fusing step using a cosine function, I discovered that adding metric scaling to scale the similarities logits of the query set greatly improved the accuracy of my BCAT framework.

4.2 New Transductive Technique: PE

In this section, I explain my Supervised and semi-supervised transductive technique (PE) in detail which is illustrated in Figure 4.2. Subsection 4.2.1 proposes the pre-training step that I use before fine-tuning the model. Subsection 4.2.2 explains how to fine-tune the model using the enhancement loop.

4.2.1 Step 1: Pre-training

In Section 2.1.1, I noted Chen et al. (2020)[4] used a normal classification set up along with the base classes to pre-train the feature extractor. They then inserted the pre-trained feature extractor into their model, which they fine-tuned using a meta-learning set up[4]. In my PE technique, I used pre-training as a first step just as in my BCAT framework with the same settings used for pre-training the feature extractor. In fact, the same pre-trained feature extractor used in my BCAT framework was used in the PE technique. After the pre-training step, I fine-tuned the model using the enhancement loop, which I explain in Section 4.2.2.

4.2.2 Step 2: Fine-tuning with enhancement loop

In a meta-learning set up, fine-tuning the model began. First, without normalizing the embeddings, the support set and its labels were used to calculate the original prototypes. Most researchers normalize the embeddings of the prototypes, but I discovered that normalizing damages the enhancement loop, so I chose not to normalize. As shown in the results chapter, normalizing the prototypes to enhance once or twice provides the same accuracy as

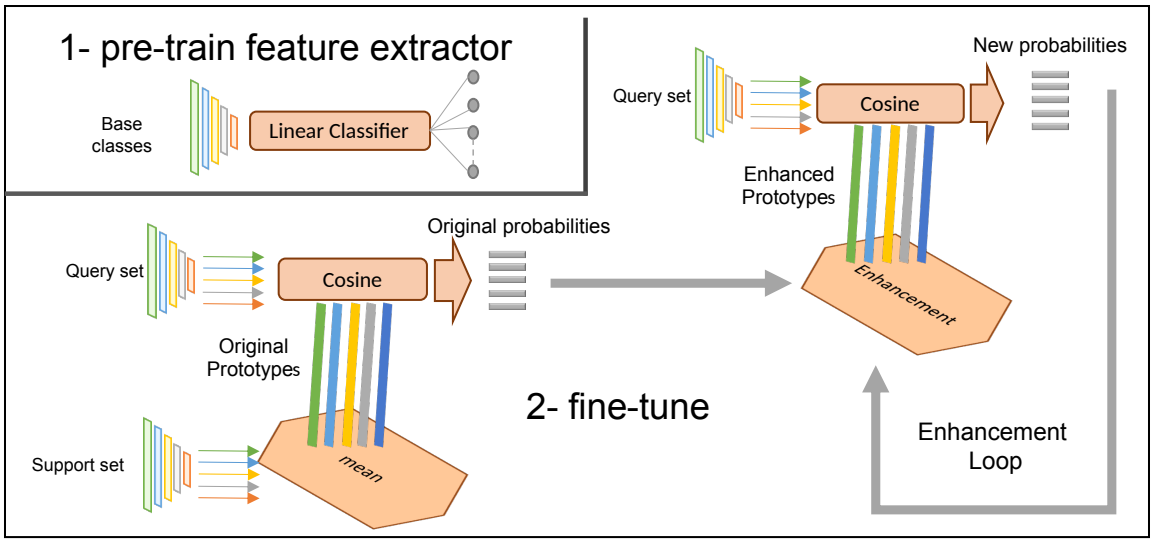

Figure 4.2: *Overview of my PE technique. First, base class labels were used to pre-train the feature extractor in a normal classification setting before fine-tuning the model in a meta-learning setting. Second, the model was fine-tuned using the embeddings of the support set to generate original prototypes that were used to obtain the query set's initial predictions using cosine similarity; then, using the initial predictions, enhanced prototypes were generated. Finally, new labels were generated for the query set using the enhanced prototypes; the newly generated labels enhanced the prototypes yet again.*

not normalizing at all, so normalization has no benefit at all. The original prototypes was calculated according to the following equation:

$$P_k = \sum_{x \in S} y.f_\phi(x). \tag{4.20}$$

Second, the probabilities of the query set were calculated using the original prototypes. I also calculated probabilities of the unlabeled set when the task was a semi-supervised learning task. The probabilities were calculated using the cosine similarity between the prototypes and each sample with the following equation:

$$p_\phi(y = k|x) = \frac{exp(\alpha.cos(f\phi(x), P_k))}{\sum_{k'} exp(\alpha.cos(f\phi(x), P_{k'}))}. \tag{4.21}$$

Finally, the enhancement loop began at this point to enhance the original prototypes. The original embeddings of the support set and the query set were used. I also used the original embeddings of the unlabeled set for any semi-supervised learning task. Again, the

embeddings I used were the original embeddings, which were not normalized according to the following equation:

$$P'_k = \sum_{x \in S \cup Q \cup U} p_\phi(y = k|x) f_\phi(x). \tag{4.22}$$

Using the enhanced prototypes P'_k, new probabilities $p'_\phi(y = k|x)$ were calculated for the query set. New probabilities were also calculated for the unlabeled set if the task involved semi-supervised learning. In practice, I set a loop that repeatedly calculated new enhanced prototypes/probabilities. This loop lasts for 2 to 6 iterations depending on the task.

A list of my own implementation using the tensorized notation as used in the previous section follows:

Let $support^{E \times S \times F}$, $query^{E \times Q \times F}$, and $unlabeled^{E \times U \times F}$ be three tensors representing the features extracted for the support, query, and unlabeled sets, where

$E =$ *number of episodes (batch size)*,

$S =$ *total number of samples in the support set*,

$Q =$ *total number of samples in the query set*,

$U =$ *total number of samples in the unlabeled set*,

and $F =$ *dimension size of the extracted features*.

Let $Y^{E \times S \times K}$ be a tensor representing one-hot encoding of labels for the support set, where

$K =$ *number of classes in an episode*,

The original prototypes, $prototypes^{E \times K \times F}$, can then be calculated using the batch matrix multiplication of the transpose of the labels, $Y^{TE \times K \times S}$ and the embeddings of the support set, $support^{E \times S \times F}$, using the following equation:

$$prototypes = Y support. \tag{4.23}$$

I expanded the size of $prototypes^{E \times K \times K}$ and $query^{E \times Q \times F}$ to be $prototypes^{E \times Q \times K \times K}$ and $query^{E \times Q \times K \times F}$ and calculated the cosine similarity between them across the fourth dimen-

sion to find the logits of the query set, $logits_{query}^{E\times Q\times K}$, using the following equation:

$$logits_{query} = \underset{d=4}{Cos}(query, prototypes). \tag{4.24}$$

After obtaining the probabilistic output of the query set, $P_{query}^{E\times Q\times K}$, a *softmax* function was applied to the third dimension of the scaled logits of the query set: $logits_{query}^{E\times Q\times K}$

$$P_{query} = \underset{d=3}{Softmax}(\alpha.logits) \tag{4.25}$$

where α is a scaling factor set to 20.

The sizes of $prototypes^{E\times K\times K}$ and $unlabeled^{E\times U\times F}$ were expanded to become $prototypes^{E\times U\times K\times K}$ and $query^{E\times U\times K\times F}$ and the cosine similarity between them was calculated across the fourth dimension to find the logits of the unlabeled set, $logits_{unlabled}^{E\times U\times K}$, using the following equation:

$$logits_{unlabled} = \underset{d=4}{Cos}(unlabeled, prototypes). \tag{4.26}$$

The probabilistic output of the unlabeled set, $P_{unlabeled}^{E\times U\times K}$ was then obtained by applying a *softmax* function to the third dimension of the scaled logits of the unlabeled set, $logits^{E\times U\times K}$, as in the following equation:

$$P_{unlabeled} = \underset{d=3}{Softmax}(\alpha.logits) \tag{4.27}$$

Where α is a scaling factor set to 50. A concatenation, $C_1^{E\times(S+Q+U)\times K}$, was then calculated between $Y^{E\times S\times K}$, $P_{query}^{E\times Q\times K}$ and $P_{unlabeled}^{E\times U\times K}$ using the following equation:

$$C_1 = Cat(Y, P_{query}, P_{unlabeled}) \tag{4.28}$$

where Cat is a concatenation function provided by major deep-learning frameworks. A concatenation, $C_2^{E\times(S+Q+U)\times K}$, was then calculated among $support^{E\times S\times F}$, $query^{E\times Q\times F}$, and

$unlabeled^{E\times U\times F}$ using the following equation:

$$C_2 = Cat(support, query, unlabeled). \tag{4.29}$$

Using the calculated concatenations, $C_1^{E\times(S+Q+U)\times K}$ and $C_2^{E\times(S+Q+U)\times F}$, the enhanced prototypes, $prototypes_{enhanced}^{E\times K\times K}$ can be calculated using the following batch matrix multiplication equation:

$$prototypes_{enhanced} = C_1^T C_2. \tag{4.30}$$

The prototypes can be enhanced a second time using the enhanced prototypes, $prototypes_{enhanced}^{E\times K\times K}$, to calculate new logits for the query, $logits_{query}^{'E\times Q\times K}$, using Equation 4.24, and for the unlabeled set, $logits_{unlabled}^{'E\times U\times K}$, using Equation 4.26. Their new probabilistic output, $P_{query}^{'E\times Q\times K}$ and $P_{unlabeled}^{'E\times U\times K}$, can then be calculated using equations 4.25 and 4.26. Finally, we can proceed with the process and calculate equations 4.28, 4.29, and 4.30 to obtain newly enhanced prototypes, which can be enhanced again or used as final prototypes in the next step.

Finally, the sizes of the final enhanced prototypes $prototypes_{enhanced}^{E\times K\times K}$ and $query^{E\times Q\times F}$ can be expanded to become $prototypes_{enhanced}^{E\times Q\times K\times K}$ and $query^{E\times Q\times K\times F}$ and the cosine similarity calculated between them across the fourth dimension to find the final $logits^{E\times Q\times K}$ using the following equation:

$$logits_{final} = \underset{d=4}{Cos}(query, prototypes_{enhanced}). \tag{4.31}$$

The final $logits_{final}^{E\times Q\times K}$ were then scaled and used in calculating the loss for the model.

4.3 Differences Between My PE Technique And BD-CSPN

My PE technique was strongly influenced by Liu et al. (2020)[3], but there are six major differences between my technique and their methods:

1. They did not incorporate fine-tuning into their work.
2. They used a calculated shifting term (referred to as Cross-Class Bias Diminishing) to shift the embeddings of their query set. I omitted this because it impaired fine-tuning.
3. They used only the top predicted query samples. I used all query samples.
4. They used mainly supervised training while I experimented with semi-supervised training as well.
5. I proposed and used a prototype enhancing loop that they did not.
6. My results for the supervised approach are superior to their results.

Removing the shifting term (Cross-Class Bias Diminishing) allowed fine-tuning the model without the overfiting mentioned by Liu et al. (2020)[3]. Fine-tuning without normalizing the prototypes and in addition, using my iterative technique to enhance the prototypes gives even better results than in Liu et al. (2020)[3]. Liu et al. (2020)[3] did use a wide ResNet-12 as a feature extractor, and although I used only a normal ResNet-12 as a feature extractor, I still achieved better results.

4.4 Summary

In this chapter, I discussed my first contribution, a 4-step framework, in detail. I explained how pre-training the feature extractor helped my framework achieve better performance. I then discussed the second step in my framework: inferring labels from the inferred attributes. A list of my own implementation steps using the tensorized notation helped explain inferences. Further, I discussed fusing the inferred labels with labels produced by the model to which my first approach was applied. Finally, I scaled my output with a learnable parameter.

My second contribution was also included in this chapter. It is a transductive approach that can be used in supervised or semi-supervised learning. My PE technique also uses a pre-trained feature extractor that is fine-tuned with my enhancement loop. This enhancement loop enhances the original prototypes and lasts for 2-6 iterations.

Finally, I explained the differences between my PE technique and the prototype rectification technique, all of which lead to better results than the prototype rectification technique.

In the next chapter, I will explain my experimental design, which includes my baseline models, the original dataset and data preparation steps, sampling protocol for FSL, experimental set up, and the evaluation metrics.

Chapter 5

Experiment Design

In this chapter, I introduce the baselines that I used in my research, including the ones to which I applied my first approach, and one to which I compared my second approach. I also discuss the dataset I used in my research. I also discuss the sampling protocol. I explain my experimental set up, including the parameters used while training models. I conclude by describing the evaluation metric I used to check the performance of my two approaches.

5.1 Baselines

In this section, I introduce the baselines I used in this work. I applied my first approach to four baselines that use ResNet-12[15] as a feature extractor. I compared my second approach to a single baseline.

5.1.1 Prototype Networks

Snell et al. (2017)[2] proposed prototypical networks for Few-Shot Learning (FSL) learning tasks. Their model can be categorized as a metric-based approach. The authors used a CNN network as a feature extractor. In my research, I replaced their CNN network with a ResNet-12[15] as a feature extractor. After extracting features from the support-set images, they used the extracted features as prototypes; they then used the Euclidean distance to

calculate similarities between prototypes and the extracted features of the query set[2].

5.1.2 MetaOptSVM/Ridge

Lee et al. (2019)[5] used MetaOptSVM and MetaOptRidg networks which are Optimization-based approaches. These networks use discriminatively trained linear predictors as base learners instead of nearest-neighbor classifiers[5]. The authors used quadratic programming to represent linear classifiers like SVM and ridge regression, which is done by solving quadratic equations using the qpth library proposed by Amos et al. (2017)[35]. The authors provide two implementations of SVM: one follows the implementation of Crammer et al. (2001)[38], and another follows the implementation of Weston et al. (1999)[39]. In my research, I used the implementation of Crammer et al. (2001)[38] in all my experiments because it provides better results for the baseline and, thus, my approach. For their feature extractor, Lee et al. (2019)[5] used a ResNet-12[15], and I did as well although I replaced their implementation with the implementation in Chen et al. (2020)[4].

5.1.3 Meta-baseline

Chen et al. (2020)[4] found that pre-training the feature extractor and then fine-tuning it for the FSL task boosted the performance of their simple model. They simply used the base categories to pre-train a Resnet then fine-tuned their pre-trained ResNet using a meta-learning set up with a simple cosine similarity to calculate the similarity between the support and query sets. They scaled the cosine similarity using a scale factor and made the scale factor learnable by the network. Chen et al.[4] experimented with different ResNet architectures as feature extractors; I used a ResNet-12[15] in my research.

5.1.4 BD-CSPN

Liu et al. (2020)[3] used the query set as pseudo-labels to rectify their prototypes by diminishing the cross-class and intra-class biases. They pre-trained the feature extractor on

the base classes and used them directly to predict novel classes in a meta-learning set up without fine-tuning. Their approach inspired my second approach, so I used their approach as a baseline to allow a comparison of their results to my results. Liu et al. (2020)[3] used a WideResNet-12/ResNet-12[15] as feature extractors; I used a ResNet-12[15] in my second approach. I did use the results from both WideResNet-12 and ResNet-12[15] to compare their results to mine.

5.2 Dataset and Data Preparation

In this section, I introduce the datasets I used in my research, which includes two benchmark datasets. The details on sampling the two datasets are also provided.

5.2.1 Original Dataset

My research used two datasets. One popular benchmark dataset for FSL is the mini-ImageNet dataset, introduced by Vinyals et al. (2016)[1]. This dataset was constructed from the Image-Net[40] dataset. The mini-ImageNet dataset has 100 categories. Ravi et al. (2017)[41] split the 100 categories into three sets that have been commonly used in the FSL research community. The training set contains 64 categories, a validation set contains 16 categories, and a testing set contains 20 categories. Each category has 600 images. Each image is 84 by 84 pixels. In my research, I used the same splits proposed by Ravi et al. (2017)[41] because they are commonly used in FSL research. For my first approach, I also extracted attributes from the popular NLP dataset, WordNet[36].

The second dataset I used in my research is the tiered-ImageNet dataset, another dataset popular among FSL researchers. It was introduced and split by Ren et al. (2018)[21] into three sets. This dataset was also constructed using the ImageNet[40] dataset. Each image is also 84 by 84 pixels. The total number of classes is 608. Those classes are gathered into 34 high-level clusters. Those clusters are then split into a training set that consists of 20 clusters, a validation set that consists of six clusters, and a testing set that consists of eight

clusters. For my first approach, I also extracted attributes from the popular NLP dataset, WordNet[36].

5.2.2 Sampling Protocol for FSL

Lee et al. (2019)[5] developed an open-source sampler and data-loaders that were originally created by Gidaris et al. (2018)[42]. In my research, I adapted Lee et al.'s (2019)[5] sampler and data-loader by extending them to also provide attributes for my first approach and unlabeled samples for my second approach. When sampling an episode for an FSL training task, K classes are first randomly sampled from the base classes. This random sampling is sampling without replacement. Then, the meta-training episode is sampled by sampling $S+Q$ images from each of the K classes where S indicates the number of images (shots) in the support set and Q indicates the number of query images. In case of a semi-supervised learning task, then the sampler also samples U images (along $S + Q$) where U indicates the number of images in the unlabeled set.

Sampling an episode for an FSL validation task is similar. Another set of K classes is randomly sampled from the validation set without using replacement. Those K classes are then used to sample the meta-validation episode by sampling another $S + Q$. In case of a semi-supervised learning task, then the sampler also samples U images.

Finally, to test the model, K classes are first randomly sampled from the novel class set without using replacement. These classes are used to sample $S + Q$ for a meta-testing episode. U images are also sampled for a semi-supervised learning task.

Looking up hierarchical labeling required using WordNet[36]. I used the K class hierarchical labeling as attributes for all $S + Q$ images in each K class. Therefore, attributes in this research are not per-sample attributes but per-class attributes as done by Lampert et al. (2009)[9].

5.3 Experimental Setup

In my research, the primary development framework was PyTorch[43], which runs using Python[44] programming language. I used a Docker[45] image provided by NVIDIA and containing the CUDA[46] library. All experiments in this work were run using NVIDIA A40 GPU's.

PyTorch[43] is a powerful deep-learning framework that comes with many features as well as support for developing deep-learning models that run on GPU. However, meta-learning in FSL differs from regular image classification tasks in many ways, but one is that images must be sampled as described in Section 1.2 and Section 5.2.2. Popular deep-learning frameworks like PyTorch[43] do not yet have a native support for sampling images for meta-learning tasks; hence, researchers in the community have written their own samplers and data-loaders using the TorchVision[47] library which work with the PyTorch[43] framework.

To obtain attributes for the support set, I used WordNet[36] from the nltk[48] library, modifying the sampler and data-loader in Gidaris et al. (2018)[42] and modified by Lee et al. (2019)[5]. I modified the sampler and data-loader to look up the synset of a category when sampling the K classes. I obtained the hierarchical labeling (the path) leading to the root of WordNet[36] and used that path as attributes for that category. Then, those attributes sampled for the k classes were hot-encoded. The hot-encoding required using the "MultiLabelBinarizer" from the sklearn[49;50] library.

I also extended the sampler and data loaders to sample extra images for the unlabeled set as described in subsection 5.2.2. Sampling for unlabeled images is much like sampling for the support and query images although labels are ignored.

In my research, the open-source implementation developed by Lee et al. (2019)[5] was extended. Their code implements prototypical networks[2], MetaOptNetSVM[5] and MetaOptNetRidge[5]. I added an implementation for the meta-baseline[4] following Chen et al. (2020)[4] and their open-source code. I used the Resnet-12 implementation in Chen et al. (2020)[4] instead of the implementation in Lee et al. (2019)[5] because it is simple and efficient.

When performing experiments, and when pre-training the feature extractor, I used the

same protocol for both approaches. Following the same pre-training protocol used by Chen et al. (2020)[4], When pre-training for mini-ImageNet, I trained a ResNet-12 using an SGD[51] optimizer with a momentum of 0.9 and a weight decay of 0.0005 in a normal classification set up for 100 epochs; the batch-size was set to 128 with the learning rate set to 0.1 and decaying at epoch 90. when pre-training for tiered-ImageNet, I trained a ResNet-12 using an SGD[51] optimizer with a momentum of 0.9 and a weight decay of 0.0005 in a normal classification set up for 120 epochs; the batch-size was set to 512 with the learning rate set to 0.1 and decaying at epoch 30 and 60. After pre-training, the weights from the last epoch were used as final weights for the pre-trained feature extractor and carry out the fine-tuning for my first and my second approach.

When fine-tuning my first approach, I also pre-train the baselines for fair comparison. I used a label smoothing of 0.5 with the pre-trained MetaOptNetSVM/Ridge[5] baselines because it increases accuracy. I normalized the logits of the pre-trained prototypical network baseline by 512 for performance. I scaled the logits of the cosine function by 10 when fine tuning the meta-baseline[4] following the scaling the original authors used. When fine-tuning my approach, I did not apply label smoothing with MetaOptNetSVM/Ridge[5] because it impaired accuracy. I reduced the normalization of the prototypical network[2] from 512 to 5 and set the cosine scaling to 50, making it learnable by the network.

when fine-tuning my second approach, I scaled the cosine similarity of the query set by 20 and the cosine similarity of the unlabeled set by 50 with the mini-Imagenet dataset; I scaled the cosine similarity of the query set by 25 and scaled the cosine similarity of the unlabeled set by 50 with the tiered-Imagenet dataset. I scaled the final logits of the model using a learnable scaling factor that I set to 50. I varied the number in the query set using multiples of 5 in supervised learning. I also varied the number in the query set and unlabeled set in the semi-supervised setting. I varied the number of enhancement loops used in my second approach to discover their effect.

In both approaches, I used the feature extractor pre-trained using the base classes. I fine-tuned for 100 epochs. Each epoch consists of 100 batches of eight episodes. I used an SGD[51] optimizer with a momentum of 0.9 and a weight decay of 0.0005. I set the learning

rate to 0.001, which decays at epoch 90 to 0.0001. I started the validation process at epoch 90. The batch-size of my validation was set to 1000. Each batch included only one episode. I used the same number of support, query, and unlabeled sets in the validation set. I tested my approach by loading the weights of the best validation result. I tested using a batch-size of 10,000. Each batch included only one episode with the same number of support, query, and unlabeled sets that I used in the training and validation sets.

I also pre-trained the ResNet-12[15] feature extractor for the baselines in my first approach for fair comparisons. I fine-tuned the baselines for my first approach.

Results are reported in Chapter 5.

5.4 Evaluation Metrics

In evaluating my results, I reported the average 5-way/5-shot and 5-way/1-shot accuracy percentage with a 95% confidence interval. Accuracy can be calculated using the following equation:

$$accuracy = \frac{correctly\ predicted\ class}{total\ testing\ class} * 100\%. \tag{5.1}$$

5.5 Summary

This chapter details my experimental design, including the four baselines to which I applied my first approach; it also includes a fifth baseline that I used for comparing the results of my second approach. I discussed the dataset and data preparation and how the splits were prepared. Two datasets were used in this research: mini-ImageNet and tiered-ImageNet. The sampling protocol for FSL was detailed for supervised and semi-supervised settings, especially how attributes for my first approach were obtained. For reproducibility, I provided many details about my experimental setup including the environment in which I conducted my experiment and how I trained my models. Finally, I concluded the chapter by introducing the evaluation metric used in this research.

In the next chapter, I present the results of my experiments for both my first and second approaches.

Chapter 6

Results and Analysis

This chapter presents the results of my research along with discussion. Section 6.1 presents the results of my first contribution. Section 6.2 presents the results of my second contribution. My first contribution (BCAT) showed good performance with the baselines as well as how much pre-training can help with performance. Applying this approach can improve pre-trained baselines even further. The second contribution (PE) performed well and varying the number of enhancement loops did affect the performance. The second approach compared well with the baseline. The results include both 1-shot and 5-shot setups.

6.1 Results of BCAT

For BCAT, subsection 6.1.1 presents the results and compares those results to the baselines to study the improvement. Subsection 6.1.2 presents an analysis of different hierarchical levels of attributes to study the effect of attributes and transfer.

6.1.1 Meta-Learning/Fusion Ensemble

For a fair comparison, I used the same pre-trained feature extractor as in my BCAT framework to conduct experiments with the baselines and report the results of those experiments. The increase in accuracy is reported in this section.

Tables 6.1, 6.2, 6.3, 6.4 list in columns from left to right the original results of the baselines as published, followed by the results of using a pre-trained feature extractor with these baselines, then the results of my BCAT framework, and finally, how much accuracy increased over the better of pre-trained and published baselines. The increase shows the marginal gain for my BCAT framework relative to all baselines.

Table 6.1: *Analysis on mini-ImageNet 5-shot. Average 5-way accuracy (%) with 95% confidence interval.*

Model	Baseline	My pre-trained	BCAT	Increase
Prototypical	68.20 ± 0.66	**78.11 ± 0.14**	82.22 ± 0.14	4.11
MetaOptNetSVM	**78.63 ±0.46**	77.21 ± 0.16	80.60 ± 0.14	1.97
MetaOptNetRidge	**77.88 ±0.46**	76.73 ± 0.15	81.14 ± 0.14	3.26
Meta-baseline	**79.26 ±0.17**	78.55 ± 0.15	81.85 ± 0.15	2.59

Table 6.1 shows the results for 5-shot/5-way mini-ImageNet experiments, with pre-training improving the accuracy of prototypical networks by 9.91%. Pre-training for the other baselines shows decreases for the other baselines. The accuracy of MetaOptNetSVM decreased by 1.4%, MetaOptNetRidge decreased by 1.15%, and Meta-baseline decreased by 1%, which was surprising because it also relies on pre-training.

My attribute transfer framework increased the better of the published baselines or my own pre-trained baseline. For prototypical networks, attribute transfer increased the pre-trained baseline by 4.11%, the published MetaOptNetSVM results increased by 1.97%, the published MetaOptNetRidge results by 3.26%, and the Meta-baseline of Chen et al. (2020)[4] by 2.59%. For the Meta-baseline, I reported a conservative underestimate as my marginal improvement because the accuracy (79.26%) that Chen et al. (2020)[4] reported was higher than my results for a pre-trained model.

Table 6.2 shows the results for experiments with 1-shot/5-way mini-ImageNet. Specifically, pre-training improved the accuracy of prototypical networks by 11.05%. Unlike the 5-shot experiments, pre-training also improved the accuracy of MetaOptNetRidge by 0.20%.

Table 6.2: *Analysis on mini-ImageNet 1-shot. Average 5-way accuracy (%) with 95% confidence interval.*

Model	Baseline	My pre-trained	BCAT	Increase
Prototypical	49.42 ±0.78	**60.46 ± 0.21**	66.70 ± 0.22	6.24
MetaOptNetSVM	**62.64 ±0.61**	61.49 ± 0.22	65.19 ± 0.21	2.55
MetaOptNetRidge	61.41 ±0.61	**61.61 ± 0.20**	66.32 ± 0.21	4.71
Meta-baseline	**63.17 ±0.23**	62.30 ± 0.21	69.40 ± 0.22	6.23

like the 5-shot experimental results, pre-training did not improve the accuracy of MetaOptNetSVM; accuracy decreased by 1.15%. The accuracy of Meta-baseline also decreased by 0.87%.

Table 6.2 does show that the attribute transfer framework increased the better of published baselines or my own pre-trained baseline. Attribute transfer improved the results of pre-training prototypical networks by 6.24%. The BCAT also increased the published MetaOptNetSVM results by 2.55% and improved the results of pre-training MetaOptNetRidge by 4.71%. It increased the Meta-baseline of Chen et al. (2020)[4] by 6.23%. As with the 5-shot experiment, for the meta-baseline, the marginal improvement is a conservative underestimate because the accuracy (63.17%) that Chen et al. (2020)[4] reported was higher than my results for a pre-trained model.

Table 6.3: *Analysis on tiered-ImageNet 5-shot. Average 5-way accuracy (%) with 95% confidence interval.*

Model	Baseline	My pre-trained	BCAT	Increase
Prototypical	72.69 ±0.74	**83.09 ± 0.16**	85.37 ± 0.16	2.28
MetaOptNetSVM	**81.56 ±0.53**	79.75 ± 0.17	84.77 ± 0.16	3.21
MetaOptNetRidge	81.34 ±0.52	**81.82 ± 0.16**	85.49 ± 0.16	3.67
Meta-baseline	83.74 ±0.18	**83.84 ± 0.16**	85.50 ± 0.16	1.66

Table 6.3, which shows the experiments for the case of 5-shot/5-way tiered-ImageNet

experiments, that pre-training improved the accuracy of prototypical networks by 10.4%. Pre-training also improved the accuracy of MetaOptNetRidge, which increased by 0.48%. The accuracy of Meta-baseline also increased with pre-training (0.1%). However, the accuracy of MetaOptNetSVM decreased by 1.81%.

The last column in Table 6.3 shows how much attribute transfer increases the better of the published baselines and my pre-trained baseline. The attribute transfer framework increased the accuracy of my pre-trained prototypical networks by 2.28%, the accuracy of MetaOptSVM increased by 2.21%, the results for the pre-trained MetaOptNetRidge accuracy increased by 3.67%, and my pre-training of Meta-baseline increased in accuracy by 1.66%.

Table 6.4: *Analysis of tiered-ImageNet 1-shot. Average 5-way accuracy (%) with 95% confidence interval.*

Model	Baseline	My pre-trained	BCAT	Increase
Prototypical	53.31 ±0.89	**65.49 ± 0.24**	71.96 ± 0.24	6.47
MetaOptNetSVM	**65.99 ±0.72**	61.37 ± 0.24	70.90 ± 0.23	4.91
MetaOptNetRidge	65.36 ±0.71	**65.61 ± 0.23**	72.69 ± 0.24	7.08
Meta-baseline	**68.62 ±0.27**	67.78 ± 0.23	76.45 ± 0.23	7.83

Finally, Table 6.4 shows the experiments for the 1-shot/5-way tiered-ImageNet experiments. Pre-training improved the accuracy of prototypical networks by 12.18%. It also increased the accuracy of MetaOptNetRidge by 0.25%. However, pre-training did not improve the accuracy of MetaOptNetSVM, which decreased by 4.62% and Meta-baseline, which decreased by 0.84%.

The last column in Table 6.4 shows how much the attribute transfer framework increased the better of published baselines and my pre-trained baseline. Attribute transfer increased the accuracy of my pre-trained prototypical networks by 6.47%, the published MetaOptNetSVM by 4.91%, the accuracy of my pre-trained MetaOptNetRidge by 7.08%. The BCAT also increased the accuracy of the Meta-baseline of Chen et al. (2020)[4] by 7.83%. As with the experiments of mini-ImageNet, for the Meta-baseline, a conservative underestimate is

shown as the marginal improvement because the accuracy (68.62%) that Chen et al. (2020)[4] reported was higher than the results for my pre-trained model.

6.1.2 Attribute Transfer

The source of attributes used with mini-ImageNet and tiered-ImageNet is WordNet[36]. As discussed in sections 5.2.2 and 5.3, the class hierarchical labelings were used as attributes. This hierarchical labeling consisted of a path of nodes that started from the class label and led to a root node in WordNet[36]. These nodes were used as attributes for all samples belonging to that class label. A good representation of the WordNet[36] hierarchical labeling

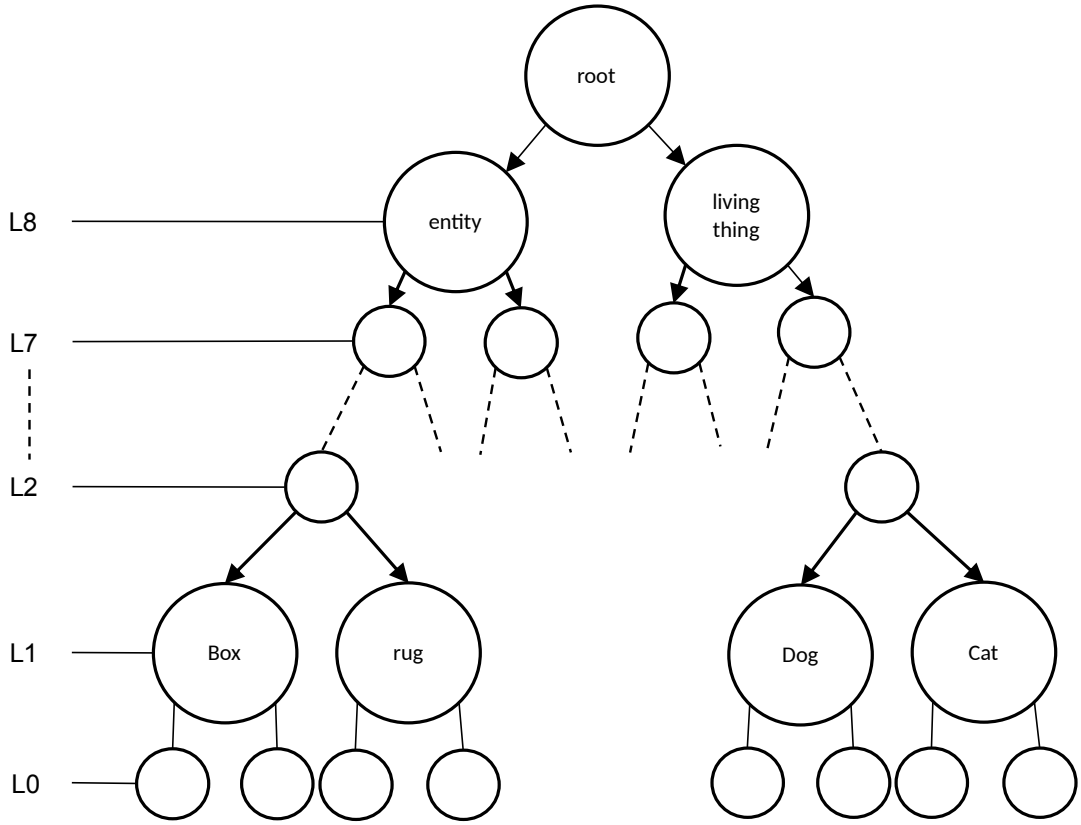

Figure 6.1: *An example of a hierarchy from WordNet showing different levels of attributes. L0 is the label level. Li for $i > 0$ are the internal levels used as attributes.*

is presented in Figure 6.1. The figure shows label nodes on level L0 and internal levels Li for $i > 0$. In experiments, I included attributes from different internal levels of the WordNet[36] hierarchical labeling. These experiments used different levels of attributes for

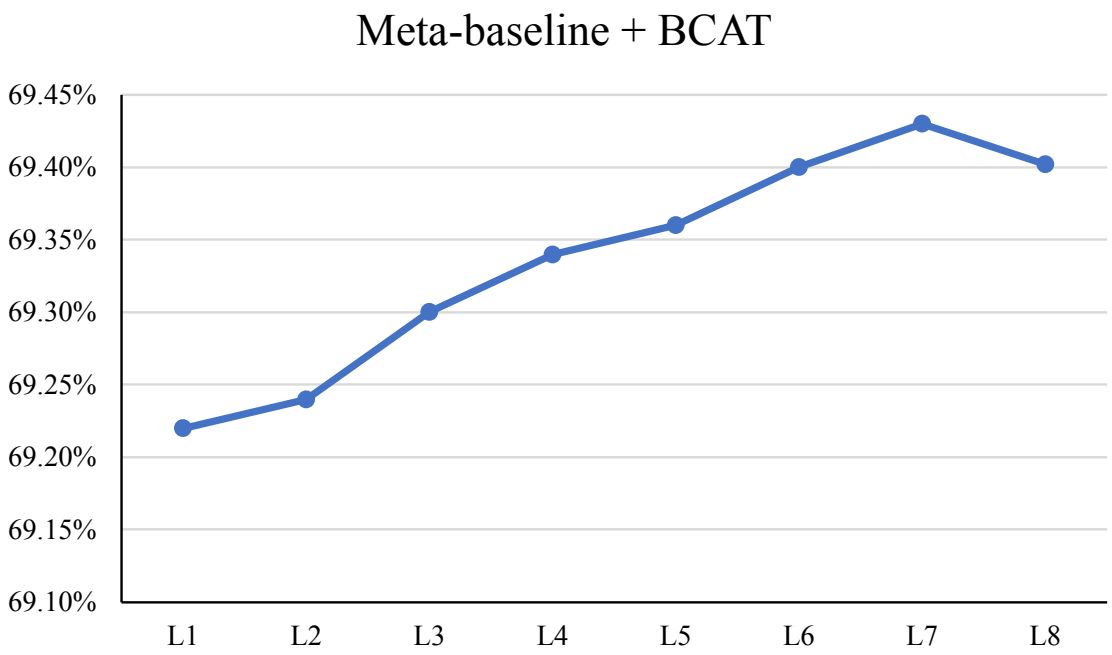

Figure 6.2: *Chart showing incremental increases in accuracy as more attributes are added from higher levels in the hierarchy. This chart is based on Meta-Baseline + BCAT with 1-shot/5-way experiments on the mini-ImageNet dataset*

the complete attribute transfer framework, and the results are graphed in Figure 6.2. Then, more experiments using only the first two steps of the BCAT were conducted and the results graphed in Figure 6.3. Specifically, after using a pre-trained feature extractor (Step 1), I continued inferring labels from inferred attributes (Step 2). In all experiments, I used the Meta-baseline.

Figure 6.2 shows the results of applying the first approach to the Meta-baseline while varying the number of levels from which attributes come. At L1, the model has an accuracy of 69.22%; L2 gives a slightly higher accuracy of 69.24%. L3 improves the accuracy further to 69.30%, followed by 69.34% for L4 and 69.36% for L5. L6 improves still further to 69.40%. L7 has the best accuracy at 69.43%. L8 includes all attributes, giving an accuracy of 69.40% (see Table 6.2).

Figure 6.3 shows the results of applying only the first two steps of the BCAT to the Meta-baseline. Here, the number of levels from which attributes are included also varied. At L1, the model has an accuracy of 62.49%. This is increased at L2 to 62.58%. L3 shows

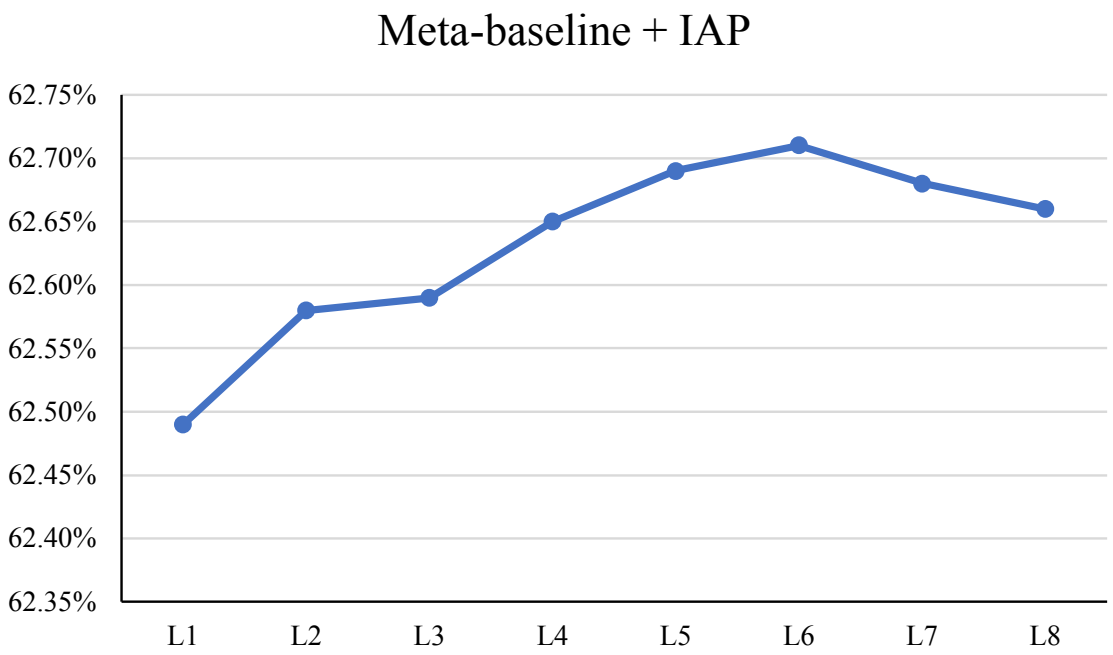

Figure 6.3: *Chart showing incremental increases in accuracy as more attributes are added from higher levels in the hierarchy. This chart is based on Meta-Baseline + IAP with 1-shot/5-way experiments on the mini-ImageNet dataset*

a slight improvement in accuracy at 62.59%. L4 improves to 62.65% accuracy. L5 also improves accuracy, rising to 62.69%. L6 has the best accuracy for the model at 62.71%. That accuracy is reduced slightly to 62.68% at L7, and L8 is still further reduced to 62.66%.

Evidence of transfer learning. Figures 6.2 and 6.3 show the marginal improvements obtained at different hierarchical levels of abstraction with transfer of attributes added. These improvements are incremental, but these experiments demonstrate that accuracy gain attributable to transfer of attributes is achieved when inferring new labels from inferred attributes.

6.2 Results of PE

This section presents the results of my second contribution, the prototype enhancement technique. Subsection 6.2.1 presents the results from using the mini-ImageNet and tiered-ImageNet for supervised and semi-supervised learning tasks and a comparison to the baseline

(BD-CSPN[3]) to reveal any gain from using my PE technique. Subsection 6.2.2 shows the enhancement loop positively affects prototypes. Subsection 6.2.3 presents an analysis of overall accuracy using different numbers of the query set and unlabeled set. Finally, subsection 6.2.4 shows how normalization of prototypes affects my PE technique.

6.2.1 Results and Comparison

In this subsection, I present the results of my PE technique and how it compares to my BD-CSPN baseline. The research done by Liu et al. (2020)[3] is closely related to this research. Liu et al. (2020)[3] reported that fine-tuning causes the feature extractor to overfit, which leads to a decline in overall accuracy. This overfitting can be avoided by removing their feature-shifting factor (or Cross-Class Bias Diminishing[3]) for the query set. Doing so allowed fine-tuning the model without any decrease in overall accuracy and, in fact, led to a slight increase in accuracy. Furthermore, using iterative prototypes enhancement while fine-tuning increased the accuracy further by approximately 8.91%. Tables 6.5 and 6.6 compare my approach with their approach. The authors experimented with ResNet-12 and WRN-28-10 (wide ResNet-12) as feature extractors while my experiments resulted in an increase in accuracy by a large margin, especially for 1-shot learning using a ResNet-12 as a feature extractor.

Model	Backbone	1-shot	5-shot
BD-CSPN[3]	ResNet-12	65.94	79.23
BD-CSPN[3]	WRN-28-10	70.31	81.89
ProtoEnh	ResNet-12	74.85	83.47
ProtoEnh *ssl*	ResNet-12	76.51	84.34

Table 6.5: *Comparison for mini-ImageNet. Average 5-way/5-shot and 5-way/1-shot accuracy (%) for the published results of DB-CSPN[3] and the iterative prototype enhancement technique used in this research.*

Table 6.5 compares the results for mini-ImageNet for the original publication of DB-CSPN

and the results of my PE technique, which shows a gain in accuracy of 8.91% for supervised 1-shot learning and 10.57% for semi-supervised 1-shot learning. For 5-shot learning, accuracy increased by 4.24% for supervised learning and 5.11% for semi-supervised learning. My PE technique shows a gain over their WRN-28-10 results of 4.54% for supervised 1-shot learning and 6.2% for semi-supervised 1-shot learning. For 5-shot learning, the increase in accuracy was 1.58% for supervised learning and 2.45% for semi-supervised learning.

Table 6.6 compares tiered-ImageNet models from published results of DB-CSPN and my PE technique. My technique shows an accuracy gain of 4.29% for supervised 1-shot learning and 5.35% for semi-supervised 1-shot learning. For 5-shot, the increase in accuracy was 1.62% for supervised learning and 1.9% for semi-supervised learning. The comparison to their WRN-28-10 results shows an increase of 1.72% for supervised 1-shot learning and 2.78% for semi-supervised 1-shot learning. In the case of 5-shot, the gain was 0.4% for supervised learning and 0.68% for semi-supervised learning.

Model	Backbone	1-shot	5-shot
BD-CSPN[3]	ResNet-12	76.17	85.70
BD-CSPN[3]	WRN-28-10	78.74	86.92
ProtoEnh	ResNet-12	80.46	87.32
ProtoEnh *ssl*	ResNet-12	81.52	87.60

Table 6.6: *Comparison for tiered-ImageNet. Average 5-way/5-shot and 5-way/1-shot accuracy (%) for the published results of DB-CSPN[3] and the iterative prototype enhancement.*

Figures 6.4 and 6.5 show the increases when the iterative prototype is used over BD-CSPN with different feature extractors in different learning tasks. Figure 6.4 shows that the task with the highest increase was the semi-supervised 1-shot learning task with the mini-ImageNet dataset, which showed an increase in accuracy of 10.57%; the task with the least increase was the supervised 5-shot learning task with the tiered-ImageNet dataset, which showed an increase of only 1.62%. Figure 6.6 shows a similar pattern with the highest

increase at 6.20% and the least increase at 0.40%. These two figures show that the iterative prototype enhancement technique is most effective on the mini-ImageNet dataset when the learning task has fewer shots. The figures also show that the semi-supervised tasks show more increase in accuracy than the supervised tasks.

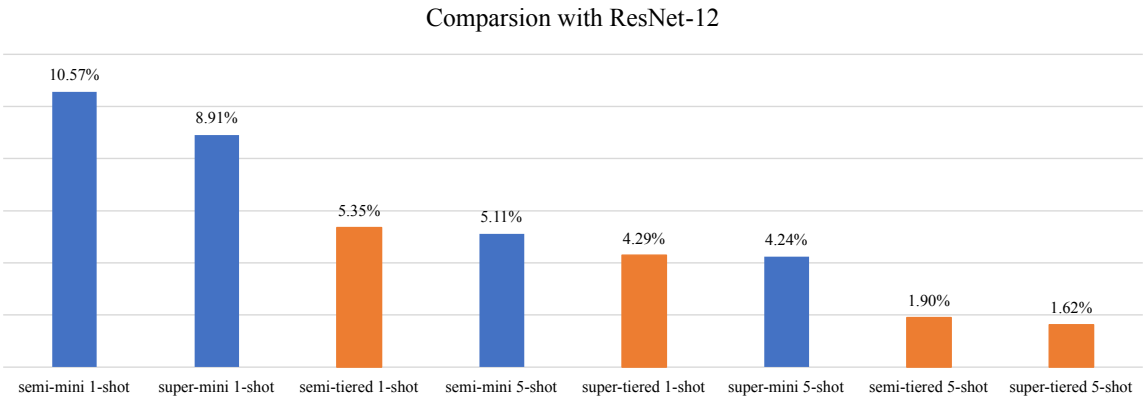

Figure 6.4: *The overall accuracy for my PE technique increased compared to BD-CSPN using a ResNet-12 as a feature extractor. Mini-ImageNet is in blue, and tiered-ImageNet is in orange.*

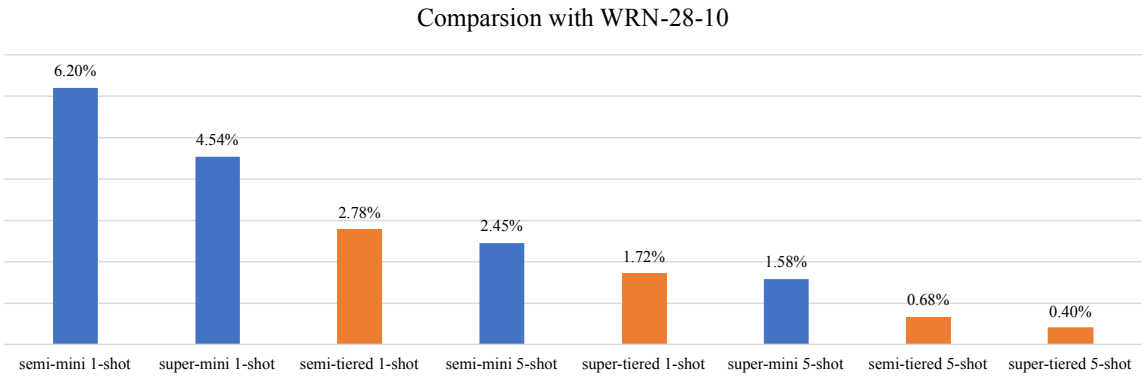

Figure 6.5: *The overall accuracy for my PE technique increased compared to BD-CSPN using a WRN-28-10 as a feature extractor. Mini-ImageNet is in blue, and tiered-ImageNet is in orange.*

6.2.2 Enhancement Loops

To study the effect of the enhancement loops, I experimented with enhancing the prototypes using several different iterations, then calculated the accuracy for each iteration. The fol-

lowing four figures graph the accuracy using 1 to 6 enhancements. Figure 6.6 illustrates the effect of enhancing prototypes more than once with enhancement loops in supervised and semi-supervised 1-shot learning tasks using the mini-ImageNet dataset. For the supervised 1-shot learning task using the mini-ImageNet dataset, acccuracy of 69.71 % was obtained after enhancing only once. Enhancing twice achieved 73.01% accuracy. Three enhancements gave an accuracy of 74.52%, and four provided an accuracy of 74.85%. The accuracy declined to 74.58% with five enhancements and to 74.16% with six. For the semi-supervised 1-shot learning task using the mini-ImageNet dataset, I obtained accuracy of 72.60% was obtained with only one enhancement. Two enhancements gave 75.74% accuracy. Three and four enhancements both provided accuracy of 76.51%. Accuracy declined with five enhancements, providing 75.49% accuracy, and six gave 74.04% accuracy, which is lower than for six iterations for the supervised task. The enhancement loop can add up to 4% improvement for 1-shot learning when enhancing more than once. For mini-ImageNet used for supervised learning, the optimal number of enhancements was four (see Table 6.5). The optimal numbers of enhancements for the semi-supervised learning with mini-Imagenet were 3, 4.

Figure 6.7 shows similar trends when the enhancement loops were applied to supervised and semi-supervised 5-shot learning tasks using mini-ImageNet.

For supervised 5-shot learning using the mini-ImageNet dataset, enhancing once gave 82.69% accuracy. Enhancing twice achieved 83.36% accuracy. Three and four enhancements gave an accuracy of 83.47%. At five enhancements, the accuracy declined slightly 83.45% and six provided 83.43% accuracy.

For the semi-supervised 5-shot learning task using the mini-ImageNet dataset, enhancing one time achieved 83.83% accuracy. Enhancing two, three, and four times achieved 84.40% accuracy. At five enhancements, accuracy declined to 84.33% and at six declined to 84.28% accuracy.

In contrast to 1-shot learning, enhancing more than once gave approximately 1% improvement in accuracy. For supervised learning using mini-ImageNet, optimal enhancements numbered three and four (see Table 6.5). The optimal numbers of enhancements for semi-supervised learning using mini-Imagenet were two, three, and four.

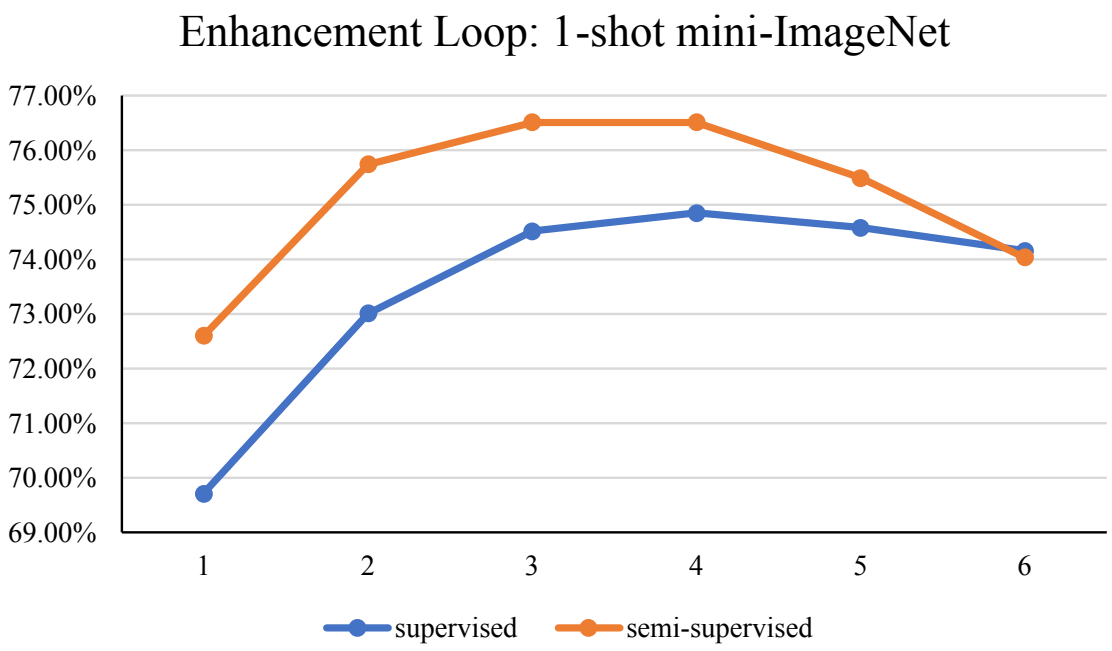

Figure 6.6: *An example of 1-shot learning with several different iterations of the enhancement loop on the mini-ImageNet dataset for supervised and semi-supervised learning. This graph shows the affect of enhancing the prototypes.*

Figure 6.8 illustrates the effect of the enhancement loop as applied to supervised and semi-supervised 1-shot learning tasks using tiered-ImageNet.

For supervised 1-shot learning using the tiered-ImageNet dataset, enhancing only once gave 76.81% accuracy. Enhancing twice achieved 79.49% accuracy. Three enhancements gave an accuracy of 80.40%, and four enhancements provided 80.46% accuracy. Accuracy declined at five enhancements, falling to 80.07% and then to 79.32% at six enhancements.

For semi-supervised 1-shot learning using the tiered-ImageNet dataset, one enhancement gave 78.55% accuracy . Enhancing twice achieved 81.00% accuracy, three enhancements gave 81.52% accuracy, four gave of 81.14% accuracy. Accuracy declined at five enhancements, falling to 80.68% and to 80.26% accuracy for six enhancements. The enhancement loop can improve accuracy up to 4% for 1-shot learning over enhancing only once. For supervised tiered-ImageNet, the optimal number of enhancements was four (see Table 6.6). The optimal number for semi-supervised learning for tiered-Imagenet was three.

Figure 6.9 shows how the enhancement loop affects both supervised and semi-supervised

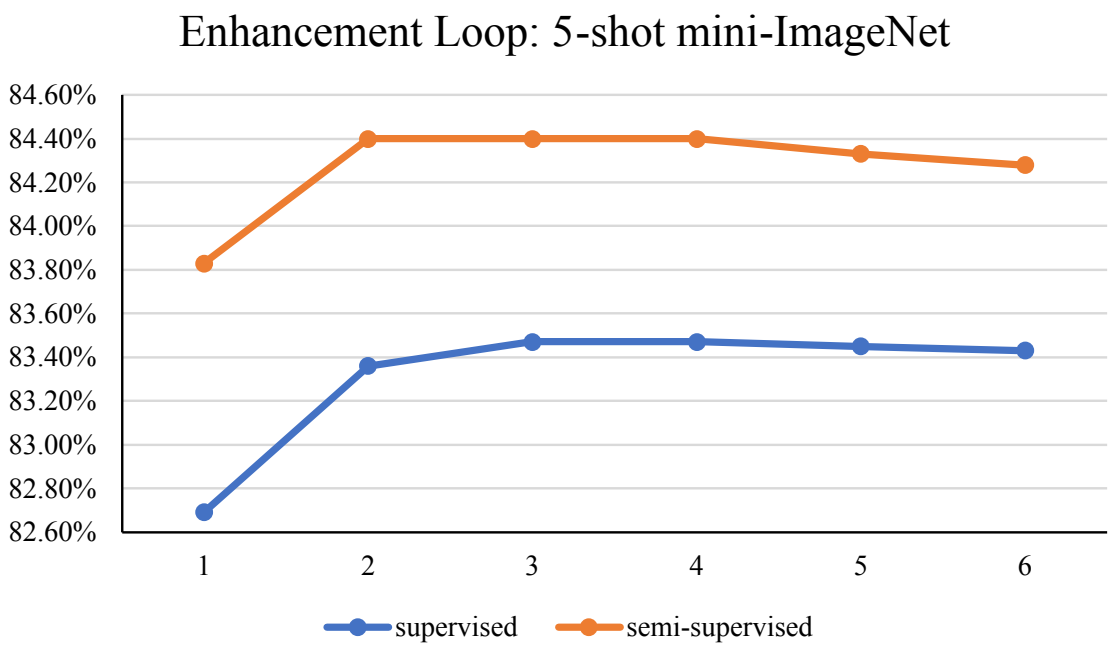

Figure 6.7: *An example of 5-shot learning with several different iterations of the enhancement loop on the mini-ImageNet dataset for both supervised and semi-supervised learning. This graph shows the effect of enhancing the prototypes.*

5-shot learning for tiered-ImageNet dataset.

For supervised 5-shot learning, accuracy was 86.79% after enhancing one time. Enhancing twice achieved 87.25% accuracy. Three enhancements gave an accuracy of 87.32%, and four an accuracy of 87.30%. The accuracy declined still further after four enhancements, 87.25% with five and finally 87.18% with six.

For semi-supervised 5-shot learning, One enhancement led to 87.36% accuracy. Two achieved 87.60% accuracy, and three an accuracy of 87.58%. With four enhancements, accuracy was 87.53%. The accuracy declined still further at five enhancements (87.42%) and finally to 87.34% with six. The enhancement loop can improve accuracy up to 4% for 5-shot learning. For the supervised tiered-ImageNet dataset, the optimal number of enhancements was three (see Table 6.6). The optimal number for semi-supervised learning using the dataset was two.

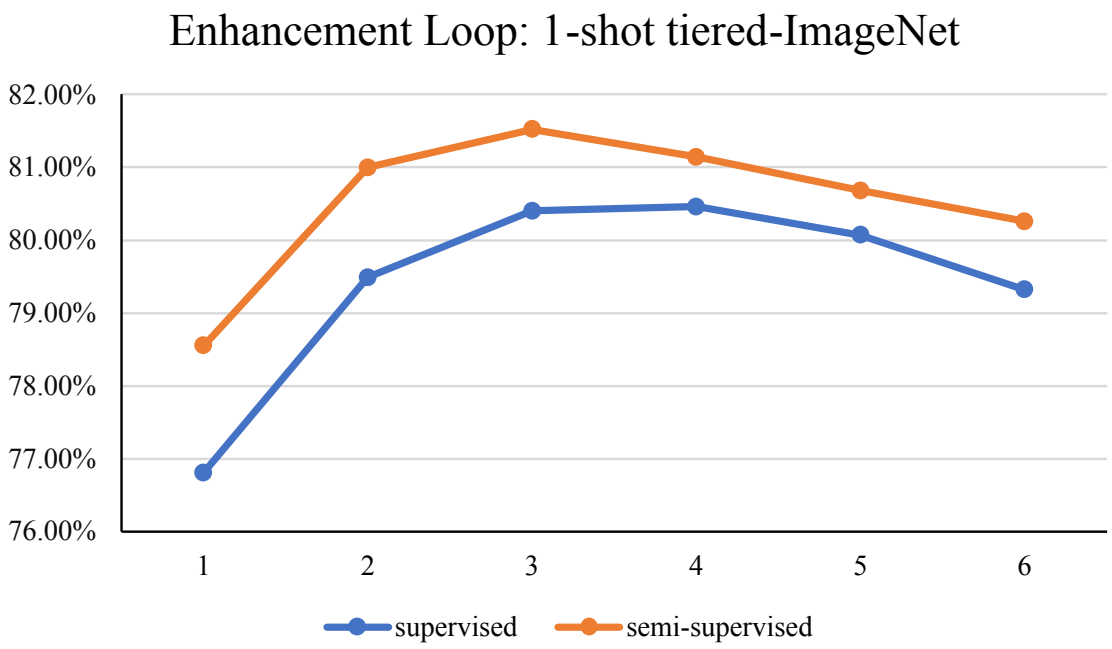

Figure 6.8: *An example of 1-shot learning using several iterations of the enhancement loop on the tiered-ImageNet dataset for both supervised and semi-supervised learning for prototypes.*

6.2.3 Optimal Size of Query Set

The number of the query samples affects the accuracy of supervised and semi-supervised classification. To find out how many query samples are needed, several experiments were conducted, varying the number in the query set for both supervised and semi-supervised learning as well as the number of unlabeled sets for semi-supervised learning. For these experiments, the query set was limited to be 30 samples at most because, in reality, and in few-shot learning applications, more than 30 samples is unrealistic.

For supervised learning tasks, five query samples were used initially and the number of samples increased by five per run until 30 samples were reached. For semi-supervised learning, a pool of 70 samples were set that both the query set and unlabeled set could draw from. So the total number of samples for the query set and the unlabeled set was 70 with different splits where each partition is a multiple of five up to 30 samples for query set. The following four figures show the results.

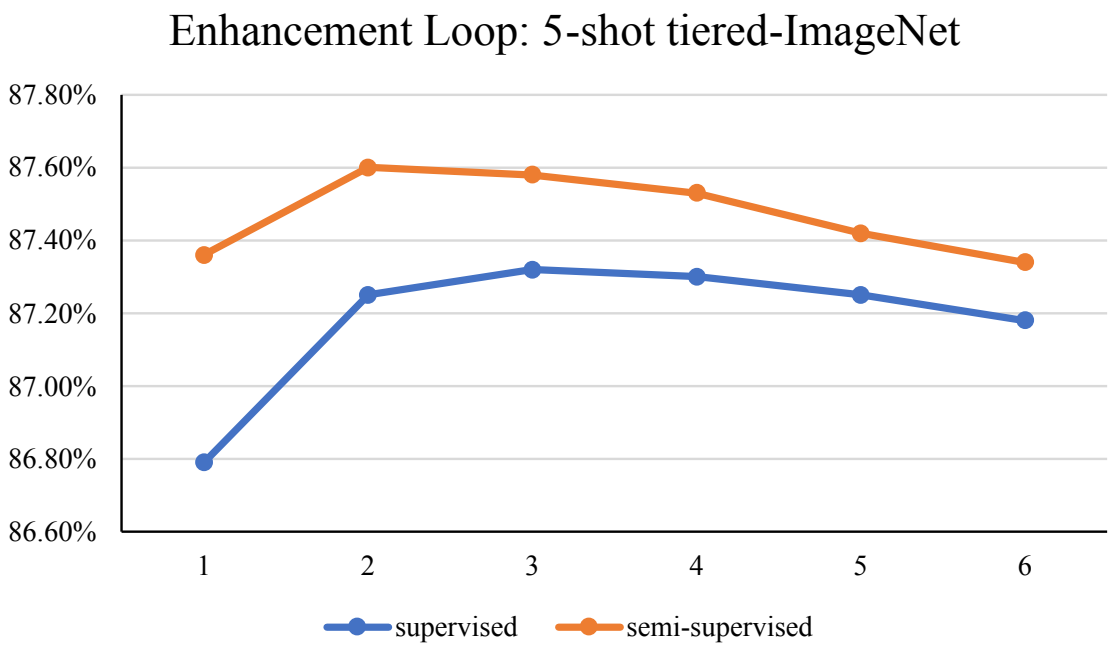

Figure 6.9: *An example of 5-shot learning with several iterations of the enhancement loop on the tiered-ImageNet dataset for supervised and semi-supervised learning.*

Figure 6.10 shows the effect of number of samples in the query set for supervised 1-shot learning tasks. As the figure shows, more samples in the query set does not always improve the performance of the model. Using the mini-ImageNet dataset at five samples the overall accuracy was 69.23%. Increasing the number of samples to ten samples resulted in an accuracy of 71.49%. At 15 samples, the accuracy increased to 72.70%. The accuracy reaches its peak at 20 samples at 74.85%. Then, the overall accuracy decreased to 74.44% at 25 samples and finally to 74.41% at 30 samples.

However, adding more samples always increased the accuracy for the tiered-ImageNet dataset. At five samples, the accuracy was 75.18%. Accuracy increased to 77.42% with ten samples. Accuracy increased to 79.17% with 15 samples, further increasing to 79.74% with 20 samples, 80.16% at 25 samples, and 80.46% at 30 samples. Thus, the optimal query set size in 1-shot learning tasks for the mini-ImageNet dataset was 20 samples as reported in Table 6.5. On the other hand, the optimal size for tiered-ImageNet was 30 samples as reported in Table 6.6.

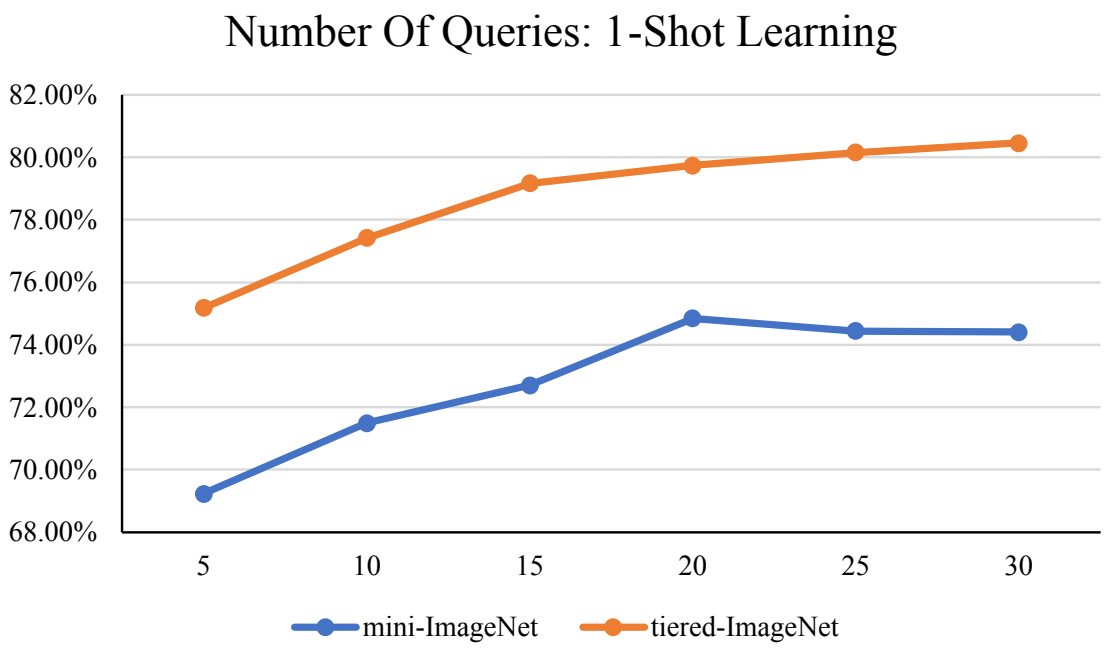

Figure 6.10: *The effect of number of samples in the query set for 1-shot supervised learning tasks using the mini-ImageNet and tiered-ImageNet datasets.*

Figure 6.11 shows the effect of the number of samples in the query set for supervised 5-shot learning tasks. Having more samples in the query set always improved model performance. In the mini-ImageNet dataset at five samples the overall accuracy was 81.28%. Increasing the number of samples to ten gave an accuracy of 82.42%. At 15 samples, the accuracy increased to 82.86%. The accuracy reached its peak with 83.30% at 20 samples. Then the overall accuracy decreased to 83.22% at 25 samples and then increased to 83.47% with 30 samples.

For the tiered-ImageNet dataset, at five samples, accuracy was 85.74%. Accuracy increased to 86.40% with ten samples. Accuracy was 86.84% with 15 samples and then decreased to 86.77% with 20 samples. Accuracy increased to 87.17% at 25 samples and finally to 87.32% at 30 samples. Thus, the optimal query set size for the mini-ImageNet dataset was 30 samples. The optimal number for the tiered-ImageNet dataset was also 30 samples. See tables 6.5 and 6.6 for the results.

Figure 6.12 shows the effect of varying the ratio of the samples in the query set and

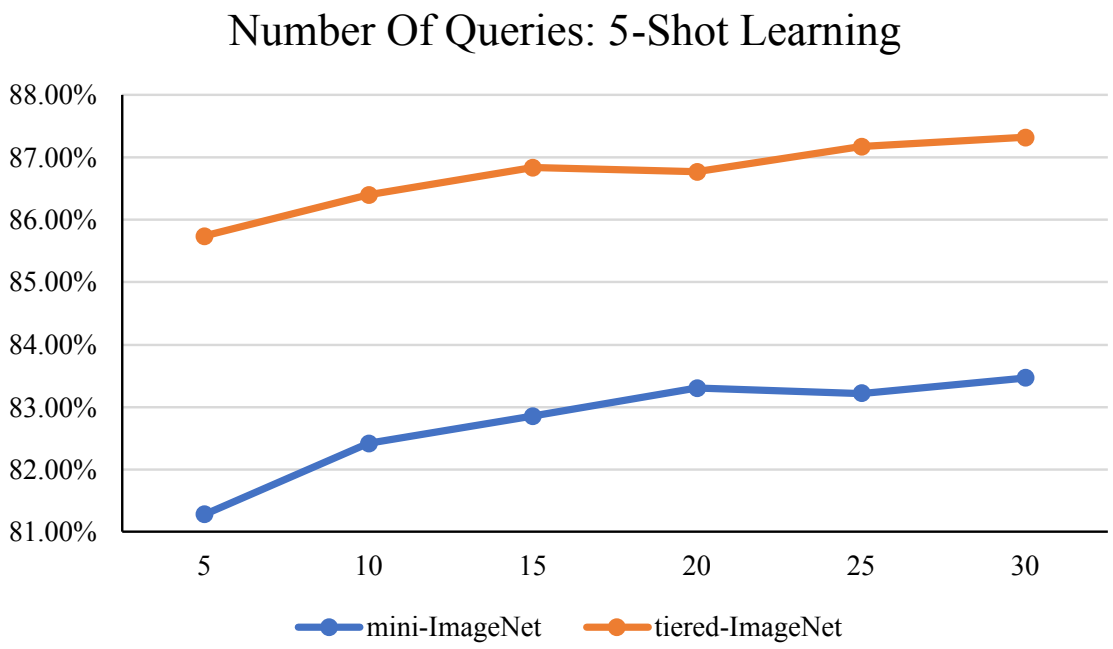

Figure 6.11: *The effect of the number of samples in the query set for 5-shot supervised learning tasks using the mini-ImageNet and tiered-ImageNet datasets.*

the samples in the unlabeled set for semi-supervised 1-shot learning tasks. Having more samples in the query set does not always improve the performance of the model. In the mini-ImageNet dataset, at the 5—65 split, the overall accuracy was 74.93%. Changing the number of query and unlabeled samples to a 10—60 split gives an accuracy of 75.37%. At the 15—55 split, the accuracy increased to 76.38%. At the 20—50 split, accuracy decreased to 75.27%. The overall accuracy reached its peak at 76.51% with the 25—45 split and finally decreased to 75.53% with the 30—40 split.

For the tiered-ImageNet dataset, the 5—65 split showed an accuracy of 81.09%. This accuracy increased to 81.28% with the 10—60 split. The 15—55 split increased accuracy to its peak at 81.52%. Accuracy then decreased to 81.26% with 20—50 split, to 80.97% with the 25—45 split, and finally to 80.87% with the 30—40 split. This shows that the optimal query set size for the mini-ImageNet dataset was the 25—45 split (see Table 6.4). On the other hand, the optimal size for the tiered-ImageNet dataset was the 15—55 split (see Table 6.6).

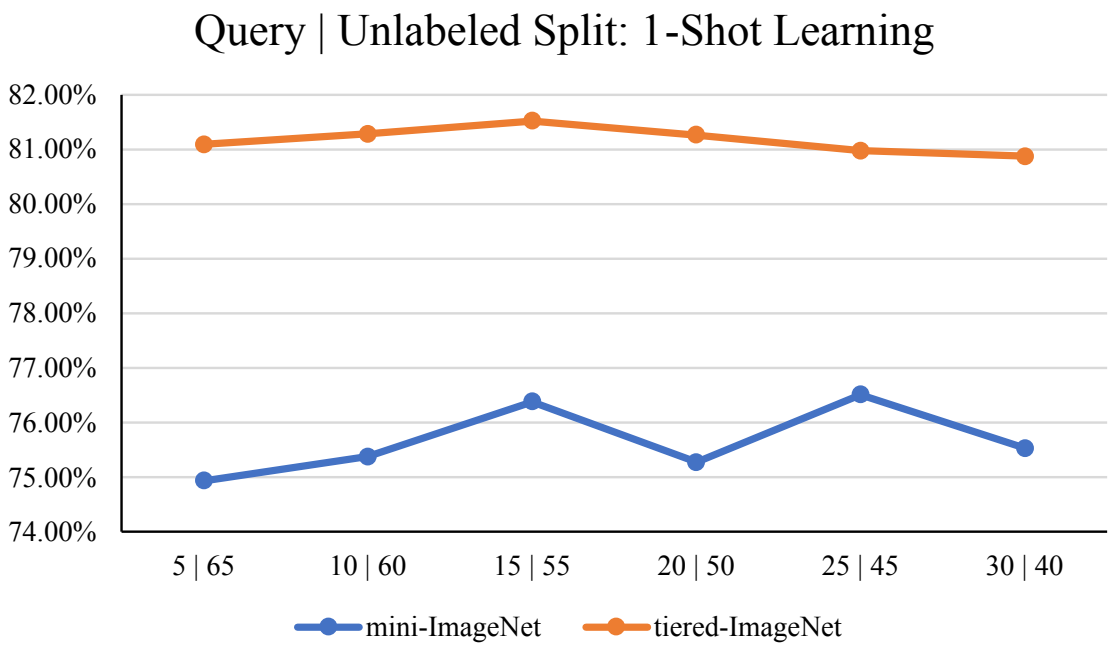

Figure 6.12: *The effect of the number of samples in the query and unlabeled sets for 1-shot semi-supervised learning using the mini-ImageNet and tiered-ImageNet datasets.*

Figure 6.13 shows the effect of varying the ratio of samples in the query set and samples in the unlabeled set for semi-supervised 5-shot learning. Having more samples in the query set does not always improve the performance of the model. In the mini-ImageNet dataset at 5—65 split, the overall accuracy was 84.40%. This was the peak of accuracy for this dataset and split. Changing the number of query and unlabeled samples to a 10—60 split gives an accuracy of 84.15%. At 15—55 samples, the accuracy decreased to 84.06%. The accuracy at 20—50 split was 84.02%. The overall accuracy increased to 84.08% with the 25—45 split and 84.34% with the 30—40 split.

For the tiered-ImageNet dataset, at the 5—65 split, the accuracy was 87.02%. Accuracy increased to 87.18% with the 10—60 split. The 15—55 split increased accuracy to a peak of 87.60%. Accuracy then decreased to 87.43% with the 20—50 split, to 87.24% at the 25—45 split, and finally to 87.23% at the 30—40 split. The optimal query set size for the mini-ImageNet dataset was the 5—65 split, while the optimal size for the tiered-ImageNet dataset was 15—55. See tables 6.5 and 6.6.

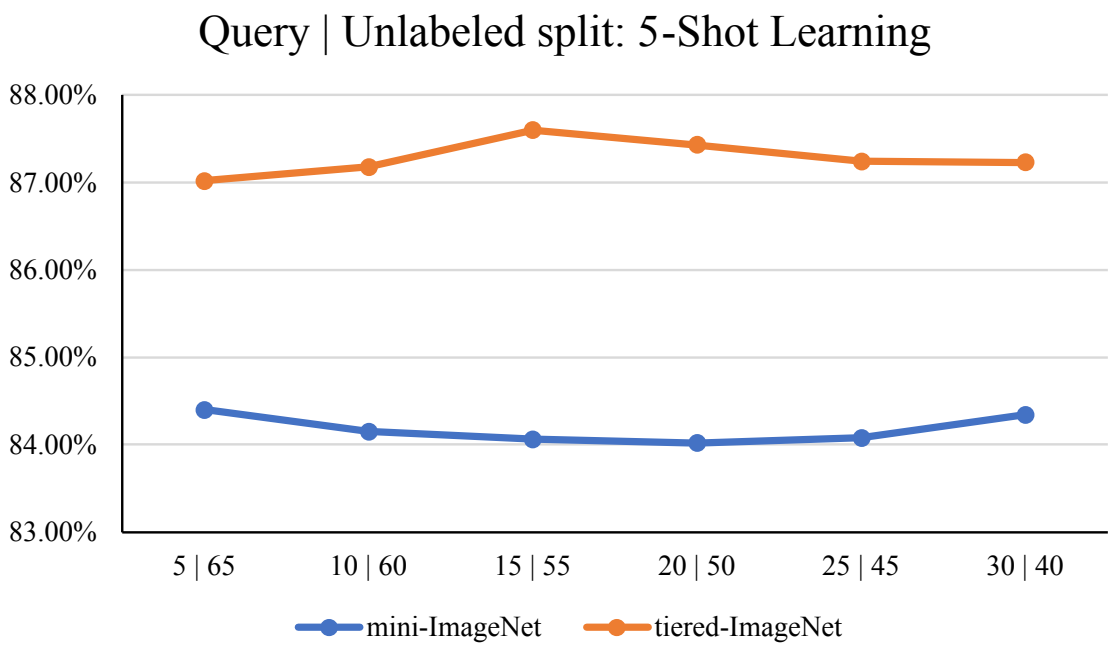

Figure 6.13: *The effect of the number of samples in the query and unlabeled sets for 5-shot semi-supervised learning using the mini-ImageNet and tiered-ImageNet datasets.*

6.2.4 Normalization Effect on Prototypes

Section 4.2.2 notes that embeddings of the prototypes by division are not normalized as in most prototypical techniques[2–4;11] because normalizing nullifies the enhancements over multiple iterations and does not appreciably improve accuracy (even with only one enhancement).

To study the effect of normalizing prototypes, the supervised 1-shot mini-ImageNet learning task was used as an example (see Figure 6.14). When enhancing prototypes only once or twice, normalization does not affect the overall accuracy of the model.

Accuracy was 69.71% for one enhancement with or without normalization. The accuracy then increased to 73.01% after two enhancements but was exactly the same with or without normalization. Furthermore, starting with the third enhancement, normalized prototypes showed less improvement in accuracy. With normalization at three enhancements, accuracy was 74.38% while the prototypes that were not normalized at the same iteration were more accurate at 74.52%. Enhancing normalized prototypes four times gave an accuracy of 74.38%.

Prototypes that were not normalized gave a better accuracy (74.85%) at four enhancements (see Table 6.4). Normalized prototypes that were enhanced five times showed a decrease in accuracy (73.65%); without normalization, prototypes showed an accuracy of 74.58%. Both decreased at five enhancements, but prototypes without normalization were still more accurate than normalized prototypes.

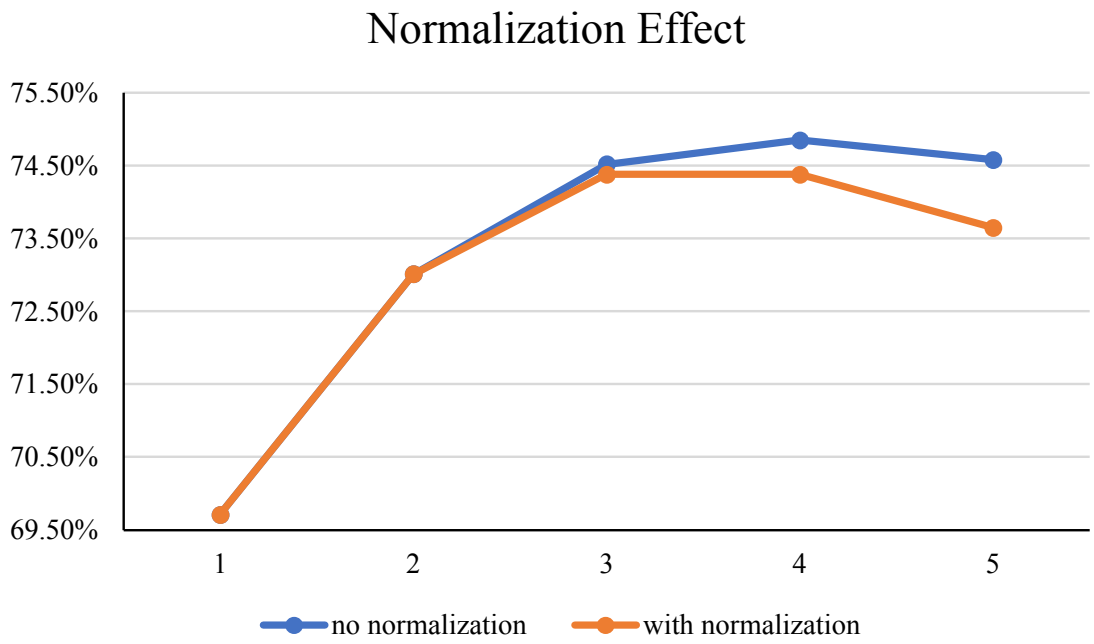

Figure 6.14: *Comparing prototypes that were normalized and not normalized with 1-shot learning using different iterations of the enhancement loop on the mini-ImageNet dataset for supervised learning.*

6.3 Summary

In this chapter, I provided the results of the experiments on the mini-ImageNet and tiered-ImageNet datasets. For my BCAT framework, I showed the performance of baselines using original published results, demonstrating how pre-training can change these results. Finally, I showed how applying my first approach improves those baselines. Additional experiments showed that my first approach does transfer attributes. Tthese experiments varied levels of the hierarchical attributes.

I also presented the results of my PE technique compared with my baselines. Even though my PE technique used only a ResNet-12 as a feature extractor, it outperformed the baselines. The PE technique can add up to 10.57% and 8.91% accuracy for both semi-supervised and supervised learning compared to baselines using a ResNet-12 as a feature extractor. The PE technique also added up to 6.20% and 4.54% accuracy for both semi-supervised and supervised learning compared to the baseline with a WRN-28-10 as a feature extractor. In addition, the proposed enhancement loop improved accuracy. After optimizing the PE technique, the optimal number of iterations required to achieve the best accuracy for all supervised and semi-supervised learning Using the mini-ImageNet and tiered-ImageNet datasets. Finally, my results showed that normalizing the prototypes was not optimal; normalization does not increase accuracy, even with the enhancement loop.

In the next chapter, I recap my findings, relate them to my overall hypothesis, and discuss future work using my contributions.

Chapter 7

Conclusions and Future Work

In this chapter, I present a review of the claims in Section 7.1. I then give an overall summary of my dissertation in Section 7.2. Finally, I present my conclusions and recommendations for future research in sections 7.3 and 7.4.

7.1 Review of Claims

In this section, I review the results of my research. Subsection 7.1.1 provides an interpretation of the empirical results. Subsection 7.1.2 presents a synopsis of contributions to the field (see Chapter 1).

7.1.1 Interpretation of Empirical Results

Chapter 6 presents the results of the experiments that I have conducted in this research work. My first contribution is an attribute transfer framework can be incorporated with existing models. Using four baselines, I applied this attribute transfer, including the meta-baseline. Further, my second contribution is a prototype enhancement technique, applied to cosine-based prototypical networks, also improves accuracy. The Meta-baseline in Chen et al. (2020)[4] can be thought of as a cosine-based prototypical network that incorporates a pre-training strategy also used in my research. Chen et al.'s (2020)[4] results can thus be

used in this section to interpret my empirical results.

My experiments with the cosine-based prototypical network using a pre-trained ResNet-12 (Meta-baseline) provided results reported in Chapter 6 for 1-shot and 5-shot learning tasks with the mini-ImageNet dataset. For 1-shot, I achieved 63.17% accuracy by re-running the Meta-baseline to replicate the results as a baseline. Applying the attribute transfer framework to the Meta-baseline increased accuracy to 69.40%, a 6.23% increase over the Meta-baseline. Moreover, prototype enhancement shows an accuracy of 74.85% for supervised learning tasks, an 11.68% increase over the Meta-baseline; it also gives an accuracy of 76.51% for semi-supervised learning, a 13.34% increase over the Meta-baseline. For 5-shot learning, re-running the Meta-baseline gives an accuracy of 78.55%. Applying the attribute transfer technique gives an accuracy of 81.85%, a 2.59% increase over the Meta-baseline. Applying the prototype enhancement technique to supervised learning tasks gives an accuracy of 83.47%, an increase of 4.92% over the Meta-baseline. Applying the prototype enhancement technique to semi-supervised learning tasks gives an accuracy of 84.34%, 5.79% increase over the Meta-baseline.

This shows the following:

1. My first approach added up to 6.23% accuracy over the meta-baseline, which indicates that my attribute transfer technique can indeed improve all the models used in this dissertation.

2. My second approach added up to 13.34% accuracy over the meta-baseline, which indicates that the prototype enhancement technique boosts the performance of cosine-based prototypical networks.

7.1.2 Synopsis of Novel Contributions

This dissertation provides two primary contributions to the field. These novel contributions are listed here:

1. **I successfully developed a framework for attribute transfer in FSL. This technique is derived from zero-shot learning and can be applied to FSL when used with a meta-learning framework.**

   Lampert et al. (2009)[9] proposed an attribute transfer technique for zero-shot learning in 2009. This technique does not work out of the box when applied to state-of-the-art FSL models with mini-ImageNet and tiered-ImageNet in a meta-learning framework. I successfully derived from Lampert's technique a technique for attribute transfer adapted to Few-Shot Learning (FSL) using pre-training[4], probabilistic fusion[8], and metric scaling[11].

   Metric-based and optimization-based models in FSL can benefit from my framework. My framework can be applied to other models, allowing them to use attributes, something that these models could not achieve before now. The results of my research show that applying my framework increased the accuracy of these models. While this increase is primarily due to the fusion strategy, my framework allows these models to experiment with natural language processing (NLP) methods by using the textual attributes.

2. **I successfully developed a technique for enhancing prototypes of cosine-based prototypical networks using an enhancement loop in supervised and semi-supervised learning.**

   Inspired by the technique developed by Liu et al. (2020)[3], I developed a prototype enhancement technique that uses pre-training[4] and an enhancement loop I developed, thus achieving higher accuracy than Liu et al. (2020)[3] when used with mini-ImageNet and tiered-ImageNet datasets in a meta-learning framework. My technique can be used by supervised and semi-supervised learning as I have shown in this work.

   My second contribution expands on the research of Liu et al. (2020)[3] in several ways. First, Liu et al. (2020)[3] do not incorporate fine-tuning due to overfitting. I did incorporate fine-tuning by omitting the calculated shifting term (which Liu et al. (2020)[3]

call Cross-Class Bias Diminishing) to shift the embeddings of the query set since it impedes fine-tuning. Liu et al. (2020)[3] sampled only the top predicted query samples to enhance the prototypes; I used all query samples by using Bayesian averaging on predictions. Liu et al. (2020)[3] experimented only with supervised learning tasks; I used both supervised and semi-supervised learning. I also used a prototype enhancing loop that increased accuracy further; Liu et al. (2020)[3] did not use an enhancement loop. Finally, my results for supervised learning were more accurate than their results. Liu et al. (2020)[3] used two different feature extractors; I experimented with only one, ResNet-12, and achieved better accuracy.

Future research should include experiments with different feature extractors.

7.2 Summary

This dissertation considers the problem of FSL, explains the meta-learning framework with an example, and offers objectives along with research questions and novel contributions in Chapter 1. I then reviewed research on FSL as it relates to my own research in Chapter 2, then I discussed how my research addresses problems in current research, including complexity and generalization. Chapter 3 provides information about attribute transfer, prototype rectification, and probabilistic fusion, three methods that apply directly to my research. I then explained in detail my two contributions in Chapter 4. I explained my experimental design in Chapter 5. Chapter 6 provides the results

7.3 Conclusion

These experiments on mini-ImageNet and tiered-ImageNet databases using my first approach show that attribute transfer derived from zero-shots learning can be applied to few-shot learning when used with the meta-learning framework. Lampert et al. (2009)[9] had already developed an attribute-transfer technique for zero-shot learning before the recent advances using convnets. This technique, however, does not work out of the box when applied to state-

of-the-art FSL models with mini-ImageNet and tiered-ImageNet databases in a meta-learning framework. I successfully developed an attribute transfer technique derived from Lampert et al. (2009)[9] and adapted to FSL; the technique uses pre-training[4], probabilistic fusion[8], and metric scaling[11]. My results show that my proposed attribute transfer framework not only outperforms the baselines but also outperforms their pre-trained versions in a completely supervised setting. Further, applying the probabilistic fusion mechanism of Zhang et al. (2021)[8] achieves results superior to meta-learning baselines and competitive with the state of the art, **and allowing further improvements using attribute transfer**.

Further experiments on mini-ImageNet and tiered-ImageNet databases using my second approach show that my own technique of prototype enhancement, inspired by Liu et al. (2020)[3], can boost the performance of cosine-based prototypical network by a large margin. My results show that my prototype enhancement technique not only outperformed the ResNet-12 results in Liu et al. (2020)[3], but also outperformed the WRN-28-10 result in Liu et al. (2020)[3].

7.4 Future Work

Given my preliminary findings on choosing transferable attributes from the WordNet[36] hierarchy, a promising direction for future research of my first approach lies in leveraging unlabeled samples within my framework, which I am pursuing in continuing research. First, taking advantage of unlabeled data using semi-supervised and unsupervised methods may boost the performance of attribute transfer because, generally speaking, semi-supervised approaches provide better performance than their supervised counterparts. Second, this framework will further allow experiments with NLP, especially such methods as ontology extraction from corpora or analogical learning cf.[52]. Here, mini-ImageNet and tiered-ImageNet databases provide a data-driven foundation for learning or inferring transferable attributes, and WordNet itself can serve as a starting point because my existing attributes consist of words with hierarchical labeling from WordNet[36].

Given my preliminary findings on prototype enhancements while using ResNet-12 as a

feature extractor, future research could focus on other feature extractors for my second approach. While ResNet-12 is a powerful feature extractor, other, more powerful feature extractors like WRN-28-10, ResNet-18, and ResNet-50 are available. Prototype-based networks like the cosine-based prototypical network rely on embeddings from the feature extractor, so a better feature extractor should give better prototypes, which in turn might benefit more from enhancement.

Finally, the fusion strategy is the most valuable factor in my first approach, and this strategy relies on reconstructing prototypes. Enhancing these prototypes using my second approach should boost the performance of the reconstructed prototypes. Hence, combining my first and second approaches into a third approach that uses attributes and also enhances the fused prototypes using enhancement loops should be a promising direction for future research.

Bibliography

[1] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29: 3630–3638, 2016.

[2] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[3] Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *ECCV*, 2020.

[4] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

[5] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[6] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-Level Semantic Feature Augmentation for One-Shot Learning. *IEEE Transactions on Image Processing*, 28(9):4594–4605, September 2019. ISSN 1057-7149, 1941-0042. doi: 10.1109/TIP.2019.2910052. URL https://ieeexplore.ieee.org/document/8684884/.

[7] Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting Few-Shot Learning With Adaptive Margin Loss. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12573–12581, Seattle, WA, USA, June 2020. IEEE. ISBN 9781728171685. doi: 10.1109/CVPR42600.2020.01259. URL https://ieeexplore.ieee.org/document/9156885/.

[8] Baoquan Zhang, Xutao Li, Yunming Ye, Zhichao Huang, and Lisai Zhang. Prototype completion with primitive knowledge for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3754–3762, 2021.

[9] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

[10] Majed Alsadhan and William Hsu. Few-shot learning in object classification using meta-learning with between-class attribute transfer. *In Proceedings of 14th International Conference on Machine Learning and Computing (ICMLC '22)(To Appear)*, 2022.

[11] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

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

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang

Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[17] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[19] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[20] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[21] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[22] C. Simon P. Koniusz R. Nock M. Harandi. Adaptive subspaces for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] Shell Xu Hu, Pablo Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empirical bayes for few-shot learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[25] Chengming Xu, Chen Liu, Li Zhang, Chengjie Wang, Jilin Li, Feiyue Huang, Xiangyang Xue, and Yanwei Fu. Learning dynamic alignment via meta-filter for few-shot learning.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[26] Yikai Wang, Li Zhang, Yuan Yao, and Yanwei Fu. How to trust unlabeled data? instance credibility inference for few-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[27] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. *CoRR*, abs/1805.10002, 2018. URL http://arxiv.org/abs/1805.10002.

[28] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[29] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. *CoRR*, abs/1805.10002, 2018. URL http://arxiv.org/abs/1805.10002.

[30] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[31] Ping Hu, Ximeng Sun, Kate Saenko, and Stan Sclaroff. Weakly-supervised compositional featureaggregation for few-shot recognition. *arXiv preprint arXiv:1906.04833*, 2019.

[32] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, Long Beach, CA, USA, June 2019. IEEE. ISBN 9781728132938. doi: 10.1109/CVPR.2019.00009. URL https://ieeexplore.ieee.org/document/8954156/.

[33] Yixiong Zou, Shanghang Zhang, Ke Chen, Yonghong Tian, Yaowei Wang, and José MF Moura. Compositional few-shot recognition with primitive discovery and enhancing. In

Proceedings of the 28th ACM International Conference on Multimedia, pages 156–164, 2020.

[34] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[35] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[36] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[37] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. *Advances in Neural Information Processing Systems*, 32:10276–10286, 2019.

[38] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[39] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[42] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[44] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[45] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[46] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 11.4.100, 2020. URL https://developer.nvidia.com/cuda-toolkit.

[47] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL https://doi.org/10.1145/1873951.1874254.

[48] Edward Loper and Steven Bird. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028, 2002. URL http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[51] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[52] Chao-Yeh Chen and Kristen Grauman. Inferring analogous attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, pages 200–207. IEEE, 2014.