Towards Label Free Few Shot Learning : How Far Can We Go?

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Computer Science and Engineering by Research

by

Aditya Bharti
201502014
aditya.bharti@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
January 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Towards Label Free Few Shot Learning: How Far Can We Go" by Aditya Bharti, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisors: Dr. C. V. Jawahar
Dr. Vineeth N. Balasubramanian

Abstract

Deep learning frameworks have consistently pushed the state-of-the-art limit across various problem domains such as computer vision, and natural language processing applications. Such performance improvements have only been made possible by the increasing availability of labeled data and computational resources, which makes applying such systems to low data regimes extremely challenging. Computationally simple systems which are effective with limited data are essential to the continued proliferation of DNNs to more problem spaces. In addition, generalizing from limited data is a crucial step toward more human-like machine intelligence. Reducing the label requirement is an active and worthwhile area of research since getting large amounts of high-quality annotated data is labor intensive and often impossible, depending on the domain. There are various approaches to this: artificial generation of extra labeled data, using existing information (other than labels) as supervisory signals for training, and designing pipeline that specifically learn using only a few samples. We focus our efforts on that last class of channels which aims to learn from limited labeled data, also known as Few Shot Learning. Few-Shot learning systems aim to generalize to novel classes given very few novel examples, usually one to five. Conventional few-shot pipelines use labeled data from the training set to guide training, then aim to generalize to the novel classes which have limited samples. However, such approaches only shift the label requirement from the novel to the training dataset. In low data regimes, where there is a dearth of labeled data, it may not be possible to get enough training samples. Our work aims to alleviate this label requirement by using no labels during training. We examine how much performance is achievable using extremely simple pipelines overall. Our contributions are hence twofold. (i) We present a more challenging label-free few-shot learning setup and examine how much performance can be squeezed out of a system without labels. (ii) We propose a computationally and conceptually simple pipeline to tackle this setting. We tackle both the compute and data requirements by leveraging self-supervision for training and image similarity for testing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of machine learning as a field is to make machines as intelligent as humans. Recent advances in computing power and availability of data have enabled deep neural networks to successfully push the boundary of the state-of-art in a variety of applications ranging from computer vision [16, 78], natural language processing [37, 74], recommendation [40], and computer graphics [72]. Even in rule-based strategy games such as Go [65], chess [66], machines outperform humans. There are a variety of factors contributing to this success. The recent increases in computing power (GPUs and distributed training) allows for training of ever larger networks with billions of parameters [10]. Algorithmic advances [38, 45] and powerful models [31] allow for more efficient domain representation and learning.

However, a large part of this success is driven by the availability of larger and larger labeled data sets in the modern world. From user actions, to social media photos, to purchase history, almost every human activity forms another annotated example for a neural network to learn and improve.

Despite these victories, the data and compute-hungry nature of these systems is far from ideal. From a purely academic standpoint, learning from a few samples is a hallmark of human intelligence. Humans are able to learn physical (object definitions, categories) as well as abstract (mathematics, sciences) concepts from a few examples. Traditional machine learning approaches are unable to transfer knowledge from a few examples. Bridging this gap between human and machine understanding is an active area of research.

The reliance on large data sets is also a barrier to wide adoption of deep learning techniques in a practical sense. Getting high-quality labeled data is challenging for various natural and artificial visual classes [33], and may not always be possible for all domains due to privacy concerns (such as medical images) or simple lack of availability (endangered species classification). This prevents the usage of deep learning methods in low-data regimes.

While state-of-the-art deep learning architectures have found immense success using billions of parameters [44, 68] and large number of parameter update computations [38, 48], this limits deep learning applications to being deployed on powerful machines (or on distributed clusters). This makes them unsuitable for embedded systems, small devices such as interet-of-things (IoT) devices, and end user

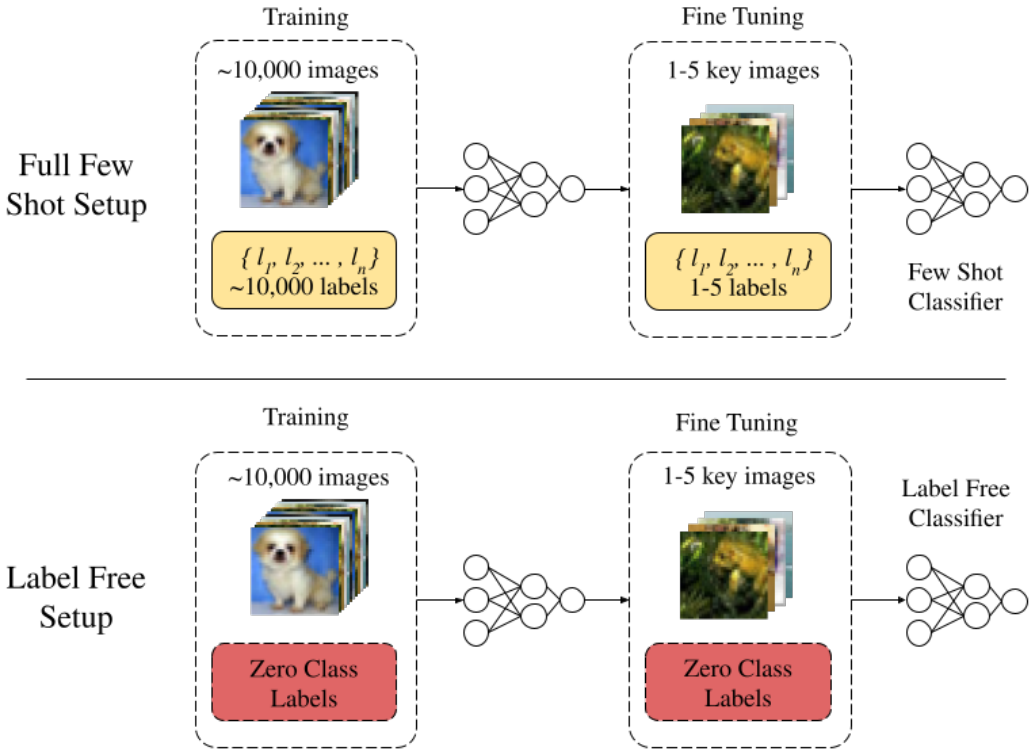

Figure 1.1: Label-free Few-shot Classification: Proposed setting *(Best viewed in color)*

mobile phones for example. Reducing the computational cost involved in training and inference will open up new avenues for deep learning model deployment.

The Few Shot Learning (FSL) domain aims to reduce the data dependence problem by creating learners which are able to generalize and efficiently transfer knowledge from few novel examples. The challenge for the learner is to effectively update its prior experience with a small amount of new data without overfitting to the new samples. In our work, we show how current FSL approaches, while removing the label requirement, still depend on large amounts of data. Our contribution is to propose a more challenging extension of this which aims to tackle both the data and compute issue. Further, our proposed pipeline shows it is possible to achieve good performance at a fraction of the data and compute cost of traditional (both FSL and otherwise) pipelines. This chapter presents a brief overview of the FSL setup before presenting our more challenging pipeline and contributions.

1.1 Proposed Few Shot Learning Setup

Traditional machine learning approaches are supervised [44, 68] and learn using a large number of labelled training samples. The labelled training examples are drawn from a set of classes $C = C_1, C_2, \ldots, C_n$. During testing, a new set of unseen samples is drawn from the same set of classes C and is used to evaluate the learner.

Table 1.1: Comparing number of labels used in traditional few shot pipelines, [1]ImageNet data was used to train the network tested on miniImageNet.

Method	**Dataset**		
	miniImageNet	**CIFAR100FS**	**FC100**
MAML [20]	50,400	48,000	-
RelationNet [73]	50,400	48,000	-
ProtoNet [69]	50,400	48,000	48,000
TADAM [56]	50,400	-	48,000
MetaOptNet [48]	-	60,000	60,000
S2M2 [50]	50,400	48,000	-
Gidaris *et al.* [24]	50,400	48,000	-
Tian *et al.* [76]	50,400	48,000	48,000
AmDimNet [13][1]	1,281,167	-	-

Few Shot Learning (FSL) systems [20, 59, 69, 80] focus on creating learners capable of generalizing to learn novel categories from a small number of labeled samples. A typical few-shot learning setup [20, 69, 80] also consists of a training and testing phase. However, instead of training and inference on the same set of classes, the two sets are disjoint. The learner has access to a potentially labeled set of classes $C_{train} = \{C^1_{train}, C^2_{train}, \ldots\}$ where each training class C^i_{train} has a large number of samples.

At test time, the network is presented with a few labeled examples (typically 1 to 5) from a different set of testing classes $C_{test} = \{C^1_{test}, C^2_{test}, \ldots\}$ such that $C_{train} \cap C_{test} = \emptyset$, and then performance is evaluated on the unlabeled samples from this set of unseen classes. Given the limited number of labels, the learner must quickly generalize to classify arbitrary examples from the test class.

This model of training on one set of possibly-labelled examples, using a small number of novel class examples, and testing on the novel class set represents a real improvement compared to traditional approaches. The number of labelled examples required for the learner to effectively adapt is greatly reduced. However, this pipeline overall still uses a lot of labelled data. The data burden has simply been shifted from the testing set to the training set. Traditional methods in this area require up to $60,000$ labels from standard benchmark datasets [56, 80] despite the 'few-shot' moniker. (See Table 1.1 for a comparison.) While this reduction in data requirements is an improvement which should not be understated, we show it is possible to push the limits of this setup. This thesis proposes a learning framework which can effectively learn to classify using almost no labels overall. In addition, we also keep our training and testing pipelines as computationally simple as possible so that it can be deployed on computationally constrained devices as well. Our proposed training and inference pipelines are thus thoroughly label-free, computationally simple, and achieve competitive performance.

1.1.1 Label-Free Training

Traditional few-shot pipelines have a very label-intensive training phase. We directly address the label requirement at training time by leveraging advances in self-supervision [14, 30, 35, 41] and metric

learning [8, 69, 73, 80] to allow our classifiers to learn effective visual representations without labels. Instead of labeled examples, self-supervised pipelines use a supervisory signal from the data itself, usually via a number of pretext tasks [18, 26, 47, 55, 91] such as colorizing images and jigsaw puzzles and classifying image rotations. Typically, the task is chosen such that it is very simple to compute the ground truth. Metric learning approaches learn to embed inputs in a high dimensional space such that semantically similar images are embedded close to each other in the latent space. Combining these two areas, we choose image similarity as our pretext task. It allows the successful learning of useful visual representations [41] and does not require manual labeling.

We formulate our image similarity pre-training task using recent advances in contrastive learning approaches [14, 30, 35, 75]. The basic premise is that a learner can improve on image similarity by learning to compare and contrast similar images. Contrastive learning frameworks improve the basic image similarity task by applying simple distortion transforms on input images to generate a set of distorted images. Each pair of distorted images forms an input to the learner, which must recognize if both pair elements are from the same original undistorted input image. By learning to predict image similarity in the presence of distortions in the input data, the network can effectively distill information, making it suitable for quick generalization on novel classes.

This approach of a self-supervised contrastive learning training phase allows for a totally label free training pipeline.

1.1.2 Label-Free Testing

At test time, few-shot learning frameworks must generalize to novel classes given only a few (possibly labeled) examples while avoiding overfitting on the limited data set. Traditional architectures, which update a few billion parameters on every update, have a tendency to overfit on data and present a significant computational burden preventing their deployment on low-power devices. Computationally simple classifiers such as nearest neighbors [81], clustering [69], and attention kernels [80], are effective in this low data regime. They also present a lower barrier to deployment on less powerful machines.

At test time, our classifier is presented with an N way, K shot task of K labeled images (called **key** images) from N classes each, followed by 15 unlabelled images (called **query** images) to be classified. We use various test time classifier variants (such as nearest neighbors [82] and attention kernels [80]) to select the most similar key image for every query image. The class of this selected key image is the predicted query image class, which allows us to maintain our label-free setting while keeping the inference pipeline computationally simple. Our ablation studies in Chapter 5 also investigate a simple extension to this label-free setting using limited label information.

1.2 Contributions

We make the following contributions to this thesis:

1. We present a new, more challenging, label-free few-shot learning setup. In addition to bridging the gap between human and machine intelligence, this also expands the fields of application which can benefit from deep learning systems.

2. Our proposed pipeline is both conceptually and computationally simple, making it easy to adapt to new areas. Our label-free framework can be used with any inference and training technique that does not use labels. In Chapter 5, we showcase this flexibility by using a separate clustering-oriented training task, instead of our proposed self-supervised contrastive learning task.

3. By leveraging self-supervision for training and image similarity for testing, we achieve competitive performance while using *zero* training or testing labels. This is *10,000 times* fewer labels than existing state-of-the-art. (See Table 1.1)

4. In our ablation studies, we examine extensions of our pipeline and show how including limited label information at test time and using a clustering-oriented task at training time influence performance.

We will be diving into the details of our framework and contributions to our thesis. The next chapter explores existing literature from related fields which have motivated our work, and presents how our contributions are novel in that context. We end with a brief qualitative assessment of our results and avenues for future work.

Chapter 2

Related Work

Few-shot learning literature is highly diverse [82]. There are various approaches to generalizing from a limited number of samples while avoiding overfitting the limited testing data. This chapter presents a broad classification of relevant few-shot literature before delving into the other related areas of metric learning, self-supervised learning, and contrastive learning, which have motivated our work.

2.1 Few Shot Learning

The goal of a machine learning program is to learn from experience such that it measurably improves performance over some class of tasks [52]. Typically, the learner incorporates experience using a supervisory signal from the input examples. The supervisory signal may or may not be directly derived from the performance measure itself. This setup is called supervised learning and requires labelled data samples where the labels form the supervisory signal. As such, traditional machine learning applications require a large number of samples with labels for the source of supervised information.

For a variety of reasons outlined in the introduction, the easy availability of labels is not a given in all domains. For this reason, the field of Few-Shot Learning (FSL) aims to perform machine learning under the constraint where only a few labelled examples are present. It is important to make clear what exactly we are constraining here. The learner is allowed to access labels for any examples not related to the final task at hand. The few-shot constraint only applies to the final task over which we measure the performance. For example, in a typical few-shot image classification setup, the learner can initially be trained using labelled images from the full ImageNet dataset, but performance is evaluated on a completely novel set of classes, of which only a few labelled examples per class are present.

In the context of deep learning, a neural network incorporates new information by using the supervisory signal to update the network parameters. Typically, this is done via a loss function related to the performance measure and task at hand, and the supervisory signal comes from the labelled examples. In this setup, neural network optimization is essentially a search through the space of all parameter combinations. Since the parameter search space is huge, finding a reasonable optimum in a reasonable amount of time is a challenge. The most straightforward approach is to use as much supervision as possible, and

use powerful machines which can perform more updates in the same amount of time. As a result, deep learning has had the most success in domains with good availability of labelled examples, coupled with the availability of increasingly powerful compute.

In the FSL setup, the lack of supervisory signal presents significant challenges. Without enough signals to guide the search, the network may find and settle in a local optimum and overfit. To generalize well, the learner must effectively distill knowledge from the limited samples present while avoiding overfitting. There are three broad categories of approaches that deal with this problem in different ways:

1. Data-focused approaches: This is the most straightforward category of approaches which effectively increases the dataset size. More data implies more signals to guide the search. Intelligent techniques are used to generate (or find) more data to train on in such a way as to improve the performance on the final few-shot task. This extra data can either be artificially generated or sourced from related tasks with labeled data available (via label propagation, for example).

2. Model-focused approaches: This second category focuses on the internal representation of the learner. Having fewer parameters to optimize reduces the size of the search space, and a better optimum can be found with limited data. Simpler architectures can be used instead of the usual billion-parameter models, which will not overfit. Generalization can also be achieved by training the learner on multiple tasks, learning with external memory, or learning to embed. This category also encompasses self-supervised and contrastive learning methods.

3. Optimizer-focused approaches: This final category aims to quickly find an optimal path through the parameter search space in as few updates as possible, allowing the model to generalize using limited data. This is akin to searching 'faster' through the search space. In many cases, the optimizer itself is a complex neural network that learns to make large meaningful updates based on limited data. Some approaches learn an optimizer itself, while another class of approaches called "meta-learning" aims to learn a model that quickly learns given limited data.

We will now go over these approaches in some detail.

2.1.1 Data Augmentation

Also known as data augmentation, this class of approaches adds extra labelled training samples for the network. New samples are generated from existing samples by modeling inter-class and intra-class variations using hallucination [29], feature transfer [64], and variational autoencoding [46] methods. It is important that the extra samples are related to the existing data set, otherwise the network may optimize for the wrong representations. Informative samples can be chosen from a related weakly supervised or unlabeled dataset (when such a dataset is available) and existing labels can be propagated to the new dataset to yield new labeled samples [19, 57]. It is also possible to use completely different datasets to guide the learning process [22, 77], effectively increasing the labeled dataset size.

However, the limitation of such approaches is that they require the presence of related datasets, reliable methods to select informative samples, and correctly propagating labels from known to unknown examples. Since there are only a few truly labelled samples and the rest need to be somewhat artificially generated, the learner is extremely sensitive to the label propagation technique, the example generation technique, and also the similarity between the task at hand the the related dataset. By increasing the dataset size using these approaches, these methods also consure more compute resources both in the data generation and network training steps. In our proposed approach, we avoid both of these limitations by using no labels at all and using extremely simple pipeline components.

2.1.2 Constraining Parameter Search Space

The main reason large number of labels are needed in the first place is due to the large size of the parameter search space. A smaller search space is easier to search through, and if an optimum set of parameters exists within this constrained space, it is more likely to be found using the limited data. While using simpler architectures with fewer parameters is the most straightforward method to constrain the search space, there are various methods in this category. Multi-task learning methods leverage knowledge from multiple tasks by learning them simultaneously [12, 93], with knowledge being shared between tasks by sharing [6,53,92] or tying parameters [87] between various task-specific learners. Since the set of parameters optimal for all tasks is strictly smaller than the optimal settings for any specific task, this restricts the size of the parameter search space. Deep learning networks encode a lot of redundant information in their parameters. Another way to reduce the parameter search space is to reduce the dimensions of the learned representations. Embedding learning methods [36] learn a semantically meaningful embedding function into a lower-dimensional space such that similar inputs are embedded closer than dissimilar ones. Both Matching Networks [80] and Prototypical Networks [69] fall into this category: [80] learns train and test specific embeddings, and [69] learns to embed class prototypes directly. Memory networks [51, 58] directly store training samples in external memory to fetch and compare again at test time. However, choosing the right examples to store is a non-trivial task [86,96], and the accuracy of the final learner is extremely sensitive to this choice. It is also possible to directly learn the underlying input data probability distribution [60, 62].

These methods place heavy constraints on the model. Labelled data from separate tasks related to the main task at hand must be found. It can be difficult to analyze if a learned embedding is semantically meaningful for the final classification task. Choosing a specific network architecture, embedding method, or probability model also ties the model into a certain representation. Even though the search space has been reduced, there is no guarantee that the remaining space is the most optimal for few-shot tasks. Our framework places no such constraints on the model, availability of tasks, or category of learned representations, allowing our framework to be more generally applicable.

2.1.3 Guiding Parameter Search Strategy

The core problem with a large search space is that there are more parameter combinations to search through before finding an optimal one. Instead of using more samples to guide the search or constraining the search space, this class of methods aims to make the search 'smarter'. We guide the search strategy for optimal parameters by either learning a good initialization or a better parameter update strategy. Approaches such as [11,42,89] directly fine-tune parameters learned from a different but related pre-text task. A hugely popular class of methods called meta-learning methods [20] aims to "learn a learner". A "meta" learner is trained from multiple related tasks to output a "base" learner, which can be optimized for specific novel tasks in relatively fewer gradient updates [7, 54]. Advances in the meta-learning space focus on incorporating task-specific information [49], modelling uncertainty [21, 28, 61, 90], or improving training strategy [2, 4]. Finally, [1, 59] learn an optimizer that directly guides parameter updates of the task-specific learner. In contrast, we do not update any network parameters at test time.

In addition to a large amount of labeled training data, this class of approaches requires significant computing resources. In addition to training more complex optimizer and learner networks, we also need to perform parameter updates at inference time. To keep our inference pipelines computationally simple and investigate how far a few-shot framework can go without using labels, we have opted for simple test time classifiers and no parameter updates at inference time.

2.2 Metric Learning

Typically, once a machine learning system has been trained on a specific set of classes, it cannot easily generalize to other classes or tasks. This is due to the nature of traditional loss functions which try to align network predictions to actual labels from each class. Instead of learning to directly classify, metric learning methods "learn to compare". Metric learners embed inputs in a semantically meaningful embedding space, which groups similar inputs close to each other and dissimilar inputs far apart. This allows them to sidestep the need for labels by using image similarity as the supervisory signal. While it is possible to assign image similarity based on image classes, this is not always necessary. In our work, we use a notion of image similarity which does not use labels at all.

Since the learnt embedding is 'semantically meaningful', the metric learner must embed similar inputs close to each other in the embedding space, and dissimilar inputs far away from each other in the final embedding space. To that end, we need a notion of a distance metric in the embedding space, and a way to assign similarity scores between pairs of inputs. The starting point of metric learning methods is the Mahalanobis distance metric. Given two inputs x_i, x_j the Mahalanobis distance d_M between them is given by:

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

Since d_M is a distance metric, it needs to satisfy certain properties:

$$\begin{aligned} d(x, y) = 0 &\implies x = y \\ d(x, y) &= d(y, x) \\ d(x, z) &\leq d(x, y) + d(y, z) \end{aligned}$$

These constraints imply that the matrix M needs to be symmetric and positive semi-definite and can be decomposed as $M = W^T W$. The Mahalanobis distance metric can then be decomposed as follows:

$$\begin{aligned} d_M(x_i, x_j) &= \sqrt{(x_i - x_j)^T M (x_i - x_j)} \\ &= \sqrt{(x_i - x_j)^T W^T W (x_i - x_j)} \\ &= \sqrt{[W(x_i - x_j)]^T [W(x_i - x_j)]} \\ &= \|W x_i - W x_j\|_2 \end{aligned}$$

The matrix W represents a linear transformation on the inputs x_i and x_j. The Mahalanobis distance in the original space can be represented as a simple Euclidean distance in the transformed space. Methods which learn such linear transformation impose fewer constraints on the input data and have been shown to be robust to overfitting [5]. Unfortunately, the linearity of the transform imposes constraints on the set of representations that can be learned. By leveraging non-linearity, metric learners can learn more representations possibly better suited to the data and task at hand. In addition to kernel methods to introduce non-linearity, metric learners replace this linear transformation matrix W by a non-linear transformation function $f(\cdot)$, allowing a greater richness in the possible set of learned representations. This allows for more natural data modelling from a wide range of domains [83, 85].

$$d_M(x_i, x_j) = \|f(x_i) - f(x_j)\|_2$$

It is also possible to choose non-Euclidean measures of distance in the final embedding space to better model similarity of embeddings. Cosine similarity is one such example:

$$s(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|}$$

By learning to predict image similarity in this way, a model can use the similarity scores to label instances of novel classes by comparing them to a few known examples. It is also an effective pretext task for few-shot learning [41]. Metric learning models use a variety of techniques in place of the linear transformation function, ranging from cosine similarity [80], Euclidean distances [69], network-based [73], ridge regression [8], convex optimization based [48], or graph neural networks [63]. Combined with supervised pipelines, regularization techniques such as manifold mixup [50] also improve performance on few-shot tasks. Metric learning methods on their own need not be label-free, as they aim to minimize intra-class distances and maximize inter-class distances. In the next section, we present a brief overview of motivating self-supervised techniques. Our proposed framework combines metric learning methods with self-supervised techniques to create a completely label-free setting.

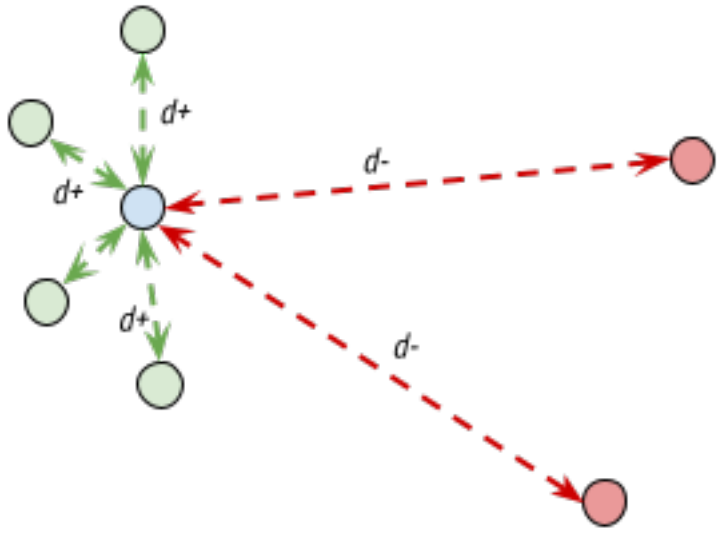

Figure 2.1: Metric Learning: Distance between similar samples must be minimized in the embedding space, while distance between dissimilar samples must be maximized. *(Best viewed in color)*

2.2.1 Sample Selection for Metric Learning

Metric learners must embed similar sampler close together and dissimilar samples farther apart in the learned embedding space. To achieve this, they are trained on pairs of samples from the training data, as opposed to single samples. A pair of samples which need to be grouped closely together is designated a positive pair, while a pair of samples which needs to be embedded far apart is designated a negative pair.

The strategy to choose positive and negative samples has a great effect on the effectiveness of the final learned system [67]. It is important to sample pairs with a high discriminative power. It has been found that easy to discriminate samples have little effect on the learned parameters. Randomly creating pairs from the input data does not achieve the best results.

There are various sampling approaches in the literature to create positive and negative samples from the input data. The goal is to ensure the network is always presented with hard to discriminate, informative sample pairs. One approach is hard negative mining [94], where false positive sample pairs (negative sample pairs which are classified to be positive) are chosen and fed into the network. The goal is to let the network adapt to discriminate between dissimilar inputs which have been classified as similar.

Another approach is to use triplets [32]. Instead of pairs of samples, a triplet of inputs consisting of anchor input, positive sample, and negative sample is created. The network is simultaneously trained to minimize distance between the anchor and the positive sample, while maximizing distance between the anchor and negative sample using a triplet loss.

2.3 Self-Supervised Learning

In machine learning, a supervisory signal is essential to measure performance and establish a clear definition of what constitutes an "optimal" outcome when searching for optimal parameter values within a search space. In a supervised classification setting, the class labels themselves serve as the supervisory signal. At its core, the supervisory signal is what guides the machine learning algorithm towards the desired outcome.

When evaluating the performance of a model with a specific parameter set on a classification task, the measure of success is how accurately it predicts the class labels. The model's accuracy is a measure of the its ability to correctly identify inputs and predict the class labels, and is therefore a good measure of overall performance. In self-supervised learning, the supervisory signal along with the performance measure is used to fine tune the model and ensure it is optimized for the task at hand.

Self-supervised methods utilize unlabelled data to learn useful representations. A popular approach is to use a supervisory signal from the data itself and train the learner on pretext tasks that help it learn useful internal representations. An effective self-supervised setup requires two properties:

- The supervisory signal must be easily extractable from the data itself without too much processing. This is to ensure that the supervisory signal effectively exposes information present in the data, and is not an artefact of the computations required to create it.

- The pretext task that the network is initially trained on must be related to the final task upon which the performance of the network must be measured. This is to ensure that the knowledge learnt to perform one task effectively transfers over to the final task.

A number of pretext tasks are available in the existing literature. Colored images are widely available, and converting them to grayscale is a simple process. Networks which accurately predict image coloration [47, 91] are effective at image segmentation tasks. In order to predict image color from grayscale images, such networks effectively learn to distill object information such as shape, size, and boundaries. This ability transfers to image segmentation tasks as well. Other tasks, such as predicting image patch positions [18,55], and predicting image rotations [26], are used in the literature. Combining self-supervision with supervised approaches [13,24,70] has also resulted in improved performance over few-shot tasks.

Self supervised approaches can fall into various categories [34]. Generative models learn representations by trying to generate artificial examples which are similar to the existing data samples. Generative Adversarial Networks (GANs) [27] have inspired a number of works due to their success, such as CycleGAN [95], StyleGAN [39], and PixelRNN [79]. However, these networks are difficult to train and are often compute heavy. A more recent class of relevant self-supervised approaches known as "Contrastive Learning" learns by comparing images against each other.

2.3.1 Contrastive Learning

Contrastive learning methods learn by comparing pairs of samples from the dataset. The goal is to group similar samples close together, and dissimilar samples far away from each other in the embedding space. This requires a notion of similarity between the input data. Formally, for a given sample input I, a sample from the same distribution is denoted a positive sample I^+, and a sample from a different distribution is denoted a negative sample I^-. Given a similarity function $d(\cdot, \cdot)$, the network must learn an embedding function $f(\cdot)$ on the input samples such that it can simultaneously

- Maximize the distance between input and negative sample: $d(I, I^-)$
- Minimize the distance between input and positive sample: $d(I, I^+)$

See Figure 2.2 for a graphical summary.

There are various approaches to generating the positive and negative sample pairs in the literature [14, 30, 35, 75]. For our experiments, we demonstrate our new framework using two recent contrastive learning approaches: SimCLR [14] and MoCo [30]. These approaches first perturb the input image using simple transforms before embedding it in the latent space and calculating similarity scores. These transformed images serve as inputs to the pipeline. Two inputs are said to belong to the same distribution if they are transformations of the same initial image, otherwise, they are said to belong to different distributions. This allows us to generate positive and negative samples without using class labels at all. By learning to predict image similarity in the presence of distorting transforms, the network is able to effectively distill information, making it suitable for quick adaptation to novel classes. This quality makes it suitable for use in our label-free few-shot framework.

2.4 Novelty

In this thesis, we ask the questions: Are labels really required for few-shot classification tasks? Are existing few-shot architectures making effective use of labels?

Existing few-shot classification methods enjoy high performance but also use a lot of labels during the training phase. We investigate how much of the final accuracy can be achieved by using no labels at all, thereby gauging how effectively these networks are extracting information from labels.

To examine the effect of label information on classification accuracy we experiment with different architectures, extremely simple classifiers, and don't fine tune on labels at all.

To the best of our knowledge, the label-free few-shot framework we propose is the first attempt to evaluate few-shot systems in the complete absence of labels. While other related works do use self supervised learning and contrastive learning methods, we differ in that we investigate the efficacy of these methods in the context of few-shot learning while achieving accuracy close to that of the state-of-the-art label using systems.

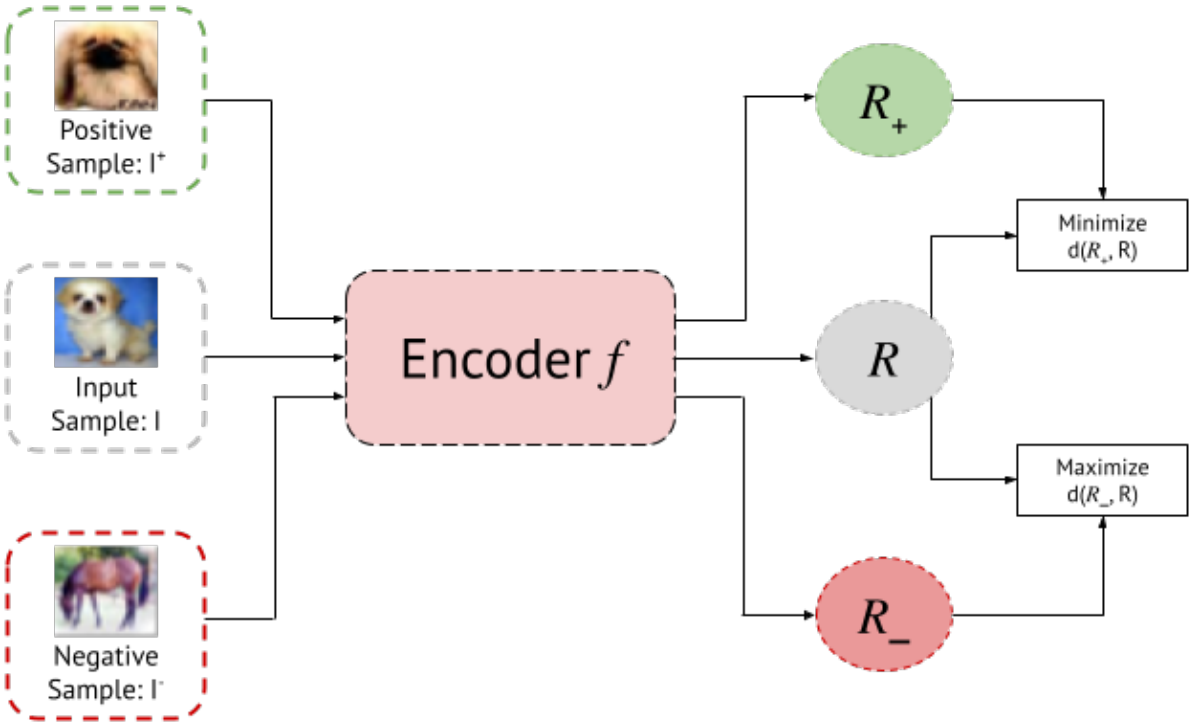

Figure 2.2: Contrastive Learning: A simple overview. For a given input image, positive and negative samples are chosen. Each input is separately embedded in the latent space before calculating similarity scores. Samples drawn from the same distribution should lie close to each other in the embedding space while samples from different distributions should lie further apart. *(Best viewed in color)*

Chapter 3

Label Free Framework

In this chapter, we provide an in-depth overview of our label-free framework and introduce the specific settings used in our experiments. The figure 3.1 graphically presents an overview of the framework. During the training phase, our network learns contrastive representations, which is achieved by the pretext task of predicting similarity between distorted images. The presence of image distortions forces our models to learn general image representations, enabling them to generalize well to novel classes that have few examples. During the testing phase, we utilize the learned representations to embed images and rely on similarity in the embedded space to perform a few-shot classification. Our framework ensures that the test-time classifier does not fine-tune itself using labels and has no label information whatsoever. Instead, labels are only utilized to calculate the final classifier accuracy.

In addition, we conduct ablation studies in Chapter 5 to investigate the effects of introducing limited label information into our pipeline. By doing so, we can further evaluate the overall effectiveness of our label-free approach and determine its optimal implementation. Overall, our framework offers a robust and efficient solution for performing few-shot classification tasks in the complete absence of labels, while also enabling our models to learn and generalize from a wide range of image representations.

In this study, we adopt a simple two-phase approach based on recent work [14, 30, 81]. During the training phase, we train the base network using self-supervised approaches on the training classes. The goal is to develop a robust and efficient network that can effectively generalize to novel classes during the testing phase. In the testing phase, we use the trained network in extremely simple classifiers for the few-shot tasks. Notably, we ensure that the network has no access to label information at any point, which aligns with our label-free setting.

Although we have access to K labeled examples per C classes, the network does not access the labels during testing. Instead, we rely on the effectiveness of our training scheme and the simplicity of our classifiers to achieve optimal results. There are two main reasons for using only very simple classifiers during the testing phase. Firstly, recent literature [14,15,69,80] has demonstrated the competitiveness of these simple classifiers, and secondly, simple classifiers require no labels during test time. This allows us to focus solely on the effectiveness of our training scheme while maintaining a label-free setting. Overall, our two-phase approach offers a simple yet effective solution for few-shot learning tasks.

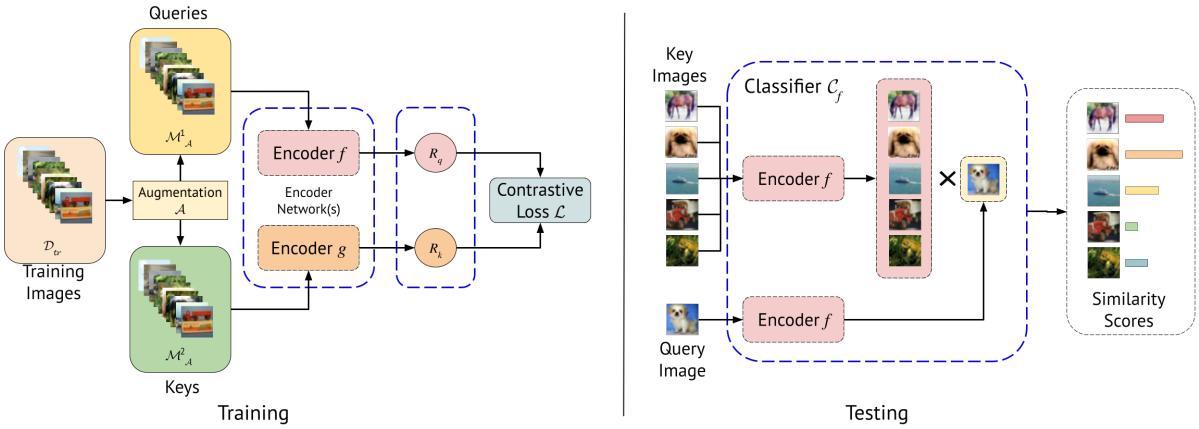

Figure 3.1: Graphical overview of our pipeline: **Left**: Our training method is designed to learn contrastive representations without the use of labels. Specifically, we use a self-supervised approach in which a single input minibatch is augmented using two simple image transforms to generate key and query image batches. The encoder networks then learn to preserve image similarity between keys and queries generated from the same input. This is accomplished by minimizing the contrastive loss function during training. For further details on our training algorithm, please refer to Algorithm 1. **Right**: During the testing phase, we use the network from the training phase to encode images and perform few-shot classification without using labels. Our approach is based on image similarity, and we employ a label-free classifier that predicts the most similar key image for every query image. This allows us to classify images at test time without labels, which is consistent with our label-free setting. For further details on our testing algorithm, please refer to Algorithm 2.

3.1 Label Free Training Framework

Few-shot learning is a challenging problem that has garnered significant interest in recent years. One effective approach to few-shot learning is to focus on learning image similarity, as demonstrated by [41]. However, unlike [41], we aim to completely ignore labels during our training phase. To accomplish this, we utilize a contrastive learning approach that focuses on learning contrastive representations.

In our contrastive learning approach, we start by applying two different data augmentations to an input image, generating two augmented images. We then train our neural network $f(\cdot)$ to learn image representations such that the encoding of two augmented images generated from the same input is as similar as possible. This allows us to learn robust image representations without relying on any label information. A detailed description of one training epoch is presented in Algorithm 1, and a visual overview of our approach can be found in Figure 3.1.

Overall, our self-supervised contrastive learning approach is quite effective for the few-shot classification task, allowing us to ignore labels completely and focus solely on learning robust image representations. This is a significant advantage over traditional few-shot learning approaches that rely on labeled data, as our approach is more flexible and can be applied to a wider range of domains.

Given an input minibatch $\mathcal{M}$, a stochastic augmentation module $\mathcal{A}$ generates two minibatches, one of the query images $\mathcal{M}^q_{\mathcal{A}}$, and the other of the key images $\mathcal{M}^k_{\mathcal{A}}$ by performing two different image transforms. For a given query image, q the key image generated from the same input is denoted k_+, and k_- otherwise. Pairs of query and key generated from the same input (q, k_+) are denoted positive and negative (q, k_-) otherwise.

Encoder networks $f(\cdot)$ and $g(\cdot)$ are used to learn representations of key and query images, respectively. Note that depending on the setting, these networks may be the same. Network $f(\cdot)$ is used for downstream test time tasks. After computing the encoded representations R_k of the key, and R_q of the query, the networks are trained to maximize the representation similarity for positive pairs, and minimize for negative pairs. This is achieved by minimizing the following contrastive loss in Equation 3.1, where τ is a temperature hyperparameter, and $s(\cdot, \cdot)$ is a similarity function.

$$\mathcal{L}(R_q, R_{k^+}, \{R_{k^-}\}) = -\log \frac{\exp s(R_q, R_{k^+})/\tau}{\exp s(R_q, R_{k^+})/\tau + \sum\limits_{R_{k^-}} \exp s(R_q, R_{k^-})/\tau} \tag{3.1}$$

In our experiments, we use two recent works SimCLR [14] and MoCo [30], which fit into our overall training framework above. In practive, both these approaches differ in how they generate positive and negative sample pairs. We now present the specific details and training settings.

3.1.1 SimCLR Base

This training setting operates on minibatches of images rather than the entire dataset. From each input minibatch of N images, two minibatches of key and query images are generated using stochastic data augmentation. After applying image transforms, augmented minibatches of key and query images

Algorithm 1 Overall Training Methodology

Input: Augmentation Module $\mathcal{A}(\cdot)$
Input: Encoders $f(\cdot)$ $g(\cdot)$
Input: Constrastive Loss Module $\mathcal{L}(q, k^+, \{k^-\})$
Data: Training dataset $\mathcal{D}_{tr}$
Result: Trained network $f(\cdot)$

for *minibatch* $\mathcal{M}$ *in* $\mathcal{D}_{tr}$ **do**
  $\mathcal{M}^q_{\mathcal{A}}, \mathcal{M}^k_{\mathcal{A}} = \mathcal{A}(\mathcal{M})$; `// get augmented minibatches`
  $\{R_q\} = f(\mathcal{M}^q_{\mathcal{A}})$; `// encode query representations`
  $\{R_k\} = g(\mathcal{M}^k_{\mathcal{A}})$; `// encode key representations`
  for *query* R_q *in* $\{R_q\}$ **do**
    R_{k^+} = `ChoosePositive(`R_q, $\{R_k\}$`)` ; `// positive key image`
    $\{R_{k^-}\}$ = `ChooseNegative(`R_q, $\{R_k\}$`)` ; `// negative key images`
    $\mathcal{L}(R_q, R_{k^+}, \{R_{k^-}\})$; `// minimize contrastive loss`
    `UpdateParams(`f, g`)` ; `// update network parameters`
  end
end
return f

are treated on an equal footing with no distinction. Each query image has a corresponding positive key image, generated from the same input, and $2N - 2$ negative key images, generated from different inputs. Taking image order into account, this leads to a total of $2N$ positive pairs and $2N(2N - 2)$ negative pairs. The same neural network, $f(\cdot)$, is used to encode both the key and query images. During training, the network is optimized to maximize the similarity between the embeddings of positive image pairs and minimize the similarity between negative pairs using the contrastive loss in Equation 3.1. The cosine similarity function $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}/|\mathbf{x}||\mathbf{y}|$, is used to calculate the similarity between image embeddings. This setup is referred to as OURS˙S in the results.

3.1.2 MoCo Base

In this setting, the few-shot task is formulated as a dictionary lookup problem, decoupling the number of negative samples from the batch size. The dictionary consists of *key* images and the unknown image to be looked up is the *query*. The goal of the encoder network is to map the query image to a vector that is close to the vector of the correct class in the dictionary. This is done by minimizing a contrastive loss function that encourages the encoded query vector to be close to the encoded vector of the correct class, while pushing it away from the encoded vectors of the other classes.

In contrast to the previous setting, we use two different encoders for this task: a momentum encoder and a non-momentum encoder. The momentum encoder is used to encode the key images and maintain a set of positive and negative samples for each query, while the non-momentum encoder is used to encode the query images from the dictionary. The momentum encoder is updated using a momentum-based

weight update rule, which allows it to maintain a more stable set of positive and negative samples over time.

The query (non-momentum) encoder is used for downstream few-shot tasks. This setting uses a dot product as the similarity function for contrastive loss $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$ and is referred to as OURS_M in our results.

3.2 Label-Free Inference

Following standard literature [81], our test phase consists of multiple C-way K-shot tasks, where the model has to learn to classify new classes with very limited labeled data. In each task, the model is presented with C classes that it has not seen during training, and it is given only K labeled examples (key images) for each class. The model must then use these examples to classify 15 query images (unlabeled) for each of the C classes.

The use of class labels in task generation is only for ensuring that the task is a valid few-shot learning scenario, and does not provide any additional information to the model during testing. In fact, the classifier used during testing, $\mathcal{C}_f$, does not have access to class labels and can only match each query image q to its corresponding key image k based on its learned representation.

The matching process is based on the similarity between the query and key image representations learned by the network $f(\cdot)$. The classifier $\mathcal{C}_f$ returns the index j of the key image that is most similar to the query image q in terms of their learned representations. This is a form of nearest neighbor classification, where the label of the query image is inferred from the label of its nearest neighbor in the key set. This approach is often used in few-shot learning scenarios where the labeled data is limited, and it has been shown to be effective in a variety of tasks. More importantly, this allows us to maintain our label free setting. No parameter updates were made to the classifiers using the key or query images. See Algorithm 2 for details and Figure 3.1 for a visual overview.

3.2.1 Label Free Classifiers

The inference framework in our experiments is based on using simple classifiers that are both computationally efficient and label-free. We draw inspiration from the work of [15, 81] and use the 1-Nearest Neighbour classifier (1NN) from SimpleShot [81], as well as a simplified version of the soft cosine attention kernel (ATTN) from Matching Networks [80].

Our choice of these simple classifiers allows us to focus on the effectiveness of our training methods while avoiding the need for labelled data during testing. This is especially important in few-shot learning scenarios where labelled data may be scarce or non-existent.

The algorithm for the test phase is presented in Algorithm 2.

The 1NN classifier is a simple Nearest Neighbour classifier. It simple chooses the key image which minimizes the Euclidean distance in the embedding space between the key and the query image under

Algorithm 2 Test Phase: N-way, K-shot Task

Input: Trained Encoder f
Input: Classifier $\mathcal{C}_f$
Input: Similarity Function $s(\mathbf{x}, \mathbf{y})$
Data: $N \times Q$ query images: $\{(q_i, y_{q_i})\}$
Data: $N \times K$ test images: $\{(k_i, y_{k_i})\}$
Result: Accuracy on task
correct $\leftarrow 0$ **foreach** *query image* q_i **do**
    `// return index of most similar key since classifier has no label access`
    $l = \mathcal{C}_f(q_i, \{k_j\})$ **if** $y_{q_i} == y_{k_l}$ **then** correct = correct $+ 1$;
end
return correct$/(N \times Q)$

consideration.

$$\mathcal{C}_f(q, \{k\}) = \arg\min_j |f(q) - f(k_j)|^2 \tag{3.2}$$

The ATTN classifier used in the experiments described in the paper uses an attention mechanism to choose the key image corresponding to each query. Specifically, the attention mechanism calculates a softmax over the cosine similarity between the query and each key image. However, unlike the original Matching Networks algorithm [80], which uses a weighted average over the labels of the key image set, the ATTN classifier simply takes the $\arg\max$ of the attention weights to identify the most similar key image. This is because our classifier has no access to the probability distribution over the labels or the number of labels, and we want to keep the inference framework label-free. More details on the inference process can be found in Algorithm 2.

$$\begin{aligned} \mathcal{C}_f(q, \{k\}) &= \arg\max_j a_{\{k\}}(q, k_j) \\ a_{\{k\}}(q, k_j) &= \frac{\exp c(f(q), f(k_j))}{\sum_i \exp c(f(q), f(k_i))} \\ c(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}| \cdot |\mathbf{y}|} \end{aligned} \tag{3.3}$$

Chapter 4

Label Free Experiments

We test various settings of our label-free framework using standard Few Shot Image Classification benchmarks and evaluation protocols. This chapter presents details of our experiments and our results.

We test various settigns of our label-free framwork using standard Few-Shot Image Classification benchmarks and evaluation protocols. These benchmarks evaluate few-shot learning algorithms on three different datasets. This chapter presents details of our experiments and our results.

4.1 Experimental Setup

4.1.1 Datasets and Evaluation Protocol

We empirically evaluate the effectiveness of our label-free pipeline on three common few-shot image classification benchmark datasets: **miniImageNet** [80], **CIFAR-100FS** [56], and **FC100** [56] (FewShot CIFAR100).

The miniImageNet dataset [80] is a subset of ImageNet [17] containing 100 classes and 600 examples per class. Following [59], we split the dataset to have 64 base classes, 16 validation classes, and 20 novel classes. Following [80], we resize the images to 84 $\times$ 84 pixels via rescaling and center cropping. The CIFAR-100FS dataset is a subset of CIFAR-100 [43], containing 100 image classes, with each class having 600 32$\times$32 images. Following the setup proposed in [56], we split the classes into 60 base, 20 validation, and 20 novel classes for few-shot learning. The FC100 dataset is also a subset of CIFAR-100. The 100 classes of the CIFAR-100 [43] dataset are grouped into 20 superclasses to minimize information overlap between the superclasses. The train split contains 60 classes belonging to 12 superclasses, and the validation and test splits contain 20 classes belonging to 5 superclasses each, following the setup in [56].

Following established literature in the field, we follow a standard evaluation protocol [61,81]. At test time, the classifier is presented with 10,000 tasks to calculate average accuracies and 95% confidence intervals. Given the set of C novel classes, we generate an N-way K-shot task as follows. N classes are uniformly sampled from the set of C classes without replacement. From each class, K key and

$Q = 15$ query images are uniformly sampled without replacement. The classifier is presented with the key images and then used to classify the query images. Following prior work [81], we focus on 5-way 1-shot and 5-way 5-shot benchmarks. To highlight the efficacy of self-supervised label-free training and keep our testing methods label-free, we use extremely simple non-parametric classifiers (1NN and ATTN) over the base networks.

4.1.2 Models and Implementation Details

All experiments use a ResNet-50 [31] backbone. SimCLR [14] pre-training is done for 500 epochs, a learning rate of 0.1, nesterov momentum of 0.9, and weight decay of 0.0001 on the respective datasets. For the training phase, distorting transforms of RandomResizedCrop and ColorDistortion were found to achieve the best results. The augmentations use the default hyperparameters from the published work [14]. MoCo [30] pre-training is done for 800 epochs over the entire ImageNet [17] dataset using the default parameters and details mentioned in their paper. Downstream tasks use the query (non-momentum) encoder network. All code is published on `https://github.com/adbugger/FewShot`.

4.2 Results

This work aims to examine the performance of our label-free pipeline for Few-Shot Learning and critique the performance benefits of traditional label-intensive frameworks. To that end, in addition to simply comparing classification accuracy we categorize frameworks into supervised, semi-supervised using labels, and semi-supervised without using labels. Further, a comparison of the number of labels used is also presented. The number of labels used by the pipelines is compared using the following strategy: if the network trains by performing gradient updates over the training labels, we count the labels in the training set; if the network fine-tunes over the test labels or uses test labels to compute class representations, we count the labels in the test set; if the network uses training and validation data to report results, we count training and validation labels. Unless otherwise specified in the published works, we assume that the validation set has not been used to publish results, and that the train and test pipelines are the same.

4.2.1 Empirical Analysis

Tables 4.1, 4.2 and 4.3 present our results on the *mini*ImageNet, CIFAR100FS and FC100 datasets respectively. For a more comprehensive comparison, we also adapt the work presented in Wu *et al.* [84] to include another unsupervised method in these results. Accuracies are averaged over 10,000 tasks and reported with 95% confidence intervals. The number of labels used by the methods is computed as follows: if the network trains by performing gradient updates over the training labels, we count the labels in the training set; if the network fine-tunes over the test labels or uses test labels to compute

Table 4.1: Average accuracy (in %) on the miniImageNet dataset. [1]Results from [3], which did not report confidence intervals. [2]AmDimNet [13] used extra data from the ImageNet dataset for training the network used to report mini-Imagenet numbers. [3]Results from our experiments adapting the published training code from [84]. [4]Results on other datasets not available. OURS was implemented here using OURS_S pipeline and ATTN classifier.

	Approach	Setting		Labels Used
		1-shot	5-shot	
Fully Supervised	MAML [20]	49.6 ± 0.9	65.7 ± 0.7	50,400
	CloserLook [15]	51.8 ± 0.7	75.6 ± 0.6	50,400
	RelationNet [73]	52.4 ± 0.8	69.8 ± 0.6	50,400
	MatchingNet [80]	52.9 ± 0.8	68.8 ± 0.6	50,400
	ProtoNet [69]	54.1 ± 0.8	73.6 ± 0.6	50,400
	Gidaris *et al.* [25]	55.4 ± 0.8	70.1 ± 0.6	50,400
	TADAM [56]	58.5 ± 0.3	76.7 ± 0.3	50,400
	SimpleShot [81]	62.8 ± 0.2	80.0 ± 0.1	38,400
	Tian *et al.* [76]	64.8 ± 0.6	82.1 ± 0.4	50,400
	S2M2 [50]	$\mathbf{64.9 \pm 0.2}$	$\mathbf{83.2 \pm 0.1}$	50,400
	Gidaris *et al.* [24]	63.77 ± 0.45	80.70 ± 0.33	50,400
Semi Supervised With Finetuning	Antoniou *et al.* [3][1]	33.30	49.18	21,600
	AmDimNet [13][2]	$\mathbf{77.09 \pm 0.21}$	$\mathbf{89.18 \pm 0.13}$	21,600
Semi Supervised And Label Free	Wu *et al.* [84][3]	32.4 ± 0.1	39.7 ± 0.1	0
	BoWNet [23][4]	**51.8**	**70.7**	0
	Ours	50.1 ± 0.2	60.1 ± 0.2	0

Table 4.2: Average accuracy (in %) on the CIFAR100FS dataset. [1]Results from [48]. [2]Results from our experiments adapting the published training code from [84]. OURS was implemented here using OURS_S pipeline and ATTN classifier.

	Approach	Setting		Labels Used
		1-shot	5-shot	
Fully Supervised	MAML [20][1]	58.9 ± 1.9	71.5 ± 1.0	48,000
	RelationNet [73][1]	55.0 ± 1.0	69.3 ± 0.8	48,000
	ProtoNet [69][1]	55.5 ± 0.7	72.0 ± 0.6	48,000
	R2D2 [8][1]	65.3 ± 0.2	79.4 ± 0.1	48,000
	MetaOptNet [48]	72.8 ± 0.7	85.0 ± 0.5	60,000
	Tian *et al.* [76]	73.9 ± 0.8	86.9 ± 0.5	48,000
	S2M2 [50]	$\mathbf{74.8 \pm 0.2}$	$\mathbf{87.5 \pm 0.1}$	48,000
	Gidaris *et al.* [24]	73.62 ± 0.31	86.05 ± 0.22	48,000
Semi Supervised And Label Free	Wu *et al.* [84][2]	27.1 ± 0.1	31.3 ± 0.1	0
	Ours	$\mathbf{53.0 \pm 0.2}$	$\mathbf{62.5 \pm 0.2}$	0

Table 4.3: Avg accuracy (in %) on FC100 dataset. [1]Results from [48]. [2]Results from our experiments adapting published training code from [84]. OURS was implemented here using OURS_S pipeline and ATTN classifier.

	Approach	Setting		Labels Used
		1-shot	5-shot	
Fully Supervised	ProtoNet [69][1]	35.3 ± 0.6	48.6 ± 0.6	48,000
	TADAM [56][1]	40.1 ± 0.4	56.1 ± 0.4	48,000
	MTL [71]	45.1 ± 1.8	57.6 ± 0.9	60,000
	MetaOptNet [48]	$\mathbf{47.2 \pm 0.6}$	$\mathbf{62.5 \pm 0.6}$	60,000
	Tian *et al.* [76]	44.6 ± 0.7	60.9 ± 0.6	48,000
Semi Supervised And Label Free	Wu *et al.* [84][2]	27.4 ± 0.1	32.4 ± 0.1	0
	Ours	$\mathbf{37.1 \pm 0.2}$	$\mathbf{43.4 \pm 0.2}$	0

class representations, we count the labels in the test set; if the network uses training and validation data to report results, we count training and validation labels. Unless otherwise specified in the respective works, we assume that the validation set is not used to publish results, and that the train and test pipelines are the same.

Our method achieves strong baselines on the benchmarks while using extremely limited label information, as can be seen in the comparison with Wu *et al.* [84], which operates in the same setting. These are the only two methods across all benchmark datasets that use **almost no** label information. Other methods are provided for comparison and the label count is calculated accordingly. The supervised methods use tens of thousands of labels, which can be very expensive depending on a particular domain. Our methodology seeks to provide a pathway to solving problems in such settings with no annotation cost whatsoever.

BowNet [23] operates in a similar setting and performs well on the mini-Imagenet benchmark. There are similarities to our pipeline but we believe that their 2-phase training approach can be seen as a more information-rich version of our pipeline and explains the higher performance on 5-shot tasks. Phase 1 uses rotation [26] as a pre-training task to train network A. This network A is used to perform k-means clustering in the representation space. Their hypothesis is that these clusters represent high level features. In Phase 2, the k-means cluster assignments generated using network A are used as supervisory signal to train network B to predict the cluster assignment probability in the presence of image perturbations. This task of predicting a probability distribution over the cluster assignments is more information rich than our task of detecting whether two perturbed images are coming from the same input. The supervisory signal is a probability vector, instead of a binary decision. The explicit decision to predict high level features also helps few-shot classification accuracy.

A higher number of input images increases the classification accuracy, as seen in our 5-way-5-shot tasks. The best results are achieved over the challenging miniImageNet dataset, followed by CIFAR100FS and FC100 datasets. This is expected as FC100 is a coarse-grained classification task and is specifically constructed to have dissimilar classes.

Chapter 5

Further Inquiry

Our label-free framework is meant to be the first step towards machine learning systems that bridge the gap between human and machine intelligence. While the previous chapter established the viability of label-free few-shot, this chapter presents a qualitative analysis before exploring simple extensions to our methods.

5.1 Qualitative Analysis

We present a visual representation of the performance of our image classification pipeline through visual inspection of the learned representations. We demonstrate the generalizability of our model and how it is able to capture meaningful visual features that are crucial for accurate classification. Additionally, we analyze the impact of optimizing for a different metric (instead of image similarity) on the model's performance and provide insights into how they affect the learned representations. This qualitative analysis not only serves as a useful tool for understanding the inner workings of our model but also provides valuable insights for improving its performance in future work.

5.1.1 Classification Quality

We provide a qualitative analysis of our image classification results on the *mini*ImageNet dataset using our OURS˙S pipeline. Figure 5.1 displays a mosaic of several examples, showcasing both successful and failed classification instances. Our results are generally positive, with the network successfully detecting an hourglass in the foreground of an image, even when a person is present in the background. However, the network also made some mistakes, such as classifying a black and white polka-dotted dog (DALMATIAN) as a HUSKY, which is a fine-grained classification task and a challenging few-shot problem due to the close similarity between the two dog breeds. Nevertheless, our results demonstrate that our method can achieve reasonable performance with limited label information when the classes are coarse-grained and well-separated.

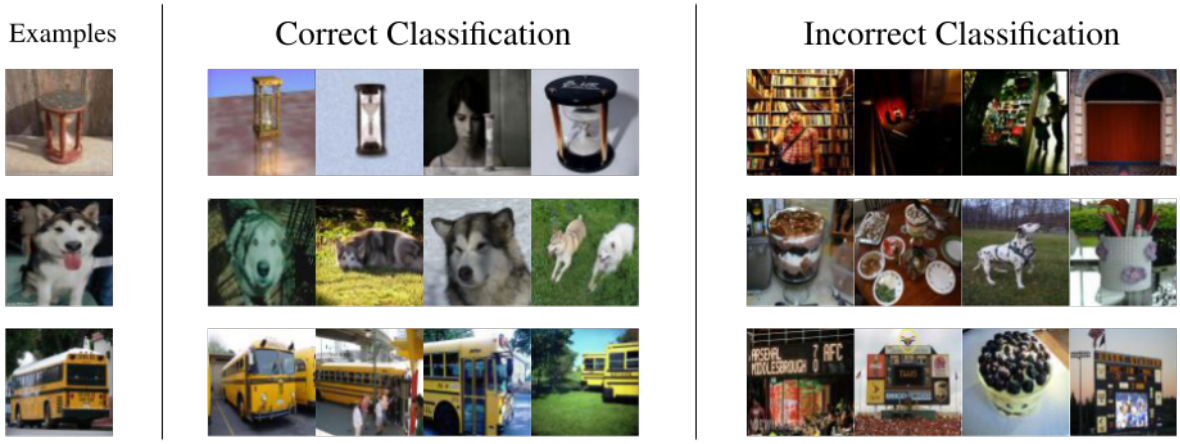

Figure 5.1: Visualizing a few examples from the miniImagenet test set using the OURS_S pipeline. **Far Left:** One labelled example visualized per class. **Middle:** Few correctly classified examples from the test set. **Right:** Mis-classified examples. Similarity in texture and coarse object category are contributing factors for mis-classification.

5.1.2 Clustering Quality

Our label-free pipeline is based on the idea of image similarity, which means that images that are similar should be represented by points that are closer together in a high-dimensional space. Conversely, dissimilar images should be represented by points that are farther apart from each other. This can be seen as a clustering problem, where similar images belong to the same cluster and dissimilar images belong to different clusters. To evaluate the effectiveness of our pipeline, we use a tSNE embedding technique to visualize the clusters formed by the representations of the miniImageNet test dataset that were learned by our MoCo trained network. The quality of the learned representations is reflected in the degree of separation between clusters in the tSNE embedding space. If the clusters are more distinct and well-separated, it indicates that the network has learned better representations. Hence, we believe that improving the semantic information captured by the learned representations in the final embedded space is an important direction for future research.

In Figure 5.2, we show a tSNE plot of the entire test dataset embedded using the OURS˙M trained network. Despite the dataset's complexity, we can observe evident cluster separation in the tSNE embedding. However, there is still room for improvement to further improve the cluster separation, which can be achieved by exploring ways to enhance the semantic information in learned representations in the final embedded space. One possible promising direction for future work is to use self-supervision to improve the cluster separation further. In fact, we have experimented with such a pipeline in Chapter 5. By improving cluster separation via self-supervision, we can obtain more pronounced clusters in the final embedded space, leading to better performance in image similarity-based classification.

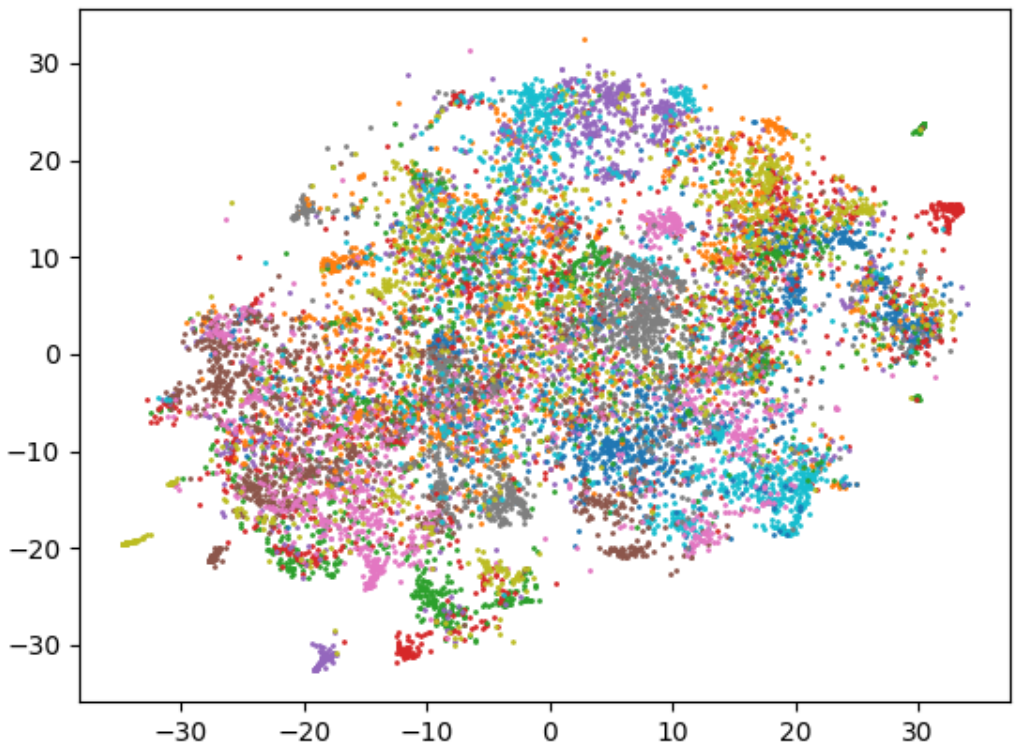

Figure 5.2: tSNE embedding of the miniImageNet dataset using our MoCo trained network. Considering the complexity of the dataset, the separation is evident. Improving the clustering quality should further improve test results, which is investigated in Section 5.2.1

Table 5.1: A comparison of multiple classifiers on the miniImagenet dataset. Average accuracy and 95% confidence intervals are reported over 10,000 rounds. The `_centroid` classifiers use class labels to compute the centroids per class. Best results per few-shot task are in **bold**.

Training	Testing	Setting	
		1-shot	5-shot
Ours_S	1NN	48.7 ± 0.2	59.0 ± 0.2
	Attn	50.1 ± 0.2	60.1 ± 0.2
	1NN_centroid	-	64.6 ± 0.2
	Attn_centroid	-	63.6 ± 0.2
Ours_M	1NN	44.2 ± 0.2	61.4 ± 0.2
	Attn	$\mathbf{59.8 \pm 0.2}$	73.1 ± 0.1
	1NN_centroid	-	69.3 ± 0.1
	Attn_centroid	-	$\mathbf{75.7 \pm 0.2}$
Ours_SF	1NN	46.8 ± 0.2	61.2 ± 0.1
	Attn	55.6 ± 0.2	68.2 ± 0.2
	1NN˙centroid	-	72.4 ± 0.1
	Attn_centroid	-	72.1 ± 0.1

5.2 Extending the Label-Free Framework : Directions for Future Work

5.2.1 Focusing on clustering

The core of our proposed image classification framework lies in the comparison of similarity scores of the input representations, which in turn relies on the minimization of the distance metric (described in Equations 3.2 and 3.3) for samples belonging to the same class, and maximization for samples from different classes. This distance minimization task is essentially equivalent to a clustering problem. Recognizing this similarity, we explore the idea of integrating clustering as a training step. This clustering step can be seamlessly incorporated into our framework, given its flexible and general nature. Moreover, since clustering is also a label-free approach, it aligns well with our overall label-free setting, making it a natural extension to our framework.

In our experiments, we study the utility of the SelfLabel [88] (OURS_SF in Tables 5.1, 5.2, and 5.3) method as a pre-training framework. This work simultaneously performs representation learning and image label assignment. Since the label assignment is constrained to form good clusters in the representation space, this work is a good candidate to test our hypothesis, while requiring no labels at all. This is a natural extension to our pipeline and fits in our label free few-shot setting, showcasing the adaptability of our proposed pipeline. As shown in Table 5.1, this improves performance on mini-ImageNet by a significant amount. However, OURS_SF approach does not yield benefits on other datasets. More complex clustering techniques and metrics can be investigated for future work.

Table 5.2: A comparison of multiple classifiers on the CIFAR100FS dataset. Average accuracy and 95% confidence intervals are reported over 10,000 rounds. The `_centroid` classifiers use class labels to compute the centroids per class. Best results per few-shot task are in **bold**.

Training	**Testing**	**Setting**	
		1-shot	**5-shot**
Ours_S	1NN	52.0 ± 0.2	61.7 ± 0.2
	Attn	$\mathbf{53.0 \pm 0.2}$	62.5 ± 0.2
	1NN_centroid	-	$\mathbf{65.8 \pm 0.2}$
	Attn_centroid	-	63.8 ± 0.2
Ours_M	1NN	32.2 ± 0.1	45.1 ± 0.2
	Attn	44.6 ± 0.2	56.7 ± 0.1
	1NN_centroid	-	52.8 ± 0.1
	Attn_centroid	-	59.5 ± 0.2
Ours_SF	1NN	42.5 ± 0.2	56.1 ± 0.2
	Attn	49.3 ± 0.2	61.6 ± 0.2
	1NN˙centroid	-	65.7 ± 0.2
	Attn_centroid	-	64.9 ± 0.2

Table 5.3: A comparison of multiple classifiers on the FC100 dataset. Average accuracy and 95% confidence intervals are reported over 10,000 rounds. The `_centroid` classifiers use class labels to compute the centroids per class. Best results per few-shot task are in **bold**.

Training	**Testing**	**Setting**	
		1-shot	**5-shot**
Ours_S	1NN	36.0 ± 0.2	42.6 ± 0.2
	Attn	37.1 ± 0.2	43.4 ± 0.2
	1NN_centroid	-	47.2 ± 0.2
	Attn_centroid	-	46.0 ± 0.1
Ours_M	1NN	32.0 ± 0.2	41.3 ± 0.2
	Attn	$\mathbf{38.3 \pm 0.2}$	47.4 ± 0.2
	1NN_centroid	-	47.7 ± 0.2
	Attn_centroid	-	$\mathbf{48.8 \pm 0.2}$
Ours_SF	1NN	30.9 ± 0.1	39.0 ± 0.2
	Attn	33.4 ± 0.2	41.3 ± 0.2
	1NN˙centroid	-	44.6 ± 0.2
	Attn_centroid	-	43.5 ± 0.2

Visualizing clustering quality per few-shot task

To investigate the impact of clustering on the final N-way K-shot task, we analyse tSNE plots of each task below (See Figures 5.4 and 5.3).

Each subplot consists of a visualization of one instance of a 5-way 5-shot task. Each task contains 5 key images and 5 query images for each of the 5 classes. The 100 inputs are then embedded using our learned network weights and then visualized.

From our experiments, the various instances fall into two broad categories. The visualizations in Figure 5.3 show a certain degree of inter-class separation, but also have intra-class separation. Classification accuracy in these cases would be improved by a better clustering which maximizes inter-class separation while simultaneously minimizing intra-class separation. This is a challenging setup in a label-free environment.

The visualizations in Figure 5.4 show a different story. All images are clustered into a small common center with a few far outliers. These cases show a complete failure of clustering. We hypothesize that these represent local minima of our loss functions, where the contrastive loss is maximized by having all inputs go to a common cluster center with some examples embedded a large distance away. These cases require a more thorough examination to show consistent improvement.

5.2.2 Dimensionality Reduction

At test time, we classify images using image similarity. Since the similarity is directly affected by the distance between representations in the embedded space, we also experiment with reducing the dimension of our learned representations using Principal Component Analysis (PCA). We perform this set of experiments using the OURS_SF pipeline since it trains by performing simultaneous clustering and labeling, hence this pipeline is most likely to be affected by the embedding dimension. Without PCA, the original dimension of our embeddings is 2048.

Once our backbone encoder network is trained, we use the learned weights to embed our entire training set and perform PCA. At test time, we use the same transformation matrix to reduce the dimension of our test set. We experiment with various output dimension sizes and present our results in Tables 5.4, 5.5, and 5.6. From the tables, we can see that the reducing the dimension using PCA does not affect the accuracies much. The quality of the learned representations is not affected by a simple linear transform. More complex dimensionality reduction techniques can be attempted as future work to study this further.

5.2.3 Introducing Limited Label Information at Test Time

A natural extension to our label-free framework is to incorporate the limited number of labels (1-5) from novel samples at test time to guide classification. We introduce CENTROID versions of our classifiers: 1 Nearest Neighbour Centroid (1NN_CENTROID), and Soft Cosine Attention Centroid (ATTN_CENTROID), in the multi-shot setting. First the encoder network converts each novel key image

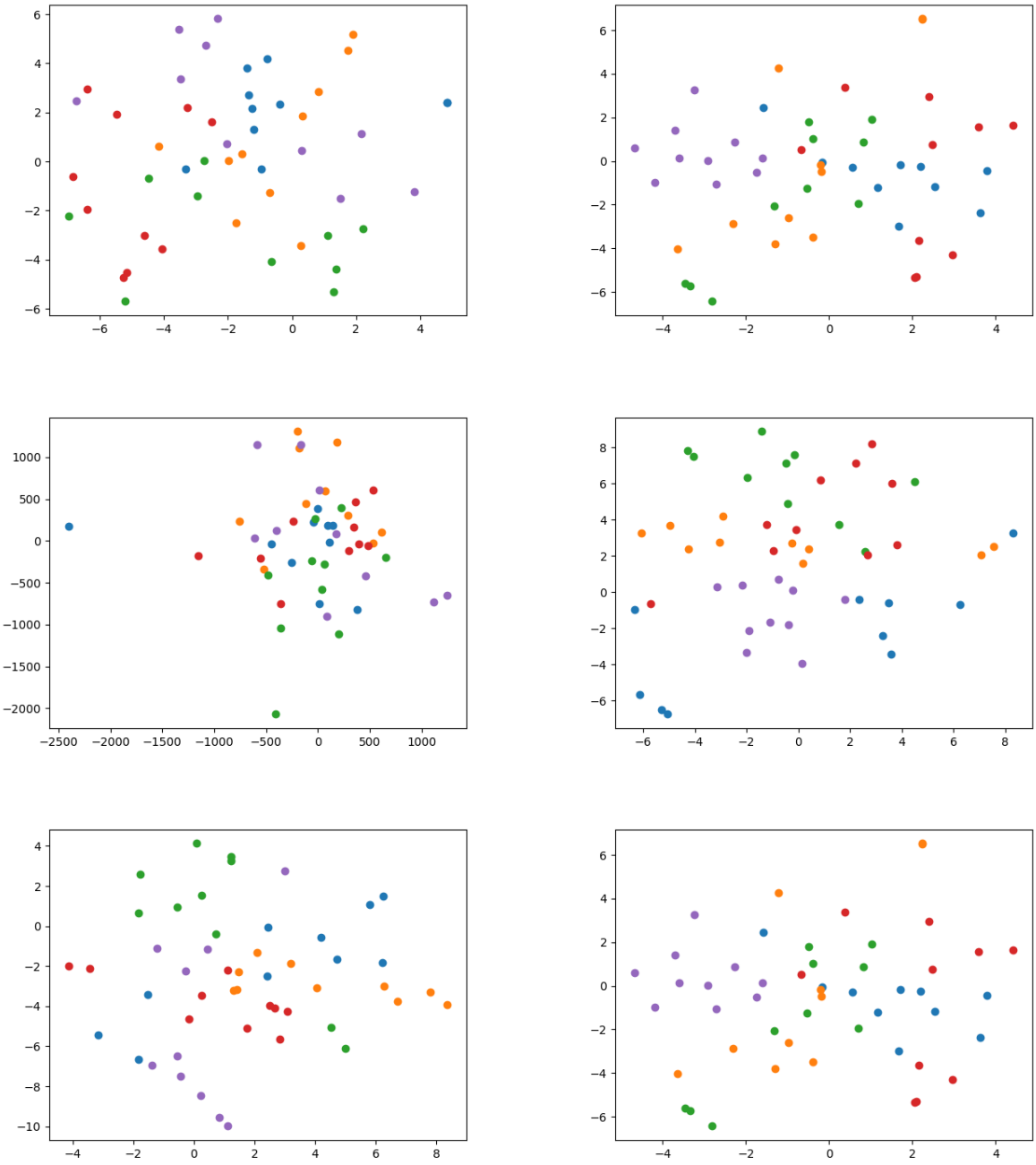

Figure 5.3: Visualizing the clustering quality for specific instances of few-shot tasks. Each color represents an input from the same class. Best viewed in color.

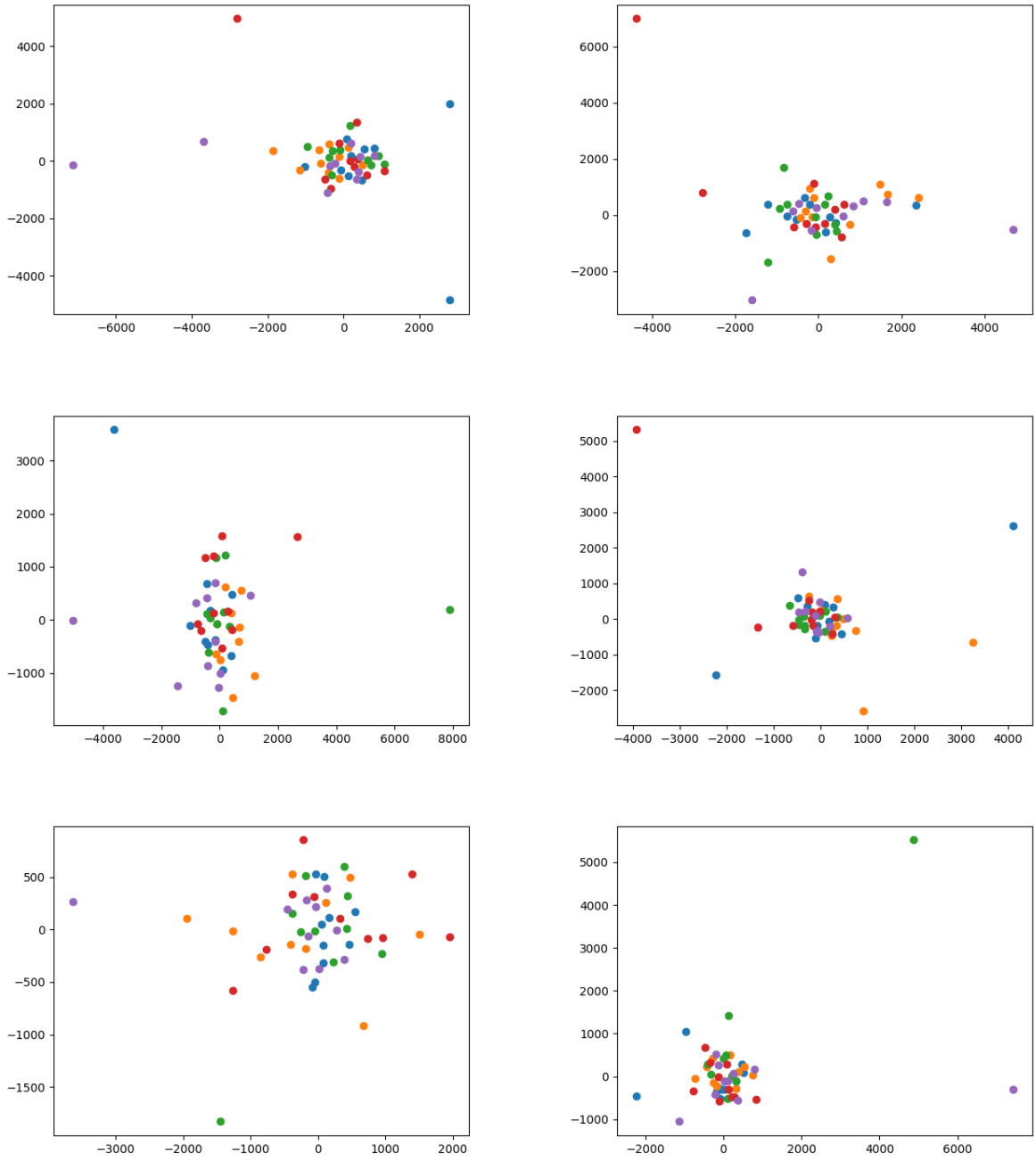

Figure 5.4: Visualizing the clustering quality for specific instances of few-shot tasks. Each color represents an input from the same class. Best viewed in color.

Table 5.4: Results of our experiments on the miniImageNet dataset with different embedding dimensions, using the Ours_SF pipeline. After learning network weights during training, we embed the training set and perform PCA to reduce the dimensionality of the representations. At test time we use the same learned transformation matrix to reduce the dimensionality of the test set. Best results for each dimension in **bold**. Accuracies averaged over 10,000 tasks and 95% confidence intervals are reported.

Output Dimensions	Classifier	miniImageNet	
		1-shot	5-shot
2048 (no PCA)	1NN	46.8 ± 0.2	61.2 ± 0.2
	Attn	$\mathbf{55.5 \pm 0.2}$	$\mathbf{68.2 \pm 0.2}$
1024	1NN	46.9 ± 0.2	61.2 ± 0.2
	Attn	$\mathbf{55.6 \pm 0.2}$	$\mathbf{68.5 \pm 0.2}$
512	1NN	46.7 ± 0.2	61.1 ± 0.2
	Attn	$\mathbf{55.5 \pm 0.2}$	$\mathbf{68.2 \pm 0.2}$
256	1NN	47.0 ± 0.2	61.2 ± 0.2
	Attn	$\mathbf{55.6 \pm 0.2}$	$\mathbf{68.3 \pm 0.2}$
128	1NN	47.9 ± 0.2	61.8 ± 0.2
	Attn	$\mathbf{55.2 \pm 0.2}$	$\mathbf{67.8 \pm 0.2}$

Table 5.5: Results of our experiments on the CIFAR100FS dataset with different embedding dimensions, using the Ours_SF pipeline. After learning network weights during training, we embed the training set and perform PCA to reduce the dimensionality of the representations. At test time we use the same learned transformation matrix to reduce the dimensionality of the test set. Best results for each dimension in **bold**. Accuracies averaged over 10,000 tasks and 95% confidence intervals are reported.

Output Dimensions	Classifier	CIFAR100FS	
		1-shot	5-shot
2048 (no PCA)	1NN	42.5 ± 0.2	56.1 ± 0.2
	Attn	$\mathbf{49.3 \pm 0.2}$	$\mathbf{61.6 \pm 0.2}$
1024	1NN	42.5 ± 0.2	55.89 ± 0.2
	Attn	$\mathbf{49.4 \pm 0.2}$	$\mathbf{61.6 \pm 0.2}$
512	1NN	42.6 ± 0.2	56.1 ± 0.2
	Attn	$\mathbf{49.5 \pm 0.2}$	$\mathbf{61.7 \pm 0.2}$
256	1NN	42.8 ± 0.2	56.2 ± 0.2
	Attn	$\mathbf{49.2 \pm 0.2}$	$\mathbf{61.5 \pm 0.2}$
128	1NN	43.2 ± 0.2	56.6 ± 0.2
	Attn	$\mathbf{49.1 \pm 0.2}$	$\mathbf{61.4 \pm 0.2}$

Table 5.6: Results of our experiments on the FC100 dataset with different embedding dimensions, using the Ours_SF pipeline. After learning network weights during training, we embed the training set and perform PCA to reduce the dimensionality of the representations. At test time we use the same learned transformation matrix to reduce the dimensionality of the test set. Best results for each dimension in **bold**. Accuracies averaged over 10,000 tasks and 95% confidence intervals are reported.

Output Dimensions	Classifier	FC100	
		1-shot	5-shot
2048 (no PCA)	1NN	30.9 ± 0.1	38.9 ± 0.2
	Attn	$\mathbf{33.4 \pm 0.2}$	$\mathbf{41.3 \pm 0.2}$
1024	1NN	31.0 ± 0.1	38.9 ± 0.2
	Attn	$\mathbf{33.3 \pm 0.2}$	$\mathbf{41.5 \pm 0.2}$
512	1NN	30.9 ± 0.1	38.9 ± 0.2
	Attn	$\mathbf{33.4 \pm 0.2}$	$\mathbf{41.5 \pm 0.2}$
256	1NN	31.1 ± 0.14	38.9 ± 0.2
	Attn	$\mathbf{33.4 \pm 0.2}$	$\mathbf{41.3 \pm 0.2}$
128	1NN	31.0 ± 0.2	38.9 ± 0.2
	Attn	$\mathbf{33.0 \pm 0.2}$	$\mathbf{41.0 \pm 0.2}$

to its representation. Following [69, 80, 81], the CENTROID versions of these classifiers then compute class representatives as the centroids of these representations. Few-shot classification is then done by comparing each query image against each class centroid, essentially treating the class representative (or exemplar) as the new key image for that class.

As shown in Tables 5.1, 5.2, and 5.3, the centroid versions of the classifiers increase accuracy by 5-10% in the multi-shot setting. Innovative methods of introducing label information without computational overhead present an interesting area for future work.

5.2.4 Data Transformation Techniques

The first step in our training pipeline is a data transformation step where we apply distortion transforms to input data. This step generates augmented minibatches for self-supervised contrastive learning. The goal of our pipeline is to maximize the similarity of learned representations for inputs generated from the same image and minimize it for inputs generated from different images. The goal is to learn representations which can be quickly generalized to new classes.

The choice of image transforms is a critical factor that can affect the quality of learned representations. Therefore, we extensively investigate the effect of the choice of data transformations on the final accuracy. Following existing literature [14] we analyze a wide range of transformations involving random cropping, random flipping, color jittering, and gaussian blurring, and evaluate their impact on the classification accuracy of our framework. Random cropping allows the network to learn translation invariance, while horizontal flipping enables learning of rotational invariance. Color jittering can help the network learn to be invariant to color changes in the input images, while rotation can improve the

network's ability to recognize objects in different orientations. However, different transforms may have varying degrees of impact on the final performance, and some may even negatively impact it. Hence, we conducted a series of experiments to determine the optimal combination of transforms for our specific task. This is crucial in ensuring the learned representations are of high quality, and our model can generalize well to new, unseen data.

Based on [14] we experiment with three candidate data augmentation transforms: RANDOMRESIZEDCROP, GAUSSIANBLUR, and COLORDISTORTION. We describe each of our data transformations before presenting pseudocode for replicability.

For the RANDOMRESIZEDCROP transform, we first choose a random crop of the input image, resize to the original size, and apply a random horizontal flip with probability 0.5

```
from torchvision.transforms import (
  Compose, RandomResizedCrop, RandomHorizontalFlip, Normalize )


def RandomResizedCrop(options):
  return Compose([
    RandomResizedCrop(size=options.image_size),
    RandomHorizontalFlip(0.5),
    Normalize(options.image_mean, options.image_std),
  ])
```

For the COLORDISTORTION transform, we first apply the color jitter transform over the 4 channels of the image (red-green-blue-alpha) using a jitter strength hyperparameter, with a probability of 0.8, followed by converting the image to grayscale with probability 0.2.

```
from torchvision.transforms import (
  Compose, RandomResizedCrop, RandomHorizontalFlip, Normalize )


def ColorDistortion(options):
  s = options.jitter_strength
  return Compose([
    RandomApply([
      ColorJitter(0.8*s, 0.8*s, 0.8*s, 0.2*s)
    ], p=0.8),
    RandomGrayscale(p=0.2),
```

```
    Normalize(options.image_mean, options.image_std),
  ])
```

For the GAUSSIANBLUR transform, we apply Gaussian Blur with a probability of 0.5. The blurring kernel is 10% of the input image size, with a standard deviation chosen randomly between 0.1 and 2.0 .

```
from torchvision.transforms import (
  Compose, RandomResizedCrop, RandomHorizontalFlip, Normalize )


def GaussianBlur(options):
  kernel_size = options.image_size / 10
  return Compose([
    RandomApply([
      GaussianSmoothing(
        channels=options.image_channels,
        kernel_size=kernel_size,
        sigma=random.uniform(0.1, 2.0),
        dim=2)
    ], p=0.5),
    Normalize(options.image_mean, options.image_std),
  ])
```

Since each input pipeline uses 2 image transforms, we experiment with all three pairs of transformation functions across all our datasets and test time classifiers and report our results in Tables 5.7, 5.8, 5.9. From these tables, it is clear that the best results are achieved by a combination of RANDOMRESIZED-CROP and COLORDISTORTION which outperform the other combinations by a $\sim$ 15-20% across each dataset and setting. Such a drastic improvement in accuracy based on the choice of data transform alone indicates that experimenting with carefully crafted data augmentation transforms is a viable direction for future work and improvements.

Table 5.7: Results of our experiments with different combinations of data transforms. **Crop** refers to RandomResizedCrop. **Blur** refers to GaussianBlur. **Distort** refers to ColorDistortion. Representations learned using the Ours_S pipeline on the respective datasets. Accuracies averaged over 10,000 tasks and 95% confidence intervals reported. Best results in each setting in **bold**.

Transforms	**Classifier**	**miniImageNet**	
		1-shot	**5-shot**
Crop+Blur	1NN	33.47 ± 0.16	41.92 ± 0.16
	Attn	34.79 ± 0.17	42.65 ± 0.16
Crop+Distort	1NN	48.65 ± 0.20	58.98 ± 0.18
	Attn	$\mathbf{50.13 \pm 0.21}$	$\mathbf{60.09 \pm 0.18}$
Distort+Blur	1NN	21.96 ± 0.09	24.68 ± 0.10
	Attn	22.40 ± 0.10	25.77 ± 0.10

Table 5.8: Results of our experiments with different combinations of data transforms. **Crop** refers to RandomResizedCrop. **Blur** refers to GaussianBlur. **Distort** refers to ColorDistortion. Representations learned using the Ours_S pipeline on the respective datasets. Accuracies averaged over 10,000 tasks and 95% confidence intervals reported. Best results in each setting in **bold**.

Transforms	**Classifier**	**CIFAR100FS**	
		1-shot	**5-shot**
Crop+Blur	1NN	36.17 ± 0.17	47.43 ± 0.18
	Attn	37.97 ± 0.18	47.79 ± 0.18
Crop+Distort	1NN	51.96 ± 0.24	61.68 ± 0.20
	Attn	$\mathbf{52.90 \pm 0.24}$	$\mathbf{62.46 \pm 0.21}$
Distort+Blur	1NN	24.90 ± 0.12	30.23 ± 0.15
	Attn	25.55 ± 0.13	31.10 ± 0.15

Table 5.9: Results of our experiments with different combinations of data transforms. **Crop** refers to RandomResizedCrop. **Blur** refers to GaussianBlur. **Distort** refers to ColorDistortion. Representations learned using the Ours_S pipeline on the respective datasets. Accuracies averaged over 10,000 tasks and 95% confidence intervals reported. Best results in each setting in **bold**.

Transforms	**Classifier**	**FC100**	
		1-shot	**5-shot**
Crop+Blur	1NN	31.91 ± 0.15	41.14 ± 0.16
	Attn	32.80 ± 0.16	41.30 ± 0.16
Crop+Distort	1NN	36.00 ± 0.18	42.54 ± 0.17
	Attn	$\mathbf{37.09 \pm 0.19}$	$\mathbf{43.39 \pm 0.18}$
Distort+Blur	1NN	22.08 ± 0.08	25.08 ± 0.10
	Attn	22.92 ± 0.10	26.93 ± 0.11

Chapter 6

Conclusions

Deep Neural Networks have applications in many vision-related domains, such as generic image-understanding to self-driving cars [9]. This success is driven, in part, by the wide availability of large amounts of data present in these domains. However, such networks require a huge amount of data and compute power and cannot be easily adapted to domain with limited data or label availability.

To address this problem and to bridge the gap between human and machine intelligence, the field of Few-Shot Learning aims to create learners that can generalize to novel classes using extremely limited data. Conventional approaches in this area shift the label requirement from the novel set to the training set. While such approaches are effective, it is not always possible to find high-quality labeled data which is also related to the final task at hand.

To that end, this thesis proposes a Few-Shot learning framework that uses no labels. This has been a more challenging setup compared to existing Few-Shot learners. Our proposed training and testing pipeline is computationally simple, which makes it easier to adapt and modify. By leveraging self-supervision for training and image similarity for testing, we achieve competitive performance while using zero training or testing labels. However, this is 10,000 times fewer labels than existing state-of-the-art.

In our ablations studies and extensions, we investigate the role of clustering quality in classification accuracy. Due to the conceptually simple nature of our framework, we were able to adapt our pipeline to use a clustering-oriented method, introduce limited label information at test time, and investigate the effect of dimensionality. We also present a qualitative examination of classification performance and present some failure cases. Finally, we show that by introducing limited label information at test time we can achieve a 5-10% increase in accuracy without significant computational cost.

This thesis serves as a first step toward developing Few-Shot learners which require no labeled data, increasing the applicability of deep neural networks and bridging the gap between human and machine intelligence at the same time.

Related Publications

- Aditya Bharti, Vineeth N. Balasubramanian, C.V. Jawahar. **Towards Label-Free Few-Shot Learning: How Far Can We Go?** *6th International Conference on Computer Vision and Image Processing, CVIP 2021. IIT Ropar, India.*

Bibliography

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016. 9

[2] A. Antoniou, H. Edwards, and A. Storkey. How to train your MAML. In *ICLR*, 2019. 9

[3] A. Antoniou and A. Storkey. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *ICML*, 2019. ix, 23

[4] H. S. Behl, A. G. Baydin, and P. H. Torr. Alpha maml: Adaptive model-agnostic meta-learning. *arXiv preprint arXiv:1905.07435*, 2019. 9

[5] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013. 10

[6] S. Benaim and L. Wolf. One-shot unsupervised cross domain translation. In *NeurIPS*, 2018. 8

[7] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei. On the optimization of a synaptic learning rule, 1997. 9

[8] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019. 4, 10, 23

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 38

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[11] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017. 9

[12] R. Caruana. Multitask learning. *Machine learning*, 1997. 8

[13] D. Chen, Y. Chen, Y. Li, F. Mao, Y. He, and H. Xue. Self-supervised learning for few-shot image classification, 2019. ix, 3, 12, 23

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. 2020. 3, 4, 13, 15, 17, 22, 34, 35

[15] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang, and J.-B. Huang. A closer look at few-shot classification. In *ICLR*, 2019. 15, 19, 23

[16] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes, 2021. 1

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 21, 22

[18] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 4, 12

[19] M. Douze, A. Szlam, B. Hariharan, and H. Jégou. Low-shot learning with large-scale diffusion. In *CVPR*, 2018. 7

[20] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 3, 9, 23

[21] C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018. 9

[22] H. Gao, Z. Shou, A. Zareian, H. Zhang, and S.-F. Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *NeurIPS*, 2018. 7

[23] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord. Learning representations by predicting bags of visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6928–6938, 2020. 23, 24

[24] S. Gidaris, A. Bursuc, N. Komodakis, P. P. Pérez, and M. Cord. Boosting few-shot visual learning with self-supervision. In *ICCV*, 2019. 3, 12, 23

[25] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018. 23

[26] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 4, 12, 24

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 12

[28] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018. 9

[29] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017. 7

[30] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 3, 4, 13, 15, 17, 22

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 22

[32] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pages 84–92. Springer, 2015. 11

[33] G. V. Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *CoRR*, abs/1709.01450, 2017. 1

[34] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, Dec 2020. 12

[35] X. Ji, J. F. Henriques, and A. Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019. 3, 4, 13

[36] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM-MM*, 2014. 8

[37] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online, July 2020. Association for Computational Linguistics. 1

[38] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 1

[39] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 12

[40] D. Kim and B. Suh. Enhancing vaes for collaborative filtering. *Proceedings of the 13th ACM Conference on Recommender Systems*, Sep 2019. 1

[41] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML-W*, 2015. 3, 4, 10, 17

[42] J. Kozerawski and M. Turk. Clear: Cumulative learning for one-shot one-class image recognition. In *CVPR*, 2018. 9

[43] A. Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2009. 21

[44] A. Krizhevsky, I. Sutskever, and G. 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. 1, 2

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[46] R. Kwitt, S. Hegenbart, and M. Niethammer. One-shot learning of scene locations via feature trajectory transfer. In *CVPR*, 2016. 7

[47] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016. 4, 12

[48] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. ix, 1, 3, 10, 23, 24

[49] Y. Lee and S. Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018. 9

[50] P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, and V. N. Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *WACV*, 2020. 3, 10, 23

[51] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016. 8

[52] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018. 6

[53] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto. Few-shot adversarial domain adaptation. In *NeurIPS*, 2017. 8

[54] D. K. Naik and R. J. Mammone. Meta-neural networks that learn by learning. In *IJCNN*, 1992. 9

[55] M. Noroozi and P. Favaro. Unsupervised learning of visual represention by solving jigsaw puzzles. In *ECCV*, 2016. 4, 12

[56] B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*. 2018. 3, 21, 23, 24

[57] T. Pfister, J. Charles, and A. Zisserman. Domain-adaptive discriminative one-shot learning of gestures. In *ECCV*, 2014. 7

[58] T. Ramalho and M. Garnelo. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *ICLR*, 2018. 8

[59] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 3, 9, 21

[60] S. Reed, Y. Chen, T. Paine, A. van den Oord, S. A. Eslami, D. Rezende, O. Vinyals, and N. de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *ICLR*, 2018. 8

[61] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. 9, 21

[62] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In *ICML-W*, 2012. 8

[63] V. G. Satorras and J. B. Estrach. Few-shot learning with graph neural networks. In *ICLR*, 2018. 10

[64] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NeurIPS*, 2018. 7

[65] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. 1

[66] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. 1

[67] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE international conference on computer vision*, pages 118–126, 2015. 11

[68] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[69] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*. 2017. 3, 4, 8, 10, 15, 23, 24, 34

[70] J.-C. Su, S. Maji, and B. Hariharan. Boosting supervision with self-supervision for few-shot learning. *ArXiv*, abs/1906.07079, 2019. 12

[71] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019. 24

[72] W. Sun and Z. Chen. Learned image downscaling for upscaling using content adaptive resampler. *IEEE Transactions on Image Processing*, 29:4027–4040, 2020. 1

[73] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 3, 4, 10, 23

[74] S. Takase and S. Kiyono. Lessons on parameter sharing across layers in transformers, 2021. 1

[75] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 4, 13

[76] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? *ECCV*, 2020. 3, 23, 24

[77] Y.-H. H. Tsai and R. Salakhutdinov. Improving one-shot learning through fusing side information. *arXiv preprint arXiv:1710.08347*, 2017. 7

[78] A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space, 2021. 1

[79] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 12

[80] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NeurIPS*. 2016. 3, 4, 8, 10, 15, 19, 20, 21, 23, 34

[81] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning, 2019. 4, 15, 19, 21, 22, 23, 34

[82] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 2019. 4, 6

[83] K. Q. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18, 2005. 10

[84] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. ix, 22, 23, 24

[85] E. Xing, M. Jordan, S. J. Russell, and A. Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15, 2002. 10

[86] Z. Xu, L. Zhu, and Y. Yang. Few-shot object recognition from machine-labeled web images. In *CVPR*, 2017. 8

[87] W. Yan, J. Yap, and G. Mori. Multi-task transfer methods to improve one-shot learning for multimedia event detection. In *BMVC*, pages 37–1, 2015. 8

[88] A. YM., R. C., and V. A. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 28

[89] D. Yoo, H. Fan, V. N. Boddeti, and K. M. Kitani. Efficient k-shot learning with regularized deep networks. In *AAAI*, 2018. 9

[90] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018. 9

[91] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 4, 12

[92] Y. Zhang, H. Tang, and K. Jia. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *ECCV*, 2018. 8

[93] Y. Zhang and Q. Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017. 8

[94] W. Zheng, Z. Chen, J. Lu, and J. Zhou. Hardness-aware deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 72–81, 2019. 11

[95] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 12

[96] L. Zhu and Y. Yang. Compound memory networks for few-shot video classification. In *ECCV*, 2018. 8