Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

RINU BONEY

FAST ADAPTATION OF NEURAL NETWORKS

Master's Thesis
Espoo, 26.02.2018

Supervisor: Prof. Juho Kannala
Thesis Advisor: D.Sc. (Tech.) Alexander Ilin

Aalto University
School of Science

Abstract of the Master's Thesis:

Author:	Rinu Boney
Title of the thesis:	Fast Adaptation of Neural Networks
Date:	26.02.2018
Language:	English
Number of pages:	46
Major:	Machine Learning and Data Mining
Supervisor:	Prof. Juho Kannala
Thesis Advisor:	D.Sc. (Tech.) Alexander Ilin

The ability to learn quickly from a few samples is a vital element of intelligence. Humans can reuse past knowledge and learn incredibly quickly. Also, humans are able to interact with others to effectively guide their learning process. Computer vision systems for recognizing objects automatically from pixels are becoming commonplace in production systems. These modern computer vision systems use deep neural networks to automatically learn and recognize objects from data. Oftentimes, these deep neural networks used in production require a lot of data, take a long time to learn and forget old things when learning something new.
We build upon previous methods called Prototypical Networks and Model-Agnostic Meta-Learning (MAML) that enables machines to learn to recognize new objects with very little supervision from the user. We extend these methods to the semi-supervised few-shot learning scenario, where the few labeled samples are accompanied with (potentially many) unlabeled samples. Our proposed methods are able to learn better by also making use of the additional unlabeled samples. We note that in many real-world applications the adaptation performance can be significantly improved by requesting the few labels through user feedback (active adaptation). Further, our proposed methods can also adapt to new tasks without any labeled examples (unsupervised adaptation) when the new task has the same output space as the training tasks do.

Keywords: deep learning, few-shot learning, meta-learning, active learning

ACKNOWLEDGEMENTS

This master's thesis was done at The Curious AI Company. I would like to thank Curious AI for the resources and the inspiring environment. I would like to thank everybody at the company for their encouragement and support. I would like to thank Hotloo Xiranood for his enthusiastic company and Vikram Kamath for proofreading the thesis.

I am deeply grateful to my advisor Dr. Alexander Ilin for his invaluable guidance and support.

I would like to thank my supervisor Prof. Juho Kannala for his continuous support.

I would like to thank my parents and sister for their love and support.

Finally, I thank Silu for her encouragement and love.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

MAML Model Agnostic Meta Learning

PN Prototypical Networks

BPL Bayesian Program Learning

NCM Nearest Class Mean

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

SGD Stochastic Gradient Descent

GP Gaussian Processes

1 INTRODUCTION

We explore fast adaptation of deep neural networks to new tasks from a few samples in a short amount of time.

The ability to learn quickly from a few samples is a vital element of intelligence. Humans can reuse past knowledge and learn incredibly quickly. This thesis explores fast adaptation of deep neural networks to generalize quickly to new tasks from a few samples. Training such models involve learning general internal representations that are suitable for a wide range of tasks. In real-world problems, it is often the case that the problem at hand has few labeled data and more unlabeled samples and there usually exists datasets that are similar with plenty of labeled data and practically unlimited unlabeled data. This thesis explores the question – How to quickly adapt to this new task from a few samples in a short amount of (wall clock) time? In this thesis, we characterize a new task by a change in the environment, that is change in the input distribution (*domain adaptation*) or a change in the output space.

Consider the transfer learning scenario illustrated in Figure 1. We assume that we have a set of different tasks at any given time. Each task consists of labeled and unlabeled data and we want to produce a good model for each task. Knowledge gained from training models for previous tasks can be leveraged to improve performance in training models for new tasks, an ability known as *transfer learning*. Improved performance can be obtained as a side-effect of the advantage of additional data accumulated over time. The training algorithm itself can be modeled and improved over time with experience. This problem is known as *meta-learning*. Meta-learning can enable rapid generalization to new tasks. When the number of examples for a particular task is very low, the problem is called *few-shot learning*. The system should be able to generalize from the samples by leveraging knowledge extracted from previous tasks.

Under the assumption that the tasks are all sufficiently related, it is also possible to build a single model that is able to perform all the tasks. Ideally, the model should still be able to perform previously trained tasks and does not forget them over time. This ability is known as *continual learning*. Neural networks easily forget previous tasks when trained on new tasks, a problem referred to as *catastrophic forgetting* [23]. Similar to how humans learn, the complexity of tasks can be increased over time to enable easier learning. The learning system can slowly develop more complex models from simple ones. This scenario is called *curriculum learning*. Learning multiple tasks at the same time enables learning better shared representations that improve generalization. This approach is termed as *multi-task learning*.

Ideally, under certain assumptions on the similarity of the tasks, the training time and amount of data required should gradually decrease over time.

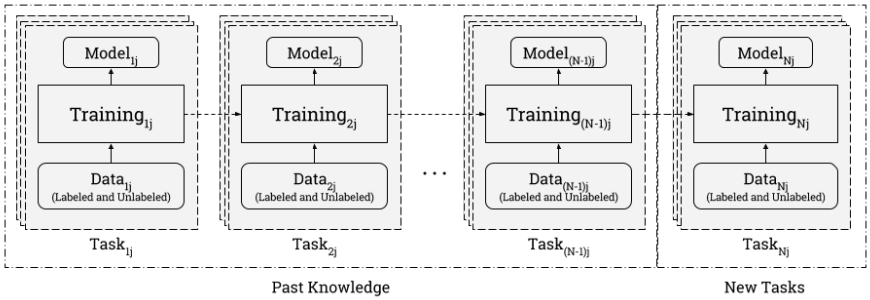


Figure 1: Illustration of different learning scenarios.

1.1 MOTIVATION

Adaptation is favorable in nonstationary environments

We present two different motivations for fast adaptation of neural networks:

- *Nonstationarity*: Fast learning and adaptation in a quickly changing environment are very important. An agent or system deployed in dynamic or interactive environments should be able to adapt to the changes in the environment.
- *Lack of data*: Oftentimes, real-world problems have few labeled samples and there usually exists data sources that are similar, with plenty of labeled data and practically unlimited unlabeled data. It is possible to adapt to these problems by leveraging the knowledge extracted from similar data sources.

In many real-world applications, it is possible to interact with the user. A lot of information is present in these interactions. In such interactive learning environments, the interactions can be used to guide the adaptation in a more effective manner. We also explore this possibility in this thesis.

A common feature of photo management applications is the automatic organization of images based on limited interactive supervision from the user. The classes relevant to a specific user are likely to be different from the classes in publicly available image datasets such as ImageNet [40], thus there is a need for adaptation. The user can facilitate the adaptation by labeling a few images according to personal preferences.

1.2 CONTRIBUTIONS

This thesis builds upon two previously introduced methods for few-shot learning – Prototypical Networks (PN) [46] and Model-Agnostic Meta Learning (MAML) [12]. The core contributions of this thesis can be listed as follows:

- Introduction of unsupervised, semi-supervised and active few-shot adaptation problems.
- Extension of PN for unsupervised, semi-supervised and active adaptation.
- Extension of MAML for unsupervised and semi-supervised adaptation.
- Extension of the metric learning approach in few-shot classification to regression problems.

Further, both methods are also compared in various settings.

1.3 STRUCTURE

The problem of few-shot learning is formally introduced in Chapter 2. Then, we list and describe the related works to the methods proposed in the thesis. We consider two previously introduced methods for few-shot learning - Prototypical Networks (PN) and Model-Agnostic Meta-Learning (MAML). They are formally introduced in Chapter 2. In Chapter 3, we introduce the problem of semi-supervised adaptation. We relate semi-supervised adaptation to the problem of semi-supervised clustering and show how to perform this using features extracted with PN. We present improvements in the miniImagenet benchmark by making use of unlabeled data. Also, we extend MAML for semi-supervised adaptation and present results on synthetic data and compare it to PN. In Chapter 4, we consider semi-supervised adaptation using PN and show how to make more effective use of unlabeled data by taking user feedback into account. We present further improvements on the miniImagenet benchmark using active adaptation with PN. In Chapter 5, we extend both PN and MAML for unsupervised adaptation and present results on synthetic data. In Chapter 6, we compare the proposed methods and discuss some further improvements and possible future research directions.

2 FEW-SHOT LEARNING

Few-shot learning is the ability to generalize quickly from few available examples.

Few-shot learning is the ability to generalize quickly from few available examples, that is learning new concepts from a few examples of the concept. Humans are easily able to do this but machines still struggle to do this well. In particular, models trained with gradient-based methods require a large amount of data to generalize well.

The more constrained problem of few-shot classification deals with the adaptation of classifiers to previously unseen classes from a few samples. Models for few-shot classification should have the capacity to model the diversity of forms in objects around us. Such models may contain tens of thousands or millions of parameters and hence will require millions of samples to learn the parameters. So, learning good representations for image classification requires a large number of labeled samples. However, a six-year-old human child can recognize more than 10^4 categories of objects and is also able to learn many new categories per day [6]. Few-shot learning is inspired by the ability of humans to classify instances of new classes from just one, or a handful of instances. Humans are able to do this by utilizing knowledge gained from previous experience. Similarly, in few-shot learning, knowledge gained from previously learned classes are used to learn new classes. A Bayesian implementation of this idea was presented by Fei-Fei, Fergus, and Perona [11] for one-shot learning of new object categories. In [11], the information abstracted from learning previous classes is represented in form of a prior density function in the space of model parameters. Then, this prior is updated based on new training samples to produce the posterior density which can be used to recognize the new categories.

The problem of few-shot learning has gained attention since the introduction of the Omniglot dataset [27]. Lake, Salakhutdinov, and Tenenbaum [27] introduced a framework called Bayesian Program Learning (BPL) to achieve human-level performance in this dataset, outperforming existing deep learning approaches at the time. In BPL, concepts are represented as simple stochastic programs that are able to explain the observed examples. The programs are enforced to be simple under a Bayesian criterion. A program represents a concept compositionally from parts, subparts and spatial relations. BPL also defines a generative model that can generate new concepts by combining its constituents in novel ways. However, BPL induces prior knowledge about the data such as pen strokes into the model. Since then, deep learning models have been developed to achieve competitive performance on this task from the observed pixels.

Metric learning [5, 25] approaches have been successfully applied to few-shot learning [24, 46, 51]. Deep convolutional Siamese networks used for image verification were tested in the Omniglot dataset [27]. Siamese networks were able to significantly outperform the baselines and demonstrate competitive performance. Siamese networks are described in detail in Section 2.4. Siamese networks have also been successfully applied to lower the amount of data required to make meaningful predictions in drug discovery [2]. Matching Networks [51] proposed an attention mechanism over a learned embedding space to perform few-shot learning. Matching Networks are described in detail in Section 2.5.

Mensink et al. [31] explored large-scale image classification methods that are able to learn new classes continuously over time. They consider distance-based non-parametric methods such as K-nearest-neighbors and Nearest Class Mean (NCM) classifiers to learn large-scale image classification datasets with 1,000 and 10,000 classes to achieve competitive results with the state-of-the-art results at the time. Prototypical Networks [46] developed for few-shot learning can be viewed as a NCM classifier in a learned embedding space. Prototypical Networks are described in detail in Section 2.7.

Another recent approach to few-shot learning is based on optimization [36]. Ravi and Larochelle [36] proposed a method called Meta-LSTM in which an LSTM called the *meta-learner* is used to adapt the parameters of another LSTM network called the *learner* to perform few-shot learning. Meta-LSTM is described in detail in Section 2.6. Finn, Abbeel, and Levine [12] took a similar approach and proposed a method called MAML to learn an initialization of the parameters of a network such that it can be adapted to a new task from a few samples in a few gradient update steps. MAML is described in detail in Section 2.8.

Various other approaches have been taken towards few-shot learning. Attentive Recurrent Comparators [45] use an RNN-based controller with an attention mechanism to match similar images to perform few-shot learning. Neural networks with augmented memory are able to encode and retrieve data efficiently enabling them to make accurate inferences from few examples [39]. Deep generative models have been exhibited to perform one-shot generalization by unconditional sampling, generating new exemplars of a given concept, and generating new exemplars of a family of concepts [39]. An extension of variational autoencoders has been used to learn to compute summary statistics of different datasets in a completely unsupervised manner [10].

In few-shot learning, during the training phase of development, the model is trained with a background set that enables the model to have general background knowledge about the data. In the testing phase, a test set of k samples from N novel classes are used and the

model is expected to assign each sample to one of the N classes. The few-shot learning problem can be approached in different ways. In this thesis, we adopt a meta-learning approach to the problem.

2.1 META-LEARNING

Meta-learning is the modeling of the process of learning itself and improving it by experience.

Meta-learning or learning to learn is a skill that enables rapid acquisition of new concepts [44, 50]. Meta-learning is the modeling of the process of learning itself and improving it by experience. Meta-learning aims to train a model on a variety of different learning tasks, such that it can solve new learning tasks using only a small number of training samples. Meta-learning is enabled by the slow extraction of the similarities in the learning process across different tasks and generalizing to new tasks. Meta-learning considers the occurrence of learning at two different levels: quick acquisition of knowledge within each task (we refer to this as *adaptation*) and slower extraction of information across all tasks. The slower learning process guides the adaptation. Meta-learning has been successfully applied to the problem of few-shot learning [12, 34, 36]. In the meta-learning approach, a model or classifier is explicitly optimized to adapt itself to new tasks or classes from a few samples. This is possible because of the knowledge extracted by the model from previous experience with related classes.

2.2 TASK DESCRIPTION

To be consistent throughout the thesis and to avoid ambiguity, this section describes the mathematical formulation of few-shot learning. In few-shot learning, the dataset D consists of different tasks/episodes. Each task t consists of a support set S_t and a query set Q_t. The dataset D is split into D_{train}, $D_{\text{validation}}$ and D_{test} as is standard practice in machine learning. However, few-shot learning operates at a higher level of abstraction where the dataset itself consists of smaller datasets for different tasks. We are interested in learning an adaptation procedure that can adapt the model to a task t from the support set S_t of the task to perform well on the query set Q_t.

Few-shot classification models are tested using N-way k-shot classification tasks.

Few-shot classification models are tested using N-way k-shot classification tasks. In an N-way k-shot classification task, the support set $S_t = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ consists of k labeled examples for N unseen classes and the model is evaluated in its ability to correctly classify new samples in the query set $Q_t = \{(\mathbf{x}_{n+1}, y_{n+1}), ..., (\mathbf{x}_{n+m}, y_{n+m})\}$ into one of the N classes where $\mathbf{x}_j$ are inputs and y_j are the corresponding labels. This episodic formulation of data in few-shot classification is illustrated in Figure 2. In practice, the support set for a new task is created by sampling N different classes from a dataset

Figure 2: Episodic formulation of data in few-shot classification.

with a larger number of classes and k samples from each of the N classes. The query set is formed by sampling a particular number of additional samples (m) from the same N classes in the support set.

2.3 APPROACHES

In this section, we outline the common approaches taken to solving the problem of few-shot learning:

1. *Similarity-Based*

   In this approach, new tasks are handled similarly to previous experience with similar tasks. There exist different ways to do this:

   a) Model-Based

   Few-shot learning can be cast as a sequence-to-sequence modeling problem where each task is a sequence of input-output pairs. This approach is taken by [32, 34, 42]. The RNN based approach used by [42] is illustrated in Figure 3. Classification tasks are sampled in an episodic manner as described in Section 2.2 and an RNN with augmented memory is trained to classify new images based on the previously provided image-label pairs in the same episode. This approach is flexible but less efficient than others because it should develop learning capabilities from scratch.

   b) Metric-Based

   This is a more explicit approach in which we learn a metric space based on the similarity of the images so that similar images are closer. New images are assigned a label based on its distance (in the learned metric space) to the images in the support set. This approach is used by [24, 46, 51].

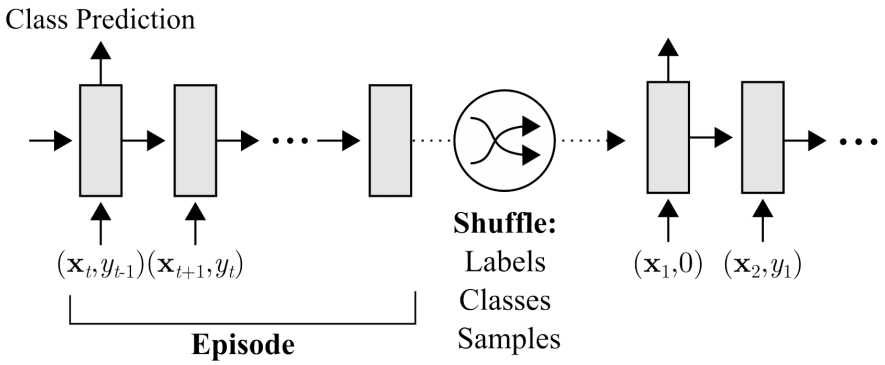


Figure 3: Illustration of RNN based approach to few-shot learning. Figure taken from [42].

2. *Optimization-Based*

   This approach involves learning a good initialization and update mechanisms to adapt to new tasks. The gradient provides useful information for unseen data. Use of gradients can help in adapting to unseen situations. This approach is taken by [36] and [12].

We consider only the metric-based (Prototypical Networks) and optimization-based (Model-Agnostic Meta-Learning) approaches in this thesis.

2.4 SIAMESE NETWORKS

If there exists a model that can compare two images and predict the probability that they belong to the same class, then unseen images in the query set can be classified by comparing it with all images in the support set and classifying it to the class with the highest probability. One such model, a Siamese network [24] consists of two networks that share weights (hence the name) to embed two images and then a logistic regression classifier on top of it to predict the probability that the images belong to the same class based on the distance between their embeddings. Siamese networks are a simple yet effective metric learning approach commonly applied in few-shot learning problems [2, 24]. Siamese networks increase the effective size of the data by pairing all the samples. However, this requires sampling pairs of samples from the whole dataset and it might take a long time to go through the whole dataset.

2.5 MATCHING NETWORKS

A basic principle in machine learning is to match training and testing conditions.

Vinyals et al. [51] points out the obvious but important fact that few-shot learning is easier if we train the network to do few-shot learning. A basic principle in machine learning is to match training and testing

conditions. In Vinyals et al. [51], the classes in the training dataset D_{train} are sampled into N-way k-shot classification tasks with support and query sets similar to testing conditions.

Neural networks require lots of data and take time to learn due to its parametric nature.

Neural networks require lots of data and are time-consuming to train due to their parametric nature. Non-parametric models such as K-nearest-neighbors are able to quickly assimilate new classes. A Matching Network [51] is essentially a fully end-to-end nearest neighbor classifier in which nearest neighbor classification is performed on features extracted by a neural network. The structure of Matching Networks is illustrated in Figure 4. Matching Networks in its simplest form can be expressed as:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \,, \tag{1}$$

where a is the attention kernel. Note that this form does not limit it to classification problems. Equation 1 can be viewed as an attention function. Generally, an *attention function* maps a query and a set of key-value pairs (support set) to an output. This is also the requirement of few-shot classification tasks.

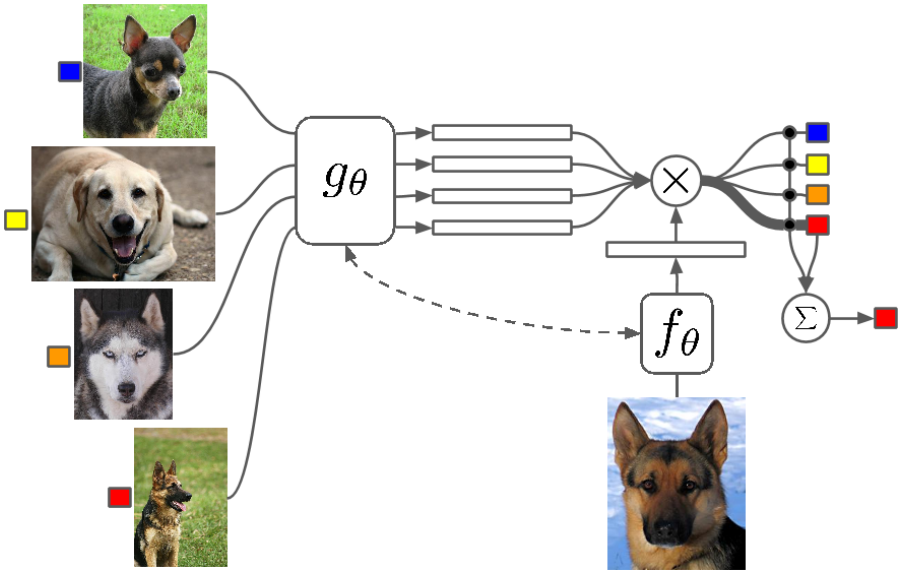


Figure 4: Illustration of the structure of Matching Networks. The support set is embedded using a feature extractor g_θ and the query sample is embedded using a different (can also be the same) feature extractor f_θ. Then, the cosine similarity of the embedding of the query sample with embeddings of the support samples are computed and the final classification is performed by weighing the class labels of the support samples with the softmax of these similarities. Figure taken from [51].

2.5.1 *Relation to Gaussian Processes*

Gaussian Processes (GP) are non-parametric probabilistic models applied to classification and regression problems. A GP defines a prior over functions. These prior functions are converted to a posterior over functions based on the training data. The distribution over the functions are defined at values at a finite, but arbitrary, set of points, say $x_1, \ldots, x_N$. GP models the similarity of data points using a kernel function and predicts the value for an unseen sample based on its similarity to all samples in the training data. In GP, the distribution over function values $p(f(x_1), ..., f(x_N))$ is assumed to be jointly Gaussian with mean μ and covariance Σ. The covariance matrix Σ is computed using a positive-definite kernel function: $\Sigma_{ij} = k(x_i, x_j)$. In GP, we can analytically compute the posterior distribution over the functions from the priors and the training samples.

Assume that we have training samples defined by input-output pairs $\{(x_1, y_1), ..., (x_N, y_N)\}$. Let

$$\begin{aligned}
\mathbf{x} &= [x_1 ... x_N]^T, \\
\mathbf{y} &= [y_1 ... y_N]^T, \\
\mathbf{f} &= [f(x_1) ... f(y_N)]^T, \\
\mathbf{k}_{\hat{x}\mathbf{x}} &= [k(\hat{x}, x_1) ... k(\hat{x}, x_N)]^T, \\
\Sigma = \mathbf{K}_{\mathbf{xx}} &= \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \ldots & k(x_N, x_N) \end{bmatrix}.
\end{aligned}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the vector of input and output values respectively. The prior is often assumed to have zero mean:

$$\mathbf{f} \sim N(0, \mathbf{K}_{\mathbf{xx}}).$$

The kernel function k is parametrized by parameters θ which can be estimated by maximizing the log marginal likelihood. Making predictions in GP is performed using the predictive distribution

$$p(\hat{y}|\hat{x}, \{(x_i, y_i)\}) = N(\hat{y}|\mu, \Sigma)$$

for an unseen sample $\hat{x}$. Then, the posterior mean is computed as:

$$\hat{\mu} = \mathbf{k}_{\hat{x}\mathbf{x}} \mathbf{K}_{\mathbf{xx}}^{-1} \mathbf{y}, \tag{2}$$

and the posterior variance is computed as:

$$\hat{\Sigma} = k(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{k}_{\hat{x}\mathbf{x}} \mathbf{K}_{\mathbf{xx}}^{-1} \mathbf{k}_{\hat{x}\mathbf{x}}^T.$$

From Equation 2, we can see that the posterior mean is computed as a linear combination of the outputs $\mathbf{y}$ in the training set. The outputs are weighted by the similarity of the unseen sample $\hat{x}$ to all the

training samples $\mathbf{x}$ (in the term $\mathbf{k}_{\hat{\mathbf{x}}\mathbf{x}}$) and also by the inverse of the correlation between all the training samples (in the term $\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}$). The basic form of Matching Networks (Equation 1) also computes the prediction of unseen input $\hat{x}$ as a linear combination of the outputs in the support set. Comparing them, we can see that Equation 1 does not model the correlation between all the training samples. Matching Networks [51] embeds each training sample x_i in the context of the support set using a bidirectional Long Short-Term Memory (LSTM).

2.6 META-LSTM

Consider the update rule of gradient descent used to train deep neural networks:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_{t-1} \,,$$

where θ_{t-1} and θ_t are the parameters before and after the update at timestep t respectively, α_t is the learning rate at time t and $\nabla_{\theta_{t-1}} \mathcal{L}_t$ is the gradient of a loss function computed with respect to the parameters θ at θ_{t-1}. Now consider the update rule for cell state in an LSTM cell [18]:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \,,$$

where c_{t-1} and c_t are the cell states at timesteps $t-1$ and t respectively and $\tilde{c}_t$ is the candidate value for new cell state. The new cell state c_t is computed by gating the old cell state c_{t-1} with a forget gate f_t and the candidate cell state $\tilde{c}_t$ with an input gate i_t. It can be seen that the parameter update rule of standard optimization algorithms such as Stochastic Gradient Descent (SGD) resembles the update rule of an LSTM cell (when $f_t = 1$, $c_{t-1} = \theta_{t-1}$, $i_t = \alpha_t$, and $\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t$). This was used in Andrychowicz et al. [3] to learn better optimizers using LSTM networks. Inspired by [3], Ravi and Larochelle [36] noted that optimization is a good model for few-shot learning. Few-shot learning can be cast as an optimization problem to adapt the parameters of a network from a few samples. This involves learning initializations and parameters update mechanisms that enable this efficient adaptation. In [36], an LSTM network (called a meta-learner) is used to learn the update rule of another LSTM network (called the learner). The structure of the network is illustrated in Figure 5. The parameters of the learner network are updated by the meta-learner network based on the gradients computed from a few samples. Parameters of the learner and the meta-learner network are trained together in an end-to-end fashion in the few-shot setting. It should be noted that the size of meta-learner LSTM cell grows rapidly with increasing the number of parameters. In practice each coordinate in the parameter vector has its own hidden and cell state values but the LSTM parameters are the same across all coordinates.

The update rule of standard optimization algorithms resembles the update rule of an LSTM cell

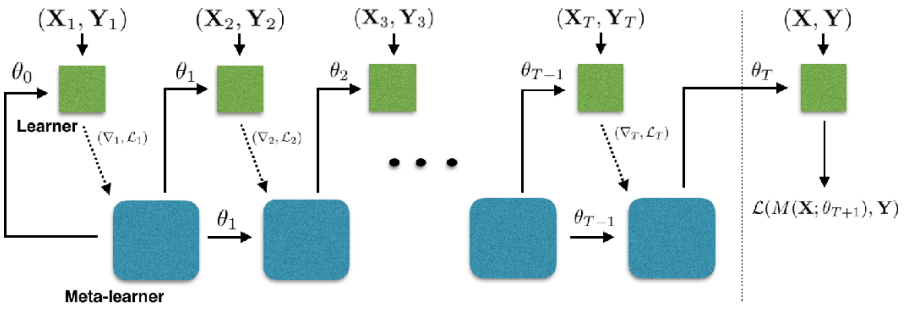


Figure 5: Illustration of the structure of a Meta-LSTM network. The meta-learner LSTM network updates the weights of the learner LSTM network based on the gradients computed using the samples in the support set. Figure taken from [36].

2.7 PROTOTYPICAL NETWORKS

Prototypical Networks (PN) is essentially a deep equivalent of a Nearest Class Mean classifier

Similar to the correspondence between Matching Networks and K-nearest neighbors, Prototypical Networks (PN) is essentially a deep equivalent of a Nearest Class Mean (NCM) classifier [31]. Prototypical Networks compute representations of the inputs $\mathbf{x}$ using an embedding function f parameterized with θ: $\mathbf{z} = f_\theta(\mathbf{x})$. Each class c is represented in the embedding space by a prototype vector which is computed as the mean vector of the embedded inputs for all the examples $S_t^{(c)}$ of the corresponding class c:

$$\mathbf{m}_c = \frac{1}{|S_t^{(c)}|} \sum_{(\mathbf{x}_j, y_j) \in S_t^{(c)}} f_\theta(\mathbf{x}_j) \,. \tag{3}$$

The distribution over predicted labels y for a new sample $\mathbf{x}$ is computed using softmax over negative distances to the prototypes in the embedding space:

$$p(y = c|\mathbf{x}, \{\mathbf{m}_c\}) = \frac{\exp(-d(f_\theta(\mathbf{x}), \mathbf{m}_c))}{\sum_{c'} \exp(-d(f_\theta(\mathbf{x}), \mathbf{m}_{c'}))} \,. \tag{4}$$

Parameters θ are updated so as to improve the likelihood computed on the query set:

$$\sum_{(\mathbf{x}_j, y_j) \in Q_t} \log p(y = y_j|\mathbf{x}_j, \{\mathbf{m}_c\}) \,,$$

which is computed using Equation 4 with the estimated prototypes. The computation graph of supervised adaptation with PN is illustrated in Figure 6.

2.8 MODEL-AGNOSTIC META-LEARNING

Bayesian hierarchical modeling can be used a theoretical framework to formalize meta-learning as inference for a set of parameters that

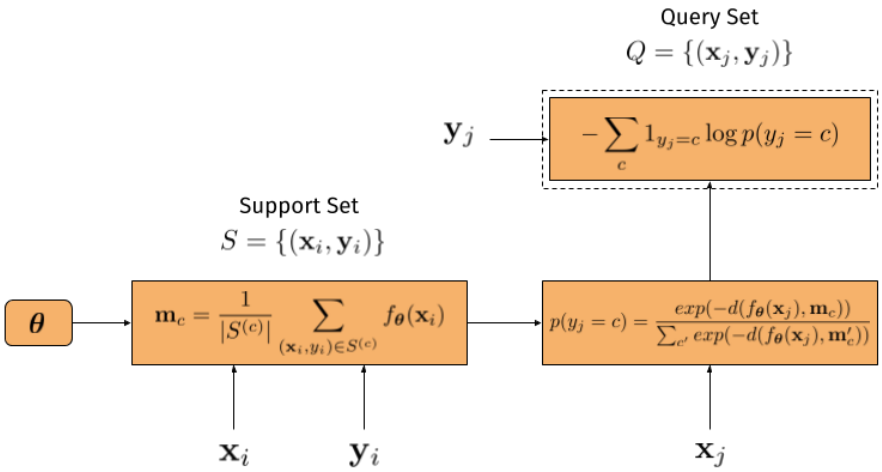


Figure 6: Computational graph of supervised adaptation with PN. Parameters are represented by rounded rectangles and computations are represented by rectangles. The loss function used for training is highlighted in a dashed rectangle.

are shared across different tasks [15]. MAML can be viewed as a method for probabilistic inference in a hierarchical Bayesian model. In MAML, we model the prior knowledge across different tasks as: an initialization in the model parameter space, the learning rate and the number of iterations required to adapt the initialization to different tasks. The initialization of the parameters is learned such that they can be adapted to a particular distribution of tasks in a few gradient steps. Suppose there exists a set of related classification tasks in which each task t is described by data $D^{(t)} = \{(\mathbf{x}_j, y_j)\}$ with inputs $\mathbf{x}_j$ and targets y_j belonging to the support set $S_t = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ and query set $Q_t = \{(\mathbf{x}_{n+1}, y_{n+1}), ..., (\mathbf{x}_{n+m}, y_{n+m})\}$. In MAML, a common classifier $y \approx f_\theta(\mathbf{x})$ is adapted to task t by updating its parameters θ using one or more gradient descent steps of the form

$$\theta_t = \theta_* - \alpha \sum_{(\mathbf{x}_j, y_j) \in S_t} \nabla_\theta L\left(f_\theta(\mathbf{x}_j), y_j\right)\Big|_{\theta=\theta_*}, \tag{5}$$

where L is the loss function computed on the samples in the support set S_t. In classification tasks, the loss function L is often the cross-entropy function:

$$L(S_t, \theta) = -\sum_{(\mathbf{x}_j, y_j) \in S_t} y_j \log p(y_j | \mathbf{x}_j, \theta)\,. \tag{6}$$

The initial values θ_* and the vector of learning rates $\boldsymbol{\alpha}$ are the parameters tuned during MAML training. Training happens by going through a set of tasks, adapting the classifier to each task t using Equation 5 and updating the parameters θ_*, $\boldsymbol{\alpha}$ so as to optimize the performance of the adapted models $f_{\theta_t}(\mathbf{x})$ on the query sets Q_t. The

objective function for meta-training can be mathematically expressed as,

$$\min_{\theta_*, \alpha} \sum_{t \sim p(t)} \sum_{(x_j, y_j) \in Q_t} L\left(f_{\theta_t}(x_j), y_j\right) . \tag{7}$$

where $p(t)$ is the distribution of tasks. This optimization is performed using standard backpropagation, which involves a gradient through a gradient since the computation of θ_t contains $\nabla_\theta L$. The computational graph of supervised adaptation using MAML is illustrated in Figure 7.

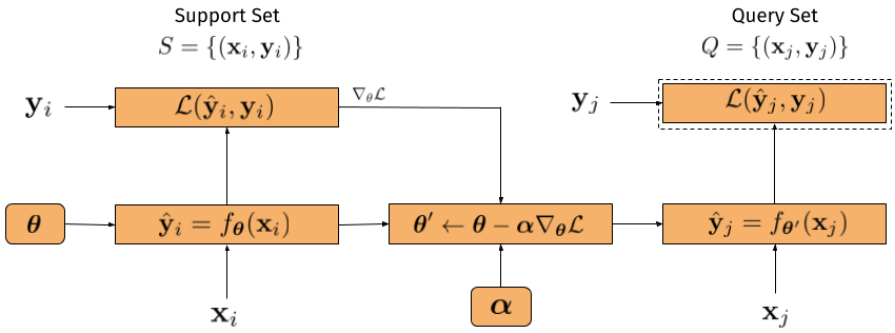


Figure 7: Computational graph of supervised adaptation with MAML. Parameters are represented by rounded rectangles and computations are represented by rectangles. The loss function used for training is highlighted in a dashed rectangle.

It is notable that MAML does not induce any additional parameters. It is applicable to any differentiable model and it can be directly applied to classification, regression or reinforcement learning problems.

2.8.1 *Learning Initialization as Meta-Learning*

One approach to meta-learning is to optimize for an initial representation that can effectively be adapted to a new task from a small number of samples in a few gradient steps.

One particular approach to meta-learning is to directly optimize for an initial representation that can effectively be adapted to a new task from a small number of samples in a few gradient steps. Finn, Abbeel, and Levine [12] showed that learning an initialization and using the standard gradient-descent update rule to adapt it is a simple yet effective approach to meta-learning. The intuition behind MAML is illustrated in Figure 8. During training, the parameters of the network are optimized to learn good initializations that are close to the optimal parameters of different tasks.

2.8.2 *Learning-Rate Schemes*

The plot in Figure 9 illustrates the convergence during training for different learning rate schemes of MAML. We use the synthetic sine re-

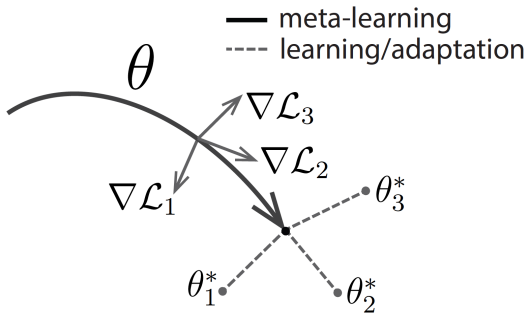


Figure 8: MAML optimizes for an initialization of the parameters θ that can be efficiently adapted to different tasks in a few gradient steps with a few samples. Figure from [12].

gression dataset introduced in [12]. Parameter-wise learning rates perform the best, converging faster and to a better minimum. This supports the results presented by Li et al. [29]. However, parameter-wise learning rates are memory expensive and layer-wise learning rates perform competitively with much smaller memory requirements.

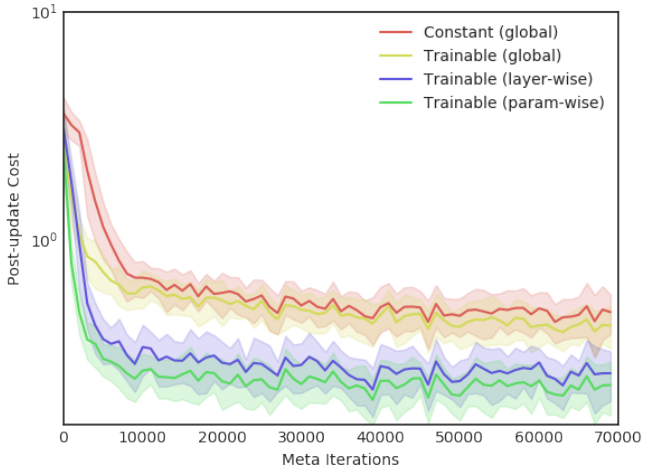


Figure 9: Comparison of different learning rate schemes.

2.9 EXPERIMENTS ON MINIIMAGENET

In this thesis, we consider two existing methods – Prototypical Networks (described in Section 2.7) and Model-Agnostic Meta-Learning (described in Section 2.8). In this section, we evaluate these methods for supervised adaptation on image data. We consider the miniImagenet recognition task proposed by Vinyals et al. [51] to test the methods proposed in this thesis. The dataset involves 60,000 color images downsampled to a resolution of 84×84; the images are from 64 training classes, 12 validation classes, and 24 test classes from ImageNet

[40]. We use the same split as [36] and follow the experimental setup which involves N-way k-shot classification, similarly to [51] that is, every task at test time contains N classes with k labeled examples from each class. Following previous works, we use 15 test samples per class for each task during training and for evaluation. We evaluate the model on 2400 tasks which involve the 24 classes reserved for testing.

For comparability, we use the same convolutional architecture used in previous works [12, 36, 46, 51] which consists of four convolutional blocks (we refer to this as the four-block architecture). Each block consists of a 3×3 convolutional layer with 64 channels followed by batch normalization [20], ReLU non-linearity and a 2×2 max-pooling layer which results in an embedding space of dimensionality 1600. In Prototypical Networks [46], the model was fine-tuned by having more classes at training time than at test time (e.g., 30-way training and 5-way testing) and a learning rate decay schedule. We use a constant learning rate and 5-way training for simplicity. In this thesis, the results obtained with our implementation of PN are distinguished as "PN (ours)".

One challenge with the miniImagenet data set is that it requires rather complex features but it contains relatively little amount of data. Therefore, preventing overfitting becomes an important issue. Most previous works [12, 36, 46, 51] used conventional convolutional networks with a small number of layers to prevent overfitting. This simple network may not be expressive enough to capture the relevant complex features. So, we also consider a Residual Network [16] as the embedding network. Residual Networks achieve state-of-the-art performance on the ImageNet dataset [19] and have also been successfully applied to other problems such as object detection [38] and semantic segmentation [8]. We use a Wide Residual Network [53] of depth 16 and a widening factor of 6. We also use an 8×8 pooling with a stride of 4 at the end to obtain embeddings of dimensionality 384. The network is regularized with dropout with a rate of 0.3. We use the Adam optimizer [22] with a learning rate of 0.01 for training the ResNet and 0.001 for training the four-block architecture. We perform early stopping to prevent overfitting.[1] We train the ResNet model with $N = 30$ classes in each task for 1-shot classification and $N = 20$ classes in each task for 5-shot classification, similarly to the original PN paper [46]. For the ResNet model, we tried varying the number of classes during training and observed that it had a much smaller impact on the results compared to the observations in [46]. This suggests that tuning this regularization parameter is less important for this architecture.

1 Residual Networks are typically trained using stochastic gradient descent with momentum and we expect better results by doing the same and fine-tuning the hyperparameters.

The 5-way testing results presented in Table 1 and 20-way testing results presented in Table 2 show that our approach scales to both feature extractor architectures. One can observe that using the Wide ResNets to learn the embedding space yields noticeable improvements of the classification accuracy compared to the baseline methods. The improvements are more significant for the 20-way classification.

MODEL	1-SHOT	5-SHOT
fine-tuning baseline [36]	28.86 ± 0.54	49.79 ± 0.79
nearest-neighbor baseline [36]	41.08 ± 0.70	51.04 ± 0.65
Meta-LSTM [36]	43.44 ± 0.77	60.60 ± 0.71
Matching nets [51]	46.6	60.0
MAML [12]	48.70 ± 1.84	63.11 ± 0.92
PN [46]	49.42 ± 0.78	68.20 ± 0.66
Meta-SGD [29]	50.47 ± 1.87	64.03 ± 0.94
PN (ours)	48.06 ± 0.82	64.65 ± 0.72
Resnet PN (ours)	$\mathbf{51.69 \pm 0.42}$	$\mathbf{69.57 \pm 0.64}$

Table 1: Average 5-way testing classification accuracy (with 95% confidence intervals) on *miniImagenet*.

MODEL	1-SHOT	5-SHOT
Meta-LSTM [36]	16.70 ± 0.23	26.06 ± 0.25
Matching nets [51]	17.31 ± 0.22	22.69 ± 0.20
MAML [12]	16.49 ± 0.58	19.29 ± 0.29
Meta-SGD [29]	17.56 ± 0.64	28.92 ± 0.35
PN (ours)	20.36 ± 0.24	34.42 ± 0.23
Resnet PN (ours)	$\mathbf{23.35 \pm 0.28}$	$\mathbf{41.10 \pm 0.25}$

Table 2: Average 20-way testing classification accuracy (with 95% confidence intervals) on *miniImagenet*.

2.10 CONTINUAL LEARNING WITH PN

Assuming that the embeddings of samples are clustered (according to the classes) in the embedding space produced by feature extractors in PN, we can classify each class by merely storing all the prototypes. A new sample can be classified by assigning it to the class of the nearest prototype in the feature space. In PN, the feature extractor

	Training shot	Testing shot	
		1-shot	5-shot
PN (ours)	1	48.06 ± 0.82	63.20 ± 0.77
	5	42.71 ± 0.75	64.65 ± 0.72
	[1..5]	48.12 ± 0.78	64.84 ± 0.70
Resnet PN	1	51.69 ± 0.47	67.43 ± 0.63
	5	50.44 ± 0.48	69.57 ± 0.59
	[1..5]	51.40 ± 0.46	68.97 ± 0.55

Table 3: Results of fully supervised adaptation using PN as a function of the number k of samples per class during training. [1..5] denotes varying k in the range [1, 5].

remains the same after training. This enables us to perform continual learning during test time.

Continual learning can be done as follows: train PN on a background training set and store the prototypes of all training classes. During test time, compute prototypes for new classes using the pre-trained feature extractor and maintain a moving average for the prototypes of all classes. Add samples of new classes to the training set. Once in a while, re-train the feature extractor with the new training set. We acknowledge this possibility but do not further explore this idea.

We propose to use a varying number of samples per class during training for better generalization to the number of shots during test time.

Snell, Swersky, and Zemel [46] showed that the classification performance of Prototypical Networks is sensitive to the number N of classes per task during training and that it was necessary to match the number k of samples per class during training and testing. We believe that using a larger number of classes (N way) works as a regularizer. However, varying N effectively changes the batch size and therefore the learning rate needs to be adapted to N. By tuning the learning rate for different N, we observed a smaller effect of this regularization on the adaptation accuracy.

The dependency on k implies that the embedding learned by the network does not generalize well to a different number of samples during test time. This is inconvenient especially if the number of the labeled example is unknown in advance and it can grow, for example, as a result of interaction with the user. To address this problem, we propose to use a varying number of samples per class during training for better generalization to the number of shots during test time. The results in Table 3 illustrate that this strategy is effective and reduces the sensitivity to k during training.

2.11 CONCLUSION

In this chapter, we formally introduced the problem of few-shot learning. Then, we introduced the existing relevant work on few-shot learning and described them. Mainly, we introduce Prototypical Networks (PN) and Model-Agnostic Meta-Learning (MAML). Further, we evaluated PN and MAML for supervised adaptation on the miniImagenet image classification benchmark. Finally, we considered the possibility of continual learning with PN and proposed to use a varying number of samples per class during training for better generalization to the number of shots during test time.

3 SEMI-SUPERVISED ADAPTATION

We consider the problem of fast adaptation to new classification tasks using a few labeled samples and (potentially many) unlabeled samples. This problem of semi-supervised few-shot adaptation is relevant to many practical applications. Consider the previously introduced application of automatic photo organization. In this case, the user can facilitate the adaptation by labeling a few images by personal preferences. However, the learning system also has access to lots of unlabeled images and it can make use of them to improve the classification accuracy.

Semi-supervised learning methods make certain assumptions on the underlying distribution of data [7]. Commonly used assumptions are:

1. Smoothness assumption: the label function is smoother in high-density regions than in low-density regions, that is the points that are close to each other are likely to share the same label.
2. Cluster assumption: the data points tend to form clusters and points in the same cluster are likely to share the same label.
3. Manifold assumption: the data lie approximately on a low-dimensional manifold.

3.1 SEMI-SUPERVISED ADAPTATION WITH PN

PN tends to produce clustered data representations on the embedding space

We address the problem of semi-supervised classification by making the cluster assumption. This is motivated by the observation that PN tends to produce clustered data representations in the embedding space. This is indirectly induced by the formulation of PN which enforces samples in a particular class to be closer to the cluster mean estimated from very few samples. The clustering approach to semi-supervised learning is often referred as semi-supervised clustering [4]. We relate semi-supervised few-shot adaptation to the problem of semi-supervised clustering. Semi-supervised clustering is a problem considered mainly in the context of text document clustering [1, 21].

We relate semi-supervised few-shot adaptation to the problem of semi-supervised clustering.

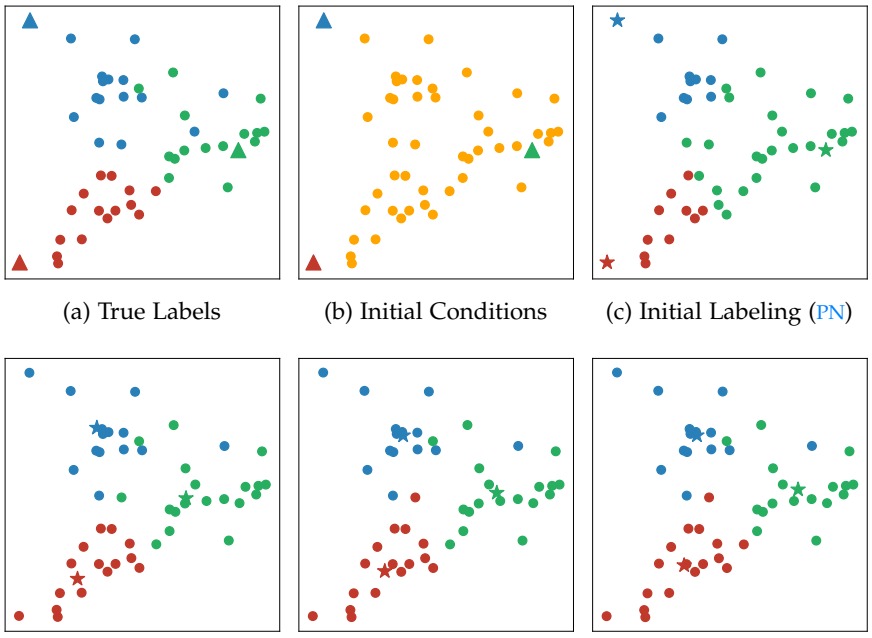
Our proposed algorithm performs semi-supervised adaptation during test time. We use a feature extractor from a PN trained with the standard training procedure which involves sampling of tasks, computing the prototypes of each class and updating parameters θ of the embedding network using stochastic gradient descent, where the gradient is computed using samples of the query set. At test time,

We perform semi-supervised adaptation by seeding K-means clustering (on the PN embedding space) with the prototypes.

the prototypes are first estimated with the labeled data using Equation 3. We then perform the standard K-means algorithm [30] on the embeddings of both labeled and unlabeled data initializing the cluster means with the prototypes computed from the labeled data. This corresponds to the seeding approach of K-means proposed by Basu, Banerjee, and Mooney [4]. The algorithm typically converges in just a few iterations (see Table 5). We have also tried the constrained K-means approach in which the cluster membership of the labeled examples is never changed. Both approaches yielded similar results and therefore we present only the results of the seeding approach, which was slightly better.

The proposed algorithm is illustrated in Figure 10. Consider a 3-way 1-shot classification task (sampled from miniImagenet). That is, a classification task with three classes and one labeled sample per class. Consider that we also have 15 unlabeled samples per class. Such a task is illustrated in Figure 10a. The two-dimensional visualizations are produced by projecting the 384-dimensional features onto the principal subspace of the prototypes computed from the support set. In Figure 10a, the true labels of both the labeled and unlabeled samples are illustrated for comparison of performance of the adaptation mechanisms. The labeled samples (support set) are segregated using triangle markers and unlabeled samples are segregated using circle markers. The initial condition with labeled and unlabeled samples are shown in Figure 10b. The three different classes are colored as red, green and blue. Unlabeled samples are colored as orange. The prototypes for each class are computed as the mean of samples in each class. In this case, the prototypes are the samples in the support set itself since there is only one labeled sample per class. The predictions produced using the original PN procedure is shown in Figure 10c. Now we initialize the K-means clustering algorithm with the class prototypes and the evolution of the prototypes and labeling of the unlabeled samples is shown in Figure 10d, Figure 10e and Figure 10f. The predictions immediately get slightly better after the first iteration but it already nears convergence in the next iteration and converges in less than 10 iterations.

The proposed algorithm is related to the one concurrently developed by Ren et al. [37] who do semi-supervised few-shot classification using the constrained K-means clustering. The main difference is that they perform clustering also at training time and therefore they use soft cluster assignments to keep the computational graph differentiable. They do only one iteration of K-means, as doing more iterations does not improve the performance. We obtained similar results using soft cluster assignment and found experimentally that K-means with hard clustering behaves more robustly (see results in Table 5). Furthermore, Ren et al. [37] consider the scenario in which the unlabeled support set may contain samples from irrelevant classes.



(a) True Labels (b) Initial Conditions (c) Initial Labeling (PN)

(d) After 1st Iteration (e) After 2nd Iterations (f) After 10 Iterations

Figure 10: Illustration of semi-supervised clustering for semi-supervised adaptation with PN.

3.2 SEMI-SUPERVISED ADAPTATION WITH MAML

We perform classification in two steps: feature extraction and classification.

For semi-supervised adaptation, we perform classification in two steps:

1. Inputs $\mathbf{x}$ are transformed into features $\mathbf{z}$ using function $\mathbf{z} = f_\theta(\mathbf{x})$ with parameters θ;
2. Predicted labels are computed using function $\hat{y} = g_\omega(\mathbf{z})$ with parameters ω.

We first adapt the feature extractor using unlabeled data and then the classifier using labeled data.

The proposed method for semi-supervised adaptation using MAML is as follows: We first use unlabeled data to adapt the feature extractor parameters θ using Equation 5 with an unsupervised cost function C:

$$\theta_t = \theta_* - \alpha \sum_{x_j \in S_t} \nabla_\theta C\left(f_\theta(x_j)\right)\Big|_{\theta=\theta_*}, \tag{8}$$

We parametrize the unsupervised loss function with a neural network and meta-learn it.

where C is some kind of a cost function that measures the quality of the extracted features. Several auxiliary cost functions using unlabeled data have been proposed in the literature to improve the classification performance in the semi-supervised scenario [see, e.g., 14, 26, 33, 35, 49]. Instead of specifying that extra cost, we propose to parametrize it with a neural network $C_{\omega_c}(\mathbf{z})$ with parameters ω_c and learn it in the same meta-training procedure.

Finally, we use labeled data to adapt the classifier parameters ω using an update rule similar to Equation 5:

$$\omega_t = \omega_* - \alpha_\omega \sum_{(x_j,y_j)\in S_t} \nabla_\omega L\left(g_\omega(f_{\theta_t}(x_j)), y_j\right)\Big|_{\omega=\omega_*} . \tag{9}$$

The difference to the fully supervised case is that the loss function is computed using the adapted features $\mathbf{z}$ instead of raw inputs $\mathbf{x}$. Similar to supervised adaptation, the parameters are tuned to optimize the performance on the query sets Q_t:

$$\min_{\theta,\alpha,\omega_c,\omega,\alpha_\omega} \sum_{t\sim p(t)} \sum_{(x_j,y_j)\in Q_t} L\left(g_\omega(f_{\theta_t}(x_j)), y_j\right) . \tag{10}$$

The tuned parameters are θ, α, ω_c, ω and α_ω. The full computational graph for semi-supervised adaptation is illustrated in Figure 11.

Note that using different learning rates α and α_ω was crucial to make this approach work. This could be attributed to the difficulty in balancing the training signals provided by the labeled and unlabeled samples. In the method proposed here, the unsupervised cost is forced to be developed because the feature extraction part of the network is adapted solely using the unsupervised loss. We explored different learning rate schemes in MAML and found layer-wise or parameter-wise learning rates to be an important factor in balancing the signals.

3.3 EXPERIMENTS ON SYNTHETIC DATA

To test the proposed methods, we created a synthetic data set which is fast to experiment with. The dataset consists of a set of two-dimensional classification tasks with two classes, in which the optimal decision boundary is a sine wave (see Figure 12). The amplitude A of the optimal decision boundary varies across tasks within $[0.1, 5.0]$ and the phase ϕ varies within $[0, \pi]$. The first dimension of the data samples is drawn uniformly from $[-5, 5]$ and the second dimension is computed as

$$x_2 = A\sin(x_1 + \phi) + \epsilon$$

where ϵ is a noise term with the Laplace distribution with mean ± 2 (depending on the class) and scale parameter 0.5. We sampled 100 tasks for training and 1000 tasks for testing.

Examples of the decision boundaries produced by PN on a semi-supervised test task are shown in Figure 13. Examples of the decision boundaries produced by MAML on two test tasks are shown in Figure 14. Note that even for a small number of labeled examples, the adapted decision boundaries resemble the sine wave, thus the knowledge is transferred between tasks.

PN generally performs better in the fully supervised setting, while MAML is much more efficient in making use of unlabeled data.

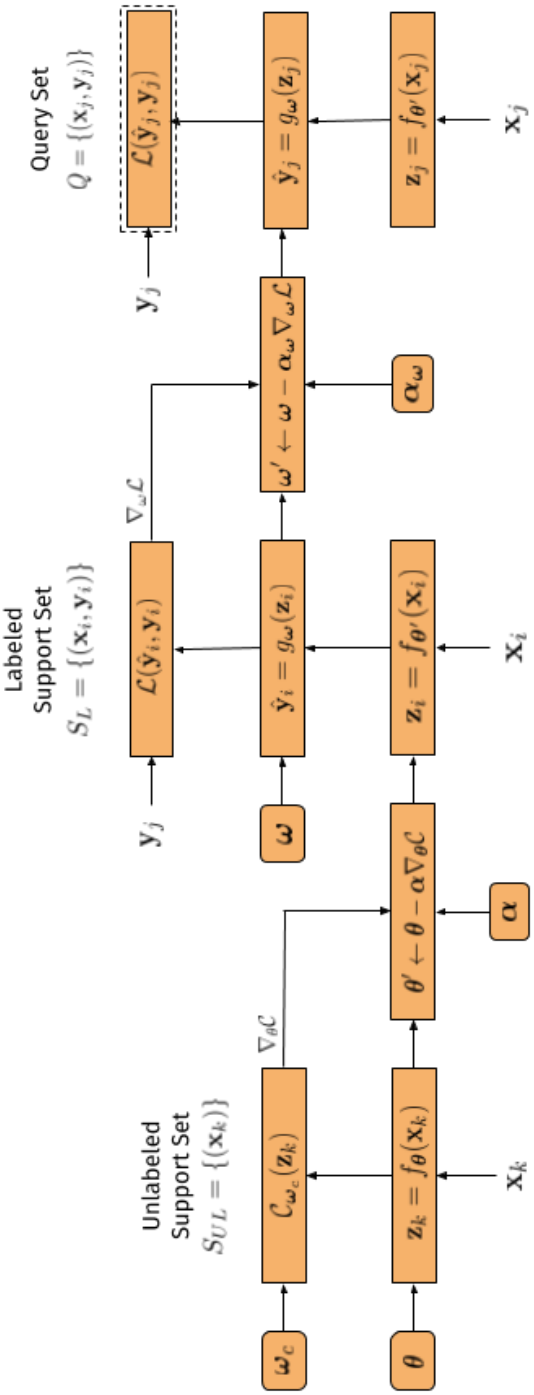


Figure 11: Computational graph of semi-supervised adaptation with MAML. Parameters are represented by rounded rectangles and computations are represented by rectangles. The loss function used for training is highlighted in a dashed rectangle.

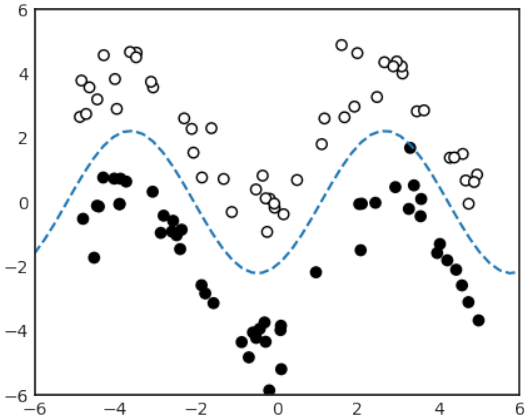


Figure 12: An example task from the sine classification dataset.

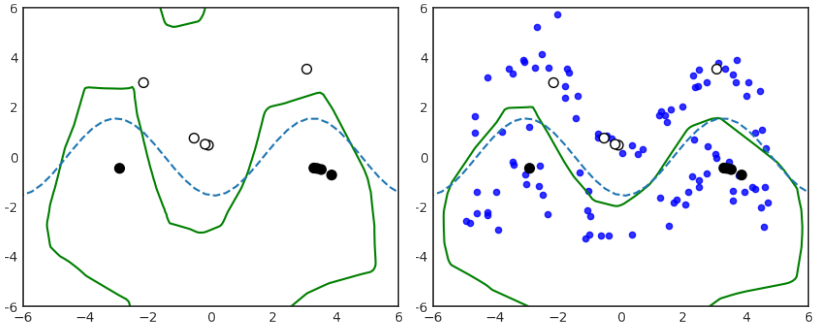


Figure 13: Left: Example of supervised adaptation using PN to a test task from 10 labeled samples. Right: Example of semi-supervised adaptation using PN to a test task from the same 10 labeled samples and 100 unlabeled samples. The blue dots correspond to unlabeled samples.

The classification accuracy of semi-supervised adaptation with PN and MAML are shown in Figure 15. We use a fully connected network with two hidden layers of size 100 with ReLU non-linearity as the feature extractor for both PN and MAML. For MAML, the sum of squares of a two-dimensional linear projection is used as the parametrization of the unsupervised cost. One can see that in this experiment, PN generally performs better in the fully supervised setting, while MAML is much more efficient in making use of unlabeled data. In fact, its performance was very close to the fully supervised case using true labels of the unlabeled samples.

3.4 EXPERIMENTS ON MINIIMAGENET

We tested the proposed method for semi-supervised adaptation with PN on the miniImagenet recognition task described in Section 2.9.

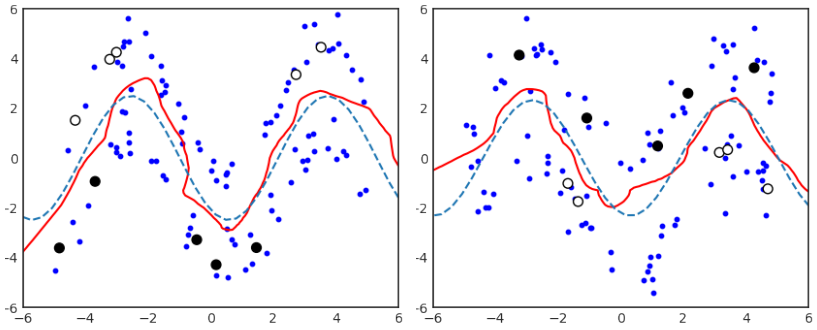


Figure 14: Examples of semi-supervised adaptation using MAML to two different test tasks with 10 labeled samples and 100 unlabeled samples. The blue dots correspond to unlabeled samples.

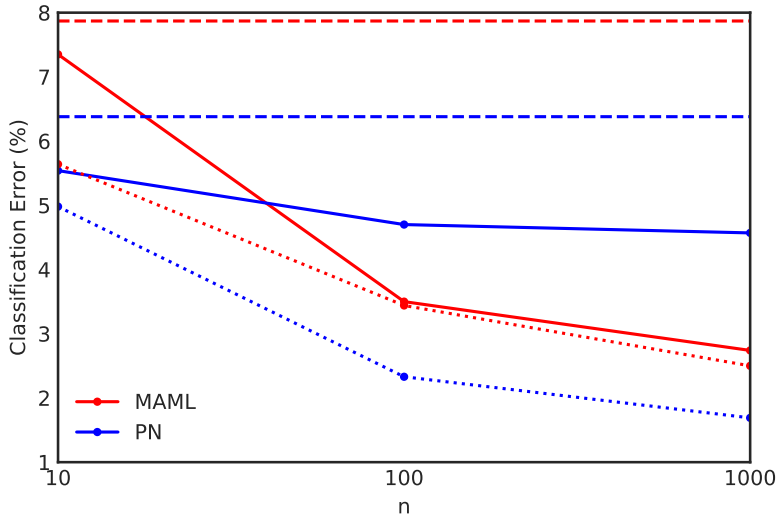


Figure 15: Semi-supervised adaptation results with MAML and PN on the sine classification dataset using 10 labeled plus n unlabeled samples. The dashed and dotted lines depicts the error rates of the (fully supervised) adapted model using 10 and $n + 10$ *labeled* samples respectively.

For semi-supervised adaptation, in addition to the k labeled samples from each class, we assume the existence of M unlabeled samples per class at test time. In Table 4, we show how the number of unlabeled examples at test time (M) affects the classification accuracy of the trained PN. The results indicate that increasing the number of unlabeled samples yields better performance, however, the improvement plateaus very quickly with the increase of the number of unlabeled samples. This agrees with the results obtained on the synthetic data reported in Section 3.3. Notably, the classification performance of the four-block architecture scales well with increasing the number of unlabeled samples, closely matching the performance of the ResNet in

the case of 120 unlabeled samples per class. The evolution of the classification accuracy with increasing the number of K-means iterations is shown in Table 5. Our proposed method for semi-supervised adaptation with MAML did not yield any improvements on this benchmark.

	M	1-SHOT	5-SHOT
PN (ours)	15	51.07 ± 0.90	65.52 ± 0.71
	30	53.41 ± 0.94	65.80 ± 0.67
	60	54.48 ± 0.98	65.94 ± 0.69
	120	55.23 ± 1.04	65.96 ± 0.68
Resnet PN	15	54.05 ± 0.47	70.92 ± 0.66
	30	54.70 ± 0.46	71.86 ± 0.59
	60	55.66 ± 0.46	72.21 ± 0.55
	120	55.67 ± 0.45	72.55 ± 0.52

Table 4: Average 5-way 1-shot classification accuracy (with 95% confidence intervals) on miniImagenet for the semi-supervised scenario for different number of unlabeled samples per class (M) available at test time.

		HARD K-MEANS	SOFT K-MEANS
PN (ours)	0	48.06 ± 0.82	48.06 ± 0.41
	1	50.13 ± 0.88	51.36 ± 0.75
	2	50.76 ± 0.93	46.53 ± 0.68
	10	51.07 ± 0.90	37.16 ± 0.34
Resnet PN	0	51.69 ± 0.42	51.69 ± 0.42
	1	53.42 ± 0.48	54.65 ± 0.45
	2	53.98 ± 0.50	50.62 ± 0.41
	10	54.05 ± 0.47	38.19 ± 0.42

Table 5: Average classification accuracy (with 95% confidence intervals) on miniImagenet for the semi-supervised scenario as a function of the number of K-means iterations. Each task consists of 1 labeled sample and 15 unlabeled samples.

3.5 CONCLUSION

In this chapter, we extended PN and MAML for semi-supervised adaptation. We evaluated semi-supervised adaptation with PN and MAML using synthetic data. Further, we evaluated semi-supervised adaptation with PN on the miniImagenet benchmark. While MAML was much more efficient in making use of unlabeled data in our experiments with synthetic data, it did not yield any improvements on the miniImagenet benchmark.

4 ACTIVE ADAPTATION

We perform active learning by first clustering and then requesting for labels

There are two sources of errors which the semi-supervised adaptation algorithm proposed in Chapter 3 can accumulate: 1) errors due to incorrect clustering of data, 2) errors due to incorrect labeling of the clusters. The second type of error can occur when the few labeled examples are outliers which end up closer to the prototype of another class in the embedding space. In this thesis, we advocate that the most practical way to correct the second type of errors can be through user feedback, since in many applications of the semi-supervised few-shot adaptation, interaction with the user is possible. This idea is inspired by the work of Cohn, Caruana, and McCallum [9] who introduced a clustering approach that allows a user to iteratively provide feedback to a clustering algorithm.

Consider the previously introduced example of few-shot learning in photo management applications. Although it is possible to ask the user to label a few photographs and use those labels to classify the rest of the pictures, it is extremely difficult and tiresome for the user to scroll through all the photos and decide which samples should be labeled. Instead, using the observation that "it is easier to criticize than to create" [9], one can initially cluster the photos and then request the user to label certain photos (or provide other types of feedback) so that the data is properly clustered and labeled.

"it is easier to criticize than to create"

4.1 ACTIVE ADAPTATION WITH PN

In this thesis, we assume that the user can provide feedback only in the form of labeling a particular sample or labeling the whole cluster. We propose to use PN as a feature extractor, cluster the samples in the embedding space using K-means and then label the clusters by requesting one labeled example for each cluster from the user. For each cluster c', we choose sample $\mathbf{z}_{c'}$ to be labeled by the user by maximizing an acquisition function $a(\mathbf{z}, c')$:

$$\mathbf{z}_{c'} = \max_{\mathbf{z} \in U_{c'}} a(\mathbf{z}, c'),$$

where $U_{c'}$ is the set of embedded inputs belonging to cluster c'. We explore a few acquisition functions:

- RANDOM: Sample a data point uniformly at random from each cluster. This is a baseline approach.

- NEAREST: Select the data point which is closest to the cluster center:
$$a(\mathbf{z}, c') = -d(\mathbf{z}, \mathbf{m}_{c'}),$$
where $\mathbf{m}_{c'}$ is the mean (cluster center) of cluster c'.

- ENTROPY: Select the sample with the least entropy:
$$a(\mathbf{z}, c') = \sum_{c} p(y = c|\mathbf{z}) \log p(y = c|\mathbf{z})$$
Thus, we select a sample with the least uncertainty that it belongs to a certain cluster.

- MARGIN: Select a sample with the largest margin between the most likely and second most likely labels.
$$a(\mathbf{z}, c') = p(y = c_1(\mathbf{z})|\mathbf{z}) - p(y = c_2(\mathbf{z})|\mathbf{z})$$
where $c_1(\mathbf{z})$ and $c_2(\mathbf{z})$ are the most likely and the second most likely clusters of embedded input $\mathbf{z}$ respectively. This quantity was proposed as a measure of uncertainty by Scheffer, Decomain, and Wrobel [43].

We also try to simulate a case when the user can label the whole cluster, as in some applications it can certainly be possible. This approach directly measures the clustering accuracy and we call it 'oracle'.

- ORACLE: We label each cluster based on the distance of the cluster mean to the prototypes computed from the true labels of all the samples.

4.2 EXPERIMENTS WITH MINIIMAGENET

The 1-shot classification accuracy with 120 unlabeled samples per class even surpassed the 5-shot accuracy of some well-recognized previous methods.

We test our proposed active learning approach using the miniImagent dataset. Similar to the semi-supervised scenario, we use a PN trained in the episodic mode as the feature extractor. We perform active learning on test tasks by first doing K-means clustering in the PN embedding space and then requesting one labeled example for each cluster using the acquisition functions described earlier. Note that multiple clusters can be labeled to the same class if the requested labels guide it that way. This is one of the largest source of error as shown in Figure 19). Table 6 and Table 7 presents the classification performance of each strategy for test tasks with one labeled sample and a varying number of unlabeled samples. There, we also present the accuracy of the oracle clustering. We use the same PN and Resnet PN feature extractors from the semi-supervised scenario. Overall, the MARGIN approach worked best in our experiments. The 1-shot classification

M	RANDOM	NEAREST	ENTROPY	MARGIN	ORACLE
15	49.19	54.42	53.95	56.12	58.96
30	49.23	54.73	56.02	57.58	60.27
60	50.73	56.12	57.63	59.24	62.09
120	50.74	57.45	57.88	61.42	63.23

Table 6: Average 1-shot classification accuracy (with 95% confidence intervals) of *PN (ours)* on miniImagenet for the active learning scenario for different number of unlabeled samples per class (M) available at test time.

M	RANDOM	NEAREST	ENTROPY	MARGIN	ORACLE
15	51.10	57.50	58.24	60.00	62.44
30	51.16	57.63	59.45	60.29	62.94
60	51.36	57.68	59.77	60.43	63.21
120	51.56	58.09	60.19	60.49	63.71

Table 7: Average 1-shot classification accuracy (with 95% confidence intervals) of *Resnet PN* on miniImagenet for the active learning scenario for different number of unlabeled samples per class (M) available at test time.

accuracy with 120 unlabeled samples per class even surpassed the 5-shot accuracy of some well-recognized previous methods. Similar to the semi-supervised scenario, the four-block architecture scales well with increasing the number of unlabeled samples closely matching the performance of the ResNet in the case of 120 unlabeled samples per class and even outperforming it while using the MARGIN strategy.

4.3 VISUALIZATIONS

In this section, we analyze the embeddings produced by the PN feature extractor. We produce two-dimensional visualizations by projecting the 384-dimensional features onto the principal subspace of the prototypes computed from the support set. The support set is represented using triangle markers, the query set using circle markers and the prototypes using star markers. The colors represent the true labels in the first column and the labels produced with different adaptation strategies in columns 2–5. Each row corresponds to one 3-way 1-shot task from miniImagenet.

Different 3-way 1-shot miniImagenet tasks are illustrated in Figures 16, 17, 18 and 19. The captions of the figures are described below:

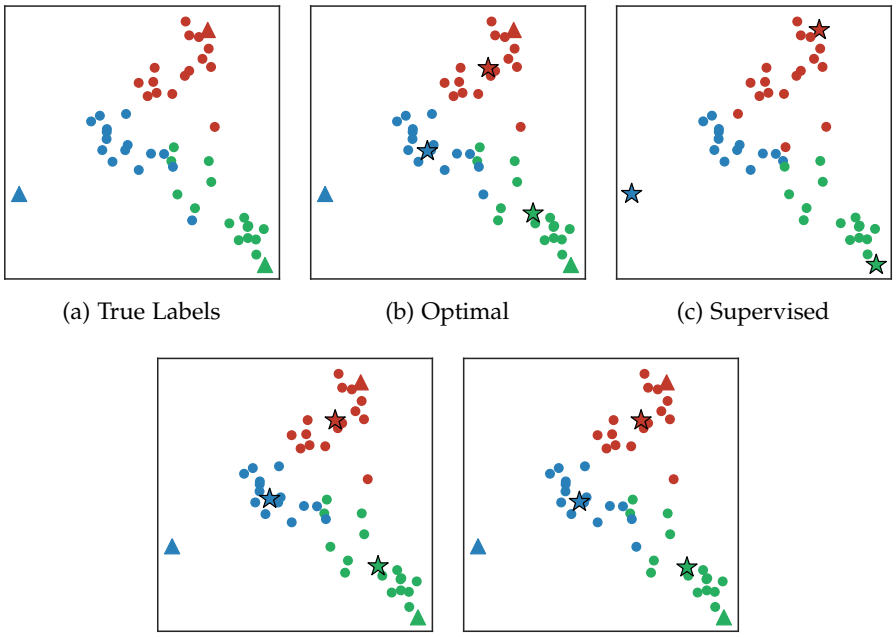
- *True Labels*: The true labels.

Number of Iterations:		1	2	3	10
PN (ours)	Random	47.29	47.74	48.08	49.19
	Nearest	52.98	53.31	53.89	53.92
	Entropy	52.51	53.80	53.95	54.42
	Margin	54.67	54.98	55.74	56.12
	Oracle	57.83	58.45	58.89	58.96
Resnet PN	Random	49.19	49.74	51.03	51.10
	Nearest	55.69	56.88	57.27	57.50
	Entropy	57.17	57.63	58.24	58.52
	Margin	57.92	58.68	59.11	60.00
	Oracle	61.36	62.32	62.39	62.44

Table 8: Average 1-shot classification accuracy (with 95% confidence intervals) on miniImagenet for the active learning scenario for different number of iterations. Each task consists of 1 labeled sample and 15 unlabeled samples.

- *Optimal*: Predictions based on the prototype computed from the true labels of all the samples.
- *Supervised*: Predictions based on the prototypes computed from the support set.
- *Semi-supervised*: Predictions based on the prototypes (or cluster means) computed after K-means clustering seeded by the prototypes of the support set.
- *Active*: Predictions based on the prototypes (or cluster means) computed by K-means and labeling the clusters using the NEAREST approach.

In Figure 16, the samples in the task are reasonably well clustered and supervised adaptation performs quite well. The labeling is further slightly improved by semi-supervised and active adaptation. In Figure 17, the support sample of the red class is an outlier and using it as a prototype leads to misclassifications. Even seeding the clustering with these prototypes (semi-supervised classification) leads to incorrect clustering. However, the active adaptation is able to find a reasonable solution. Figure 18 illustrates a failure case of the NEAREST approach of the active adaptation where the samples are properly clustered but incorrectly labeled. In Figure 19, the samples are not properly clustered in the feature space. The supervised and semi-supervised approaches fail badly, while the active approach produces somewhat usable results.



(a) True Labels (b) Optimal (c) Supervised

(d) Semi-supervised (e) Active

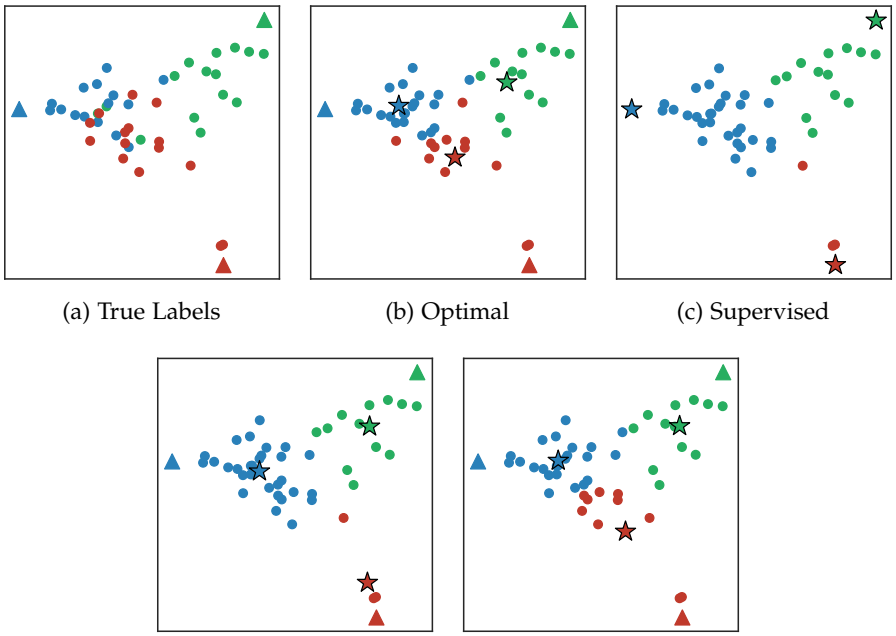
Figure 16: Illustration of a reasonably well clustered 3-way 1-shot miniImagenet task.

4.4 CONCLUSION

In this chapter, we advocated that in many real-world applications the semi-supervised adaptation performance can be significantly improved by requesting the few labels through user feedback. We demonstrate good performance of the active adaptation strategy with PN using image data.

The user can provide feedback in various forms and therefore can effectively introduce various constraints that can further guide the clustering process. For example, a user can assign the whole cluster to a particular class, assign a sample to a particular cluster, mark that a particular sample does not belong to the assigned cluster, split and combine clusters. These constraints could be easily induced in basic clustering algorithms such as K-means. For examples, Wagstaff et al. [52] introduced constraints between samples in the data set such as must-link (two samples have to be in the same cluster) and cannot-link (two samples have to be in different clusters) and the clustering algorithm finds a solution that satisfies all the constraints.

Even outside the context of few-shot learning, this active learning approach can be used to adapt a pre-trained classifier. Assume that we have a classifier that clusters the classes of a particular classification task such as ImageNet. Then, during test time it is possible to interactively split clusters to make coarse-grained classifications or to



(a) True Labels (b) Optimal (c) Supervised

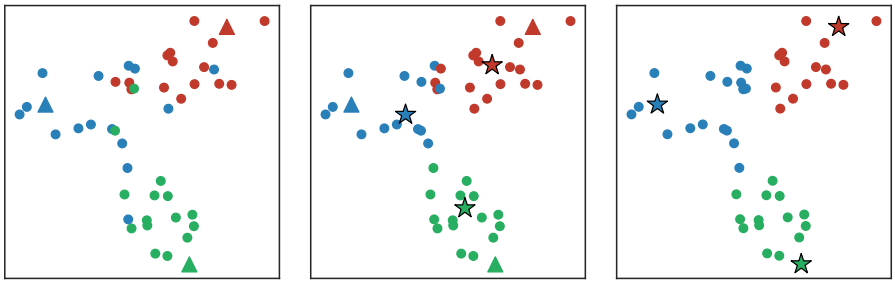
(d) Semi-supervised (e) Active

Figure 17: Illustration of a 3-way 1-shot miniImagenet task in which the supervised and semi-supervised adaptations fail but active adaptation produces reasonable results.

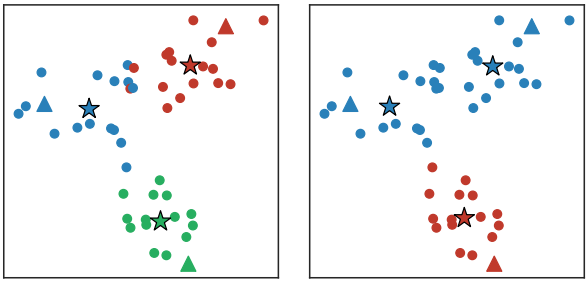
assign multiple clusters to a super-cluster (to make hierarchical predictions).

The fundamental bottleneck of the proposed approach is the ability of the feature extractor to cluster unseen *data*

The fundamental bottleneck of the proposed approach in improving the classification performance is the ability of the feature extractor to cluster *unseen* data. Although we used an embedding network trained using Prototypical Networks, the adaptation mechanisms proposed in this thesis can be performed using other feature extractors as well. A feature extractor explicitly trained to cluster data can further improve the few-shot classification performance and this is an area of active research [28, 47, 48]. Building feature extractors that allow better generalization is largely a unsolved problem and it requires further exploration [see, e.g., 17, 41].

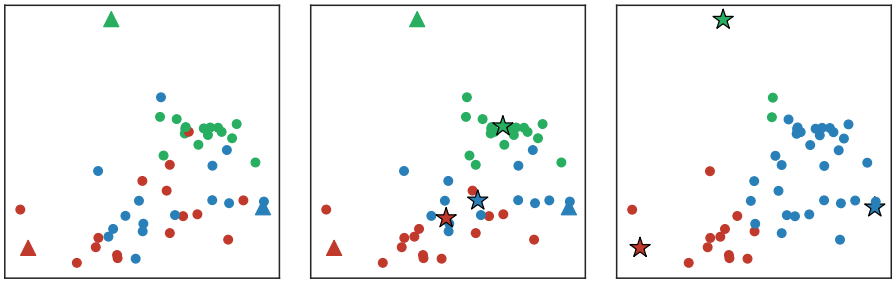


(a) True Labels (b) Optimal (c) Supervised

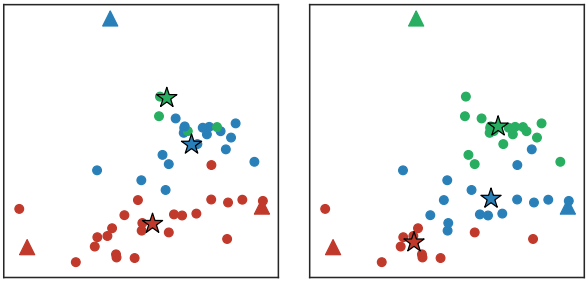


(d) Semi-supervised (e) Active

Figure 18: Illustration of a failure case of active adaptation in a 3-way 1-shot miniImagenet task.



(a) True Labels (b) Optimal (c) Supervised



(d) Semi-supervised (e) Active

Figure 19: Illustration of a 3-way 1-shot miniImagenet task in which the samples are not properly clustered.

5 UNSUPERVISED ADAPTATION

We explore unsupervised adaptation to classification tasks with the same output space

In many practical applications, the classification tasks to which a classifier needs to adapt share the same output space. The classifier needs to adapt to different input distributions (*domain adaptation*). Since the output space remains constant, the adaptation to changes in the input distributions can be performed in an unsupervised fashion.

Consider a material recognition system deployed in multiple factories. The same categories of materials need to be recognized at different sites, however, factories may have slightly different lighting conditions or other factors affecting the recognition process, motivating the need for adaptation of each recognition system.

As an additional example, consider a vehicle detection system to be deployed in different kinds of challenging environments in different parts of the world. Since data annotation is a laborious process, it is only possible to obtain labeled data from a few diverse conditions. Deploying the system in different environmental conditions such as snow or rain that was not present during training is challenging and demands adaptation.

5.1 UNSUPERVISED ADAPTATION WITH PROTOTYPICAL NETWORKS

We propose to perform unsupervised adaptation with PN as follows: after the standard PN training procedure, we compute the average of the prototypes of the corresponding classes from all the training tasks. It is to be noted that the classes are consistent across all the tasks. At test time, we cluster the samples using K-means and then simply assign each cluster to the class of the closest prototype. This is possible since the feature extractor becomes invariant to certain changes in the input distribution and produces embeddings that are clustered according to the classes in the training set.

5.2 UNSUPERVISED ADAPTATION WITH MAML

Similar to semi-supervised adaptation, we perform classification in two steps:

1. Inputs $\mathbf{x}$ are transformed into features $\mathbf{z}$ using function $\mathbf{z} = f_\theta(\mathbf{x})$ with parameters θ;
2. Predicted labels are computed using function $\hat{y} = g_\omega(\mathbf{z})$ with parameters ω.

The difference to the semi-supervised case is that the parameters ω of the classifier g_ω are not adapted due to the lack of labeled data.

The proposed method for unsupervised adaptation using MAML is as follows: We use unlabeled data to adapt the feature extractor parameters θ using Equation 5 with an unsupervised cost function C:

$$\theta_t = \theta_* - \alpha \sum_{x_j \in S_t} \nabla_\theta C\left(f_\theta(x_j)\right)\Big|_{\theta=\theta_*}, \tag{11}$$

where similar to semi-supervised adaptation, C is parametrized with a neural network $C_{\omega_c}(z)$ with parameters ω_c and learnt in the same meta-training procedure. The classifier g is not adapted and has the same output space for all tasks. Similar to supervised adaptation, the parameters are tuned to optimize the performance on the query sets Q_t:

$$\min_{\theta,\alpha,\omega_c,\omega} \sum_{t\sim p(t)} \sum_{(x_j,y_j)\in Q_t} L\left(g_\omega(f_{\theta_t}(x_j)), y_j\right), \tag{12}$$

the tuned parameters are θ, α, ω_c and ω. The full computational graph for semi-supervised adaptation is illustrated in Figure 20. This approach is similar to the two-head architecture introduced by Finn et al. [13].

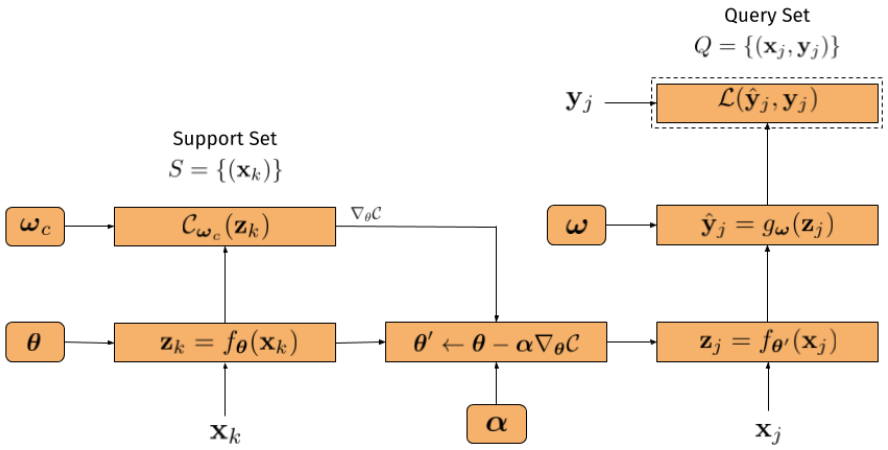


Figure 20: Computational graph of unsupervised adaptation with MAML. Parameters are represented by rounded rectangles and computations are represented by rectangles. The loss function used for training is highlighted in a dashed rectangle.

5.3 EXPERIMENTS ON SYNTHETIC DATA

We experimented with the proposed unsupervised adaptation methods on the synthetic sine classification dataset introduced in Section 3.3. Examples of the decision boundaries produced by PN on

two different unsupervised test tasks are shown in Figure 21. Examples of the decision boundaries produced by MAML on two test tasks are shown in Figure 22. Note that both methods are able to adapt despite the lack of labeled examples and the adapted decision boundaries still resemble a sine wave, closely matching the true decision boundary.

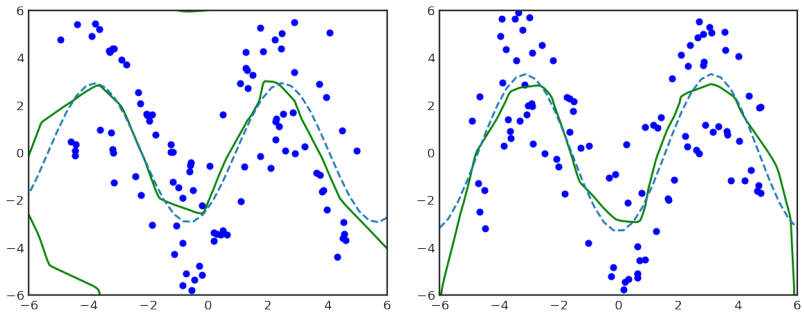


Figure 21: Examples of unsupervised adaptation using PN to two different test tasks with 100 unlabeled samples. The blue dots correspond to unlabeled samples.

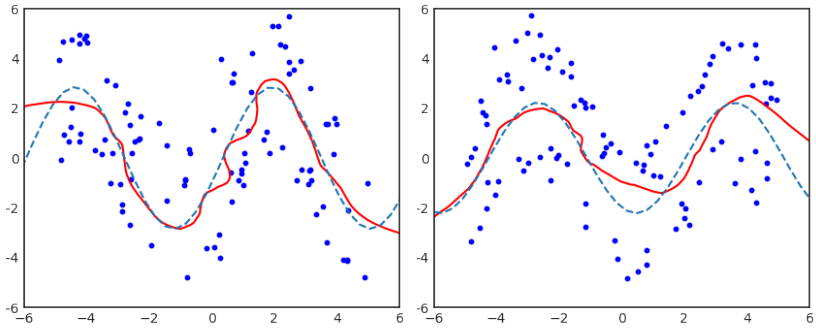


Figure 22: Examples of unsupervised adaptation using MAML to two different test tasks with 100 unlabeled samples. The blue dots correspond to unlabeled samples.

Both PN and MAML are able to make use of unlabeled data. MAML is much more efficient.

The classification accuracy of unsupervised adaptation with PN and MAML are shown in Figure 23. Similar to the semi-supervised case, we use a fully-connected network with two hidden layers of size 100 with ReLU non-linearity as the feature extractor for both PN and MAML. For MAML, the sum of squares of a two-dimensional linear projection is used as the parametrization of the unsupervised cost. It can be seen that both PN and MAML are able to make use of unlabeled data. However, MAML is much more efficient in making use of the unlabeled data. In fact, its performance was very close to the fully supervised case (using true labels of the unlabeled samples) with increasing number of unlabeled samples. This means that MAML is able to effectively label the unlabeled data.

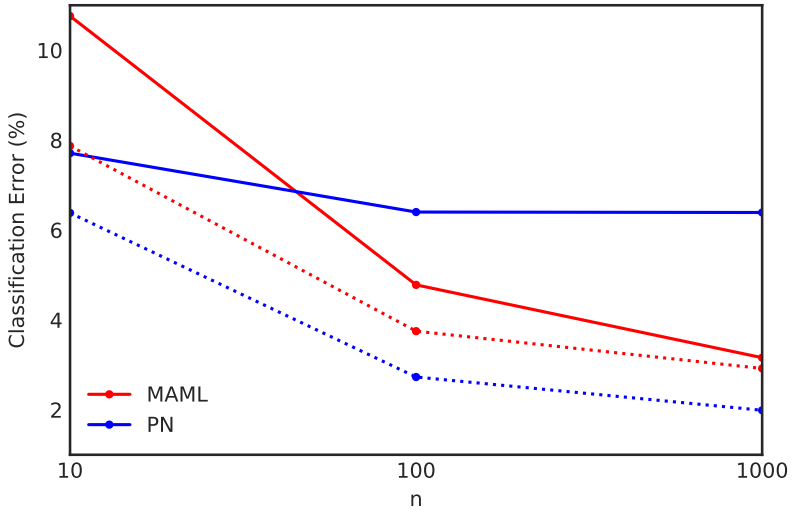


Figure 23: Unsupervised adaptation results with MAML and PN on the sine classification dataset. The dotted lines depicts the accuracy of the adapted model using the same number n of *labeled* samples.

5.4 CONCLUSION

In this chapter, we considered the problem of adaptation to new tasks without any labeled examples (unsupervised adaptation) when the new task has the same output space (classes) as the training tasks do. We proposed extensions to PN and MAML to perform unsupervised adaptation. Using synthetic data, we showed that both methods are able to adapt successfully by making use of the unlabeled data while MAML was much more efficient in making use of the unlabeled data.

6 CONCLUSION

In this thesis, we build upon two existing methods for few-shot adaptation – Prototypical Networks and Model-Agnostic Meta-Learning (MAML). We extended PN to adapt to new classification tasks in the semi-supervised few-shot learning scenario when a few labeled examples are accompanied with many unlabeled examples from the same classes. We proposed to use the clustering approach to semi-supervised classification when the clustering process is guided by the labeled examples. We also advocated that in many real-world applications it can be possible to request the few labeled examples from the user, which can yield better performance. We identify that the fundamental bottleneck in our proposed method is the ability of the feature extractor to cluster *unseen* data. Also, we proposed an extension of MAML to the cases of unsupervised and semi-supervised few-shot adaptation. Using a synthetic dataset, we show that MAML can be more efficient in using unlabeled data compared to PN. We continue investigating the possibility of combining the good properties of MAML and PN in a single adaptation scheme. Further, we make a comparison of MAML and PN under various criteria.

PN is simpler and more intuitive than MAML. PN is essentially a deep equivalent of a Nearest Class Mean (NCM) classifier. It is possible to easily induce domain specific priors into PN by modifying the feature extractor or the distance function. It is unclear how to achieve this with MAML.

In fully-supervised adaptation, we find that PN generally works better than MAML. The performance of MAML improves well with increasing number of shots (labeled samples per class). In contrast, Snell, Swersky, and Zemel [46] showed that the feature extractor of PN is sensitive to the number of shots used during training. To reduce this sensitivity, we proposed to use a varying number of samples per class during training for better generalization to the number of shots during test time in Section 2.10.

In our experiments on synthetic data, MAML adapts with close to optimal performance under the availability of increased unlabeled data. Our proposed method for unsupervised and semi-supervised adaptation with MAML makes effective use of unlabeled data in our experiments with synthetic data. However, while PN is able to make use of unlabeled data, it is not optimal. We showed how to perform active adaptation with PN by taking into account feedback from the user. With PN, active adaptation yields noticeable improvements in

the classification accuracy over semi-supervised adaptation. It is unclear how to extend MAML to perform active adaptation.

MAML is able to adapt the features depending on the task. The feature extractor in PN remains the same after training. MAML has been tested in different domains such as image classification and reinforcement learning [12]. On the other hand, PN has only been tested in few-shot image classification tasks. MAML is applicable to any differentiable model and has been used in classification and regression tasks in a straightforward manner. It is unclear how to extend PN to regression tasks.

MAML 'forgets' the previous tasks and is only adapted to one task at a time. Since the feature extractor of PN remains the same, it can be easily extended to perform continual learning. We explore this possibility in Section 2.10. Since MAML adapts the parameters of the model to different tasks, it is required to store a different copy of the parameters for each task. The number of parameters of a deep neural network is very large. A typical image classification network contains millions of parameters. Besides these expensive memory requirements, MAML requires computation of second order gradients for training (However, Finn, Abbeel, and Levine [12] showed that a first order approximation works as well on the miniImagenet benchmark). In contrast, PN only has modest memory requirements.

BIBLIOGRAPHY

[1] Charu C Aggarwal and ChengXiang Zhai. "A survey of text clustering algorithms." In: *Mining text data*. Springer, 2012, pp. 77–128.

[2] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. "Low data drug discovery with one-shot learning." In: *ACS central science* 3.4 (2017), pp. 283–293.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. "Learning to learn by gradient descent by gradient descent." In: *Advances in Neural Information Processing Systems*. 2016, pp. 3981–3989.

[4] Sugato Basu, Arindam Banerjee, and Raymond Mooney. "Semi-supervised clustering by seeding." In: *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer. 2002.

[5] Aurélien Bellet, Amaury Habrard, and Marc Sebban. "A survey on metric learning for feature vectors and structured data." In: *arXiv preprint arXiv:1306.6709* (2013).

[6] Irving Biederman. "Recognition-by-components: a theory of human image understanding." In: *Psychological review* 94.2 (1987), p. 115.

[7] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]." In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." In: *arXiv preprint arXiv:1606.00915* (2016).

[9] David Cohn, Rich Caruana, and Andrew McCallum. "Semi-supervised clustering with user feedback." In: *Constrained Clustering: Advances in Algorithms, Theory, and Applications* 4.1 (2003), pp. 17–32.

[10] Harrison Edwards and Amos Storkey. "Towards a neural statistician." In: *arXiv preprint arXiv:1606.02185* (2016).

[11] Li Fei-Fei, Rob Fergus, and Pietro Perona. "One-shot learning of object categories." In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *International Conference on Machine Learning*. 2017, pp. 1126–1135.

[13] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. "One-Shot Visual Imitation Learning via Meta-Learning." In: *arXiv preprint arXiv:1709.04905* (2017).

[14] Yves Grandvalet and Yoshua Bengio. "Semi-supervised learning by entropy minimization." In: *Advances in neural information processing systems*. 2005, pp. 529–536.

[15] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes." In: *arXiv preprint arXiv:1801.08930* (2018).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[17] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing." In: *International Conference on Learning Representations* (2018).

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[19] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In: *arXiv preprint arXiv:1709.01507* (2017).

[20] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International conference on machine learning*. 2015, pp. 448–456.

[21] Liping Jing. "Survey of text clustering." In: *Department of Mathematics, The University of Hong Kong, HongKong, China, ISBN* (2008), pp. 7695–1754.

[22] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[23] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. "Overcoming catastrophic forgetting in neural networks." In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526.

[24] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In: *ICML Deep Learning Workshop*. Vol. 2. 2015.

[25] Brian Kulis et al. “Metric learning: A survey.” In: *Foundations and Trends in Machine Learning* 5.4 (2013), pp. 287–364.

[26] Samuli Laine and Timo Aila. “Temporal Ensembling for Semi-Supervised Learning.” In: *arXiv preprint arXiv:1610.02242* (2016).

[27] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. “Human-level concept learning through probabilistic program induction.” In: *Science* 350.6266 (2015), pp. 1332–1338.

[28] Marc T Law, Raquel Urtasun, and Richard S Zemel. “Deep spectral clustering learning.” In: *International Conference on Machine Learning*. 2017, pp. 1985–1994.

[29] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. “Meta-SGD: Learning to Learn Quickly for Few Shot Learning.” In: *arXiv preprint arXiv:1707.09835* (2017).

[30] Stuart Lloyd. “Least squares quantization in PCM.” In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[31] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. “Distance-based image classification: Generalizing to new classes at near-zero cost.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2624–2637.

[32] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. “Meta-Learning with Temporal Convolutions.” In: *arXiv preprint arXiv:1707.03141* (2017).

[33] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. “Distributional smoothing with virtual adversarial training.” In: *arXiv preprint arXiv:1507.00677* (2015).

[34] Tsendsuren Munkhdalai and Hong Yu. “Meta Networks.” In: *arXiv preprint arXiv:1703.00837* (2017).

[35] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. “Semi-supervised learning with Ladder networks.” In: *Advances in Neural Information Processing Systems*. 2015, pp. 3546–3554.

[36] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning.” In: *International Conference on Learning Representations*. 2017.

[37] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. “Meta-Learning for Semi-Supervised Few-Shot Classification.” In: *International Conference on Learning Representations*. 2018.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[39] Danilo Rezende, Ivo Danihelka, Karol Gregor, Daan Wierstra, et al. "One-shot generalization in deep generative models." In: *International Conference on Machine Learning*. 2016, pp. 1521–1529.

[40] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[41] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules." In: *Advances in Neural Information Processing Systems*. 2017, pp. 3859–3869.

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

[43] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. "Active hidden Markov models for information extraction." In: *International Symposium on Intelligent Data Analysis*. Springer. 2001, pp. 309–318.

[44] Jürgen Schmidhuber. "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook." PhD thesis. Technische Universität München, 1987.

[45] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. "Attentive Recurrent Comparators." In: *arXiv preprint arXiv: 1703.00767* (2017).

[46] Jake Snell, Kevin Swersky, and Richard S Zemel. "Prototypical Networks for Few-shot Learning." In: *arXiv preprint arXiv:1703.05175* (2017).

[47] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep metric learning via lifted structured feature embedding." In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, pp. 4004–4012.

[48] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. "Deep metric learning via facility location." In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.

[49] Antti Tarvainen and Harri Valpola. "Weight-averaged consistency targets improve semi-supervised deep learning results." In: *arXiv preprint arXiv:1703.01780* (2017).

[50] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[51] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. "Matching networks for one shot learning." In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[52] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. "Constrained K-means clustering with background knowledge." In: *ICML*. Vol. 1. 2001, pp. 577–584.

[53] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." In: *arXiv preprint arXiv:1605.07146* (2016).