META-LEARNING FOR SEMI-SUPERVISED FEW-SHOT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In few-shot classification, we are interested in learning algorithms that train a classifier from only a handful of labeled examples. Recent progress in few-shot classification has featured meta-learning, in which a parameterized model for a learning algorithm is defined and trained on episodes representing different classification problems, each with a small labeled training set and its corresponding test set. In this work, we advance this few-shot classification paradigm towards a scenario where unlabeled examples are also available within each episode. We consider two situations: one where all unlabeled examples are assumed to belong to the same set of classes as the labeled examples of the episode, as well as the more challenging situation where examples from other *distractor* classes are also provided. To address this paradigm, we propose novel extensions of Prototypical Networks (Snell et al., 2017) that are augmented with the ability to use unlabeled examples when producing prototypes. These models are trained in an end-to-end way on episodes, to learn to leverage the unlabeled examples successfully. We evaluate these methods on versions of the Omniglot and *mini*ImageNet benchmarks, adapted to this new framework augmented with unlabeled examples. We also propose a new split of ImageNet, consisting of a large set of classes, with a hierarchical structure. Our experiments confirm that our Prototypical Networks can learn to improve their predictions due to unlabeled examples, much like a semi-supervised algorithm would.

1 INTRODUCTION

The availability of large quantities of labeled data has enabled deep learning methods to achieve impressive breakthroughs in several tasks related to artificial intelligence, such as speech recognition, object recognition and machine translation. However, current deep learning approaches struggle in tackling problems for which labeled data are scarce. Specifically, while current methods excel at tackling a single problem with lots of labeled data, methods that can simultaneously solve a large variety of problems that each have only a few labels are lacking. Humans on the other hand are readily able to rapidly learn new classes, such as new types of fruit when we visit a tropical country. This significant gap between human and machine learning provides fertile ground for deep learning developments.

For this reason, recently there has been an increasing body of work on few-shot learning, which considers the design of learning algorithms that specifically allow for better generalization on problems with small labeled training sets. Here we focus on the case of few-shot classification, where the given classification problem is assumed to contain only a handful of labeled examples per class. One approach to few-shot learning follows a form of meta-learning[1] (Thrun, 1998; Hochreiter et al., 2001), which performs transfer learning from a pool of various classification problems generated from large quantities of available labeled data, to new classification problems from classes unseen at training time. Meta-learning may take the form of learning a shared metric (Vinyals et al., 2016; Snell et al., 2017), a common initialization for few-shot classifiers (Ravi & Larochelle, 2017; Finn et al., 2017) or a generic inference network (Santoro et al., 2016; Mishra et al., 2017).

[1]See the following blog post for an overview: `http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/`

These various meta-learning formulations have led to significant progress recently in few-shot classification. However, this progress has been in a limited scenario, which differs in many dimensions from how humans learn new concepts. In this paper we aim to generalize the few-shot setting in two ways. First we consider a scenario in which the new classes are learned in the presence of additional unlabeled data. While there has been many successful applications of semi-supervised learning to the regular setting of a single classification task (Chapelle et al., 2010) where classes at training and test time are the same, such work has not addressed the challenge of performing transfer to new classes never seen at training time, as we consider here. Second, we consider the situation where the new classes to be learned are not viewed in isolation. Instead, many of the unlabeled examples are from different classes; the presence of such *distractor* classes introduces an additional and more realistic level of difficulty to the few-shot problem.

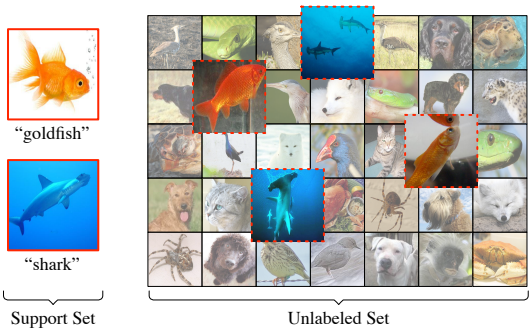

Figure 1: Consider a setup where the aim is to learn a classifier to distinguish between two previously unseen classes, goldfish and shark, given not only labeled examples of these two classes, but also a larger pool of unlabeled examples, some of which may belong to one of these two classes of interest. In this work we aim to move a step closer to this more natural learning framework by incorporating in our learning episodes unlabeled data from the classes we aim to learn representations for (shown with dashed red borders) as well as from *distractor* classes .

This work is a first study of this challenging semi-supervised form of few-shot learning. First, we define the problem and propose benchmarks for evaluation that are adapted from the Omniglot and *mini*ImageNet benchmarks used in ordinary few-shot learning. We perform an extensive empirical investigation of the two settings mentioned above, with and without distractor classes. Second, we propose and study three novel extensions of Prototypical Networks (Snell et al., 2017), a state-of-the-art approach to few-shot learning, to the semi-supervised setting. Finally, we demonstrate in our experiments that our semi-supervised variants successfully learn to leverage unlabeled examples and outperform purely supervised Prototypical Networks.

2 Background

We start by defining precisely the current paradigm for few-shot learning and the Prototypical Network approach to this problem.

2.1 Few-shot learning

Recent progress on few-shot learning has been made possible by following an episodic paradigm for few-shot learning. Consider a situation where we have a large labeled dataset for a set of classes $\mathcal{C}_{\text{train}}$. However, after training on examples from $\mathcal{C}_{\text{train}}$, our ultimate goal is to produce classifiers for a disjoint set of new classes $\mathcal{C}_{\text{test}}$, for which only a few labeled examples will be available. The idea behind the episodic paradigm is to simulate the types of few-shot problems that will be encountered at test using the large quantities of available labeled data for classes $\mathcal{C}_{\text{train}}$.

Specifically, models are trained on K-shot, N-way episodes constructed by first sampling a small subset of N classes from $\mathcal{C}_{\text{train}}$ and then generating: 1) a training (support) set $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_{N\times K}, y_{N\times K})\}$ containing K examples from each of the N classes and 2) a test (query) set $\mathcal{Q} = \{(\boldsymbol{x}^*_1, y^*_1), (\boldsymbol{x}^*_2, y^*_2), \ldots, (\boldsymbol{x}^*_T, y^*_T)\}$ of different examples from the same N classes. Each $\boldsymbol{x}_i \in \mathbb{R}^D$ is an input vector of dimension D and $y_i \in \{1, 2, \ldots, N\}$ is a class label (similarly for $\boldsymbol{x}^*_i$ and y^*_i). Training on such episodes is done by feeding the support set $\mathcal{S}$ to the model and updating its parameters to minimize the loss of its predictions for the examples in the query set $\mathcal{Q}$.

One way to think of this approach is that our model effectively trains to be a good learning algorithm. Indeed, much like a learning algorithm, the model must take in a set of labeled examples and produce a predictor that can be applied to new examples. Moreover, training directly encourages the classifier

produced by the model to have good generalization on the new examples of the query set. Due to this analogy, training under this paradigm is often referred to as learning to learn or meta-learning.

On the other hand, referring to the content of episodes as training and test sets and to the process of learning on these episodes as meta-learning or meta-training (as is sometimes done in the literature) can be confusing. So for the sake of clarity, we will refer to the content of episodes as support and query sets, and to the process of iterating over the training episodes simply as training.

2.2 Prototypical Networks

Prototypical Network (Snell et al., 2017) is a few-shot learning model that has the virtue both of being simple while obtaining state-of-the-art performance. At a high-level, it uses the support set $\mathcal{S}$ to extract a prototype vector from each class, and classifies the inputs in the query set based on their distance to the prototype of each class.

More precisely, Prototypical Networks learn an embedding function $h(\boldsymbol{x})$, parameterized as a neural network, that maps examples into a space where examples from the same class are close and those from different classes are far. All parameters of Prototypical Networks lie in the embedding function.

To compute the prototype $\boldsymbol{p}_c$ of each class c, a per-class average of the embedded examples is performed:

$$\boldsymbol{p}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c}}{\sum_i z_{i,c}}, \text{ where } z_{i,c} = \mathbb{1}[y_i = c]. \tag{1}$$

These prototypes define a predictor for the class of any new (query) example $\boldsymbol{x}^*$, which assigns a probability over any class c based on the distances between $\boldsymbol{x}^*$ and each prototype, as follows:

$$p(c|\boldsymbol{x}^*, \{\boldsymbol{p}_c\}) = \frac{\exp(-||h(\boldsymbol{x}^*) - \boldsymbol{p}_c||_2^2)}{\sum_{c'} \exp(-||h(\boldsymbol{x}^*) - \boldsymbol{p}_{c'}||_2^2)} \ . \tag{2}$$

The loss function used to update Prototypical Networks for a given training episode is then simply the average negative log-probability of the correct class assignments, for all query examples:

$$-\frac{1}{T} \sum_i \log p(y_i^* | \boldsymbol{x}_i^*, \{\boldsymbol{p}_c\}) \ . \tag{3}$$

Training proceeds by minimizing the average loss, iterating over training episodes and performing a gradient descent update for each.

Generalization performance is measured on test set episodes, which contain images from classes in $\mathcal{C}_{\text{test}}$ instead of $\mathcal{C}_{\text{train}}$. For each test episode, we use the predictor produced by the Prototypical Network for the provided support set $\mathcal{S}$ to classify each of query input $\boldsymbol{x}^*$ into the most likely class $\hat{y} = \text{argmax}_c \, p(c|\boldsymbol{x}^*, \{\boldsymbol{p}_c\})$.

3 Semi-Supervised Few-Shot Learning

We now move to defining the semi-supervised setting considered in this work for few-shot learning.

We denote our training set as a tuple of labeled and unlabeled examples: $(\mathcal{S}, \mathcal{R})$. The labeled portion is the usual support set $\mathcal{S}$ of the few-shot learning literature, containing a list of tuples of inputs and targets. In addition to classic few-shot learning, we introduce an unlabeled set $\mathcal{R}$ containing only inputs: $\mathcal{R} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_M\}$. As in the purely supervised setting, our models are trained to perform well when predicting the labels for the examples in the episode's query set $\mathcal{Q}$. Figure 2 shows a visualization of training and test episodes.

3.1 Semi-Supervised Prototypical Networks

In their original formulation, Prototypical Networks do not specify a way to leverage the unlabeled set $\mathcal{R}$. In what follows, we now propose various extensions that start from the basic definition of prototypes $\boldsymbol{p}_c$ and provide a procedure for producing refined prototypes $\tilde{\boldsymbol{p}}_c$ using the unlabeled examples in $\mathcal{R}$.

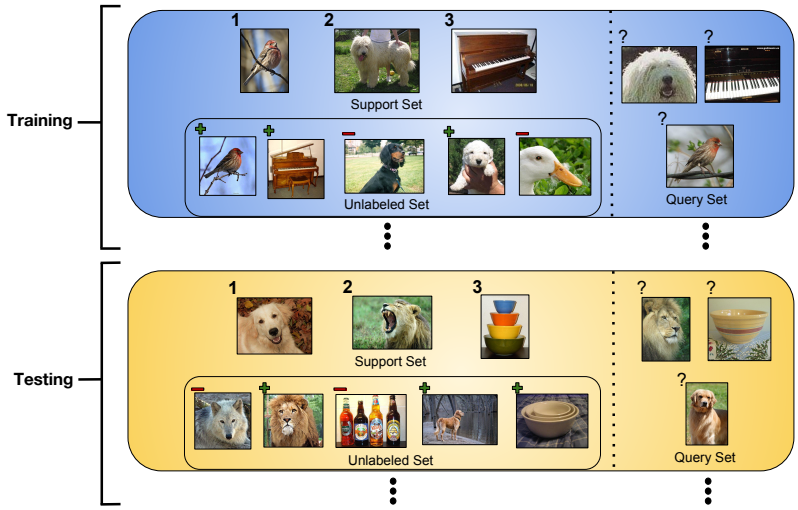

Figure 2: Example of the semi-supervised few-shot learning setup. Training involves iterating through training episodes, consisting of a support set $\mathcal{S}$, an unlabeled set $\mathcal{R}$, and a query set $\mathcal{Q}$. The goal is to use the labeled items (shown with their numeric label) in $\mathcal{S}$ and the unlabeled items in $\mathcal{R}$ within each episode to generalize to good performance on the corresponding query set. The unlabeled items in $\mathcal{R}$ may either be pertinent to the classes we are considering (shown above with green plus signs) or they may be *distractor* items which belong to a class that is not relevant to the current episode (shown with red minus signs). However note that the model does not actually have ground truth information as to whether each unlabeled example is a distractor or not; the plus/minus signs are shown only for illustrative purposes. At test time, we are given new episodes consisting of new classes not seen during training that we use to evaluate the meta-learning method.

After the refined prototypes are obtained, each of these models is trained with the same loss function for ordinary Prototypical Networks of Equation 3, but replacing $\boldsymbol{p}_c$ with $\tilde{\boldsymbol{p}}_c$. That is, each query example is classified into one of the N classes based on the closeness of its embedded position with the corresponding *refined* prototypes, and the average negative log-probability of the correct classification is used for training.

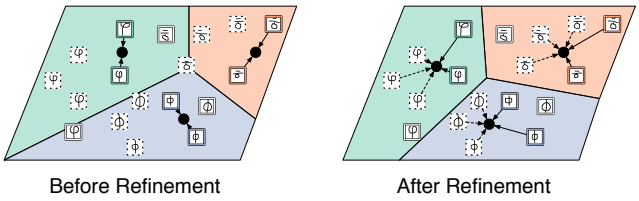

Figure 3: Left: The initialization of the prototypes to be the average of the examples of the corresponding class, as in ordinary Prototypical Networks. Support examples have solid colored borders, unlabeled examples have dashed borders, and query examples have white borders. Right: The refined prototypes obtained by incorporating the unlabeled examples. After refinement, all query examples are correctly classified.

3.1.1 Prototypical Networks with Soft k-Means

We first consider a simple way of leveraging unlabeled examples for refining prototypes, by taking inspiration from semi-supervised clustering. Viewing each prototype as a cluster center, the refinement process could attempt to adjust the cluster locations to better fit the examples in both the support and unlabeled sets. Under this view, cluster assignments of the labeled examples in the support set are considered known and fixed to each example's label. The refinement process must instead estimate the cluster assignments of the unlabeled example and adjust the cluster locations (the prototypes) accordingly.

One natural choice would be to borrow from the inference performed by soft k-means. We prefer this version of k-means over hard assignments since hard assignments would make the inference non-differentiable. We start from the regular Prototypical Network's prototypes $\boldsymbol{p}_c$ (as specified in Equation 1) as the cluster locations. Then, the unlabeled examples get a partial assignment ($\tilde{z}_{j,c}$) to each cluster based on their Euclidean distance to the cluster locations. Finally, refined prototypes are obtained by incorporating these unlabeled examples.

This process can be summarized as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j) \tilde{z}_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c}}, \text{ where } \tilde{z}_{j,c} = \frac{\exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2\right)}{\sum_{c'} \exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_{c'}||_2^2\right)} \tag{4}$$

Predictions of each query input's class is then modeled as in Equation 2, but using the refined prototypes $\tilde{\boldsymbol{p}}_c$.

We could perform several iterations of refinement, as is usual in k-means. However, we have experimented with various number of iterations and found results to not improve beyond a single refinement step.

3.1.2 Prototypical Networks with Soft k-Means with a Distractor Cluster

The soft k-means approach described above implicitly assumes that each unlabeled example belongs to either one of the N classes in the episode. However, it would be much more general to not make that assumption and have a model robust to the existence of examples from other classes, which we refer to as distractor classes. For example, such a situation would arise if we wanted to classify between pictures of unicycles and scooters, and decided to add an unlabeled set by downloading images from the web. It then would not be realistic to assume that all these images are of unicycles or scooters. Even with a focused search, some may be from similar categories, such as bicycle.

Since soft k-means distributes its soft assignments across all classes, distractor items could be harmful and interfere with the refinement process, as prototypes would be adjusted to also partially account for these distractors. A simple way to address this is to add an additional cluster whose purpose is to capture the distractors, thus preventing them from polluting the clusters of the classes of interest:

$$\boldsymbol{p}_c = \begin{cases} \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c}}{\sum_i z_{i,c}} & \text{for } c = 1 \ldots N \\ \boldsymbol{0} & \text{for } c = N+1 \end{cases} \tag{5}$$

Here we take the simplifying assumption that the distractor cluster has a prototype centered at the origin. We also consider introducing length-scales r_c to represent variations in the within-cluster distances, specifically for the distractor cluster:

$$\tilde{z}_{j,c} = \frac{\exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_c||_2^2 - A(r_c)\right)}{\sum_{c'} \exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_{c'}||_2^2 - A(r_{c'})\right)}, \text{ where } A(r) = \frac{1}{2}\log(2\pi) + \log(r) \tag{6}$$

For simplicity, we set $r_{1 \ldots N}$ to 1 in our experiments, and only learn the length-scale of the distractor cluster r_{N+1}.

3.1.3 Prototypical Networks with Soft k-Means and Masking

Modeling distractor unlabeled examples with a single cluster is likely too simplistic. Indeed, it is inconsistent with our assumption that each cluster corresponds to one class, since distractor examples may very well cover more than a single natural object category. Continuing with our unicycles and bicycles example, our web search for unlabeled images could accidentally include not only bicycles, but other related objects such as tricycles or cars. This was also reflected in our experiments, where we constructed the episode generating process so that it would sample distractor examples from multiple classes.

To address this problem, we propose an improved variant: instead of capturing distractors with a high-variance catch-all cluster, we model distractors as examples that are not within some area of any of the legitimate class prototypes. This is done by incorporating a soft-masking mechanism on the contribution of unlabeled examples. At a high level, we want unlabeled examples that are closer to a prototype to be masked less than those that are farther.

More specifically, we modify the soft k-means refinement as follows. We start by computing normalized distances $\tilde{d}_{j,c}$ between examples $\tilde{\boldsymbol{x}}_j$ and prototypes $\boldsymbol{p}_c$:

$$\tilde{d}_{j,c} = \frac{d_{j,c}}{\frac{1}{M}\sum_j d_{j,c}}, \text{ where } d_{j,c} = ||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2 \tag{7}$$

Then, soft thresholds β_c and slopes γ_c are predicted for each prototype, by feeding to a small neural network various statistics of the normalized distances for the prototype:

$$[\beta_c, \gamma_c] = \text{MLP}\left([\min_j(\tilde{d}_{j,c}), \max_j(\tilde{d}_{j,c}), \text{var}_j(\tilde{d}_{j,c}), \text{skew}_j(\tilde{d}_{j,c}), \text{kurt}_j(\tilde{d}_{j,c})]\right) \tag{8}$$

This allows each threshold to use information on the amount of intra-cluster variation to determine how aggressively it should cut out unlabeled examples.

Then, soft masks $m_{j,c}$ for the contribution of each example to each prototype are computed, by comparing to the threshold the normalized distances, as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j) \tilde{z}_{j,c} m_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c} m_{j,c}}, \text{ where } m_{j,c} = \sigma\left(\gamma_c\left(\tilde{d}_{j,c} - \beta_c\right)\right) \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function.

When training with this refinement process, the model can now use its MLP in Equation 8 to learn to include or ignore entirely certain unlabeled examples. The use of soft masks makes this process entirely differentiable[2]. Finally, much like for regular soft k-means (with or without a distractor cluster), while we could recursively repeat the refinement for multiple steps, we've found a single step to perform well enough.

4 Related Work

We summarize here the most relevant work from the literature on few-shot learning, semi-supervised learning and clustering.

The best performing methods for few-shot learning use the episodic training framework prescribed by meta-learning. The approach within which our work falls is that of metric learning methods. Previous work in metric-learning for few-shot-classification includes Deep Siamese Networks (Koch et al., 2015), Matching Networks (Vinyals et al., 2016), and Prototypical Networks (Snell et al., 2017), which is the model we extend to the semi-supervised setting in our work. The general idea here is to learn an embedding function that embeds examples belonging to the same class close together while keeping embeddings from separate classes far apart. Distances between embeddings of items from the support set and query set are then used as a notion of similarity to do classification. Lastly, closely related to our work with regard to extending the few-shot learning setting, Bachman et al. (2017) employ Matching Networks in an active learning framework where the model has a choice of which unlabeled item to add to the support set over a certain number of time steps before classifying the query set. Unlike our setting, their meta-learning agent can acquire ground-truth labels from the unlabeled set, and they do not use distractor examples.

Other meta-learning approaches to few-shot learning include learning how to use the support set to update a learner model so as to generalize to the query set. Recent work has involved learning either the weight initialization and/or update step that is used by a learner neural network (Ravi & Larochelle, 2017; Finn et al., 2017). Another approach is to train a generic neural architecture such as a memory-augmented recurrent network (Santoro et al., 2016) or a temporal convolutional network (Mishra et al., 2017) to sequentially process the support set and perform accurate predictions of the labels of the query set examples. These other methods are also competitive for few-shot learning, but we chose to extend Prototypical Networks in this work for its simplicity and efficiency.

As for the literature on semi-supervised learning, while it is quite vast (Zhu, 2005; Chapelle et al., 2010), the most relevant category to our work is related to self-training (Yarowsky, 1995; Rosenberg et al., 2005). Here, a classifier is first trained on the initial training set. The classifier is then used to classify unlabeled items, and the most confidently predicted unlabeled items are added to the training set with the prediction of the classifier as the assumed label. This is similar to our soft k-means extension to Prototypical Networks. Indeed, since the soft assignments (Equation 4) match the regular Prototypical Network's classifier output for new inputs (Equation 2), then the refinement

[2]We stop gradients from passing through the computation of the statistics in Equation 8 however, to avoid potential numerical instabilities.

can be thought of re-feeding to a Prototypical Network a new support set augmented with (soft) self-labels from the unlabeled set.

In addition to the original k-means method (Lloyd, 1982), the most related work to our setup involving clustering algorithms considers applying k-means in the presence of outliers (Hautamäki et al., 2005; Chawla & Gionis, 2013; Gupta et al., 2017). The goal here is to correctly discover and ignore the outliers so that they do not wrongly shift the cluster locations to form a bad partition of the true data. This objective is also important in our setup as not ignoring outliers (or distractors) will wrongly shift the prototypes and negatively influence classification performance.

Our contribution to the semi-supervised learning and clustering literature is to go beyond the classical setting of training and evaluating within a single dataset, and consider the setting where we must learn to transfer from a set of training classes $\mathcal{C}_{\text{train}}$ to a new set of test classes $\mathcal{C}_{\text{test}}$.

5 EXPERIMENTS

5.1 DATASETS

We evaluate the performance of our model on three datasets: two benchmark few-shot classification datasets and a novel large-scale dataset that we hope will be useful for future few-shot learning work.

Omniglot (Lake et al., 2011) is a dataset of 1,623 handwritten characters from 50 alphabets. Each character was drawn by 20 human subjects. We follow the few-shot setting proposed by Vinyals et al. (2016), in which the images are resized to 28×28 pixels and rotations in multiples of 90° are applied, yielding 6,492 classes in total. These are split into 4,800 training classes and 1,692 classes for test.

***mini*ImageNet** (Vinyals et al., 2016) is a modified version of the ILSVRC-12 dataset (Russakovsky et al., 2015), in which 600 images for each of 100 classes were randomly chosen to be part of the dataset. We rely on the class split used by Ravi & Larochelle (2017). These splits use 64 classes as training, 16 for validation, and 20 for test. All images are of size 84×84 pixels.

***tiered*ImageNet** is our proposed dataset for few-shot classification. Like *mini*Imagenet, it is a subset of ILSVRC-12. However, *tiered*ImageNet represents a larger subset of ILSVRC-12 (608 classes rather than 100 for *mini*ImageNet). Analogous to Omniglot, in which characters are grouped into alphabets, *tiered*ImageNet groups classes into broader categories corresponding to higher-level nodes in the ImageNet (Deng et al., 2009) hierarchy. There are 34 categories in total, with each category containing between 10 and 30 classes. These are split into 26 training categories and 8 testing categories (details of the dataset can be found in the supplementary material). This ensures that all of the training classes are sufficiently distinct from the testing classes, unlike *mini*ImageNet and other alternatives such as *rand*ImageNet proposed by Vinyals et al. (2016). For example, "pipe organ" is a training class and "electric guitar" is a test class in the Ravi & Larochelle (2017) split of *mini*Imagenet, even though they are both musical instruments. This scenario would not occur in *tiered*ImageNet since "musical instrument" is a high-level category and as such is not split between training and test classes. This represents a more realistic few-shot learning scenario since in general we cannot assume that test classes will be similar to those seen in training. Additionally, the tiered structure of *tiered*ImageNet may be useful for few-shot learning approaches that can take advantage of hierarchical relationships between classes. We leave such interesting extensions for future work.

5.2 EXPERIMENTAL SETUP

For each dataset, we first create an additional split to separate the images of each class into disjoint labeled and unlabeled sets. For Omniglot and *tiered*ImageNet we sampled 10% of the images of each class to form the labeled split. The remaining 90% can only be used in the unlabeled portion of episodes. For *mini*ImageNet we instead used 40% of the data for the labeled split and the remaining 60% for the unlabeled, since we noticed that 10% was too small to achieve reasonable performance and avoid overfitting.

We would like to emphasize that due to this labeled/unlabeled split, we are using strictly less label information than in the previously-published work on these datasets. Because of this, we do not

ProtoNet Model	Err.	Err. w/ D
Supervised	5.16%	5.16%
Semi-Supervised Inference	2.35%	4.70%
Soft k-Means	2.56%	4.59%
Soft k-Means+Cluster	**2.18%**	2.71%
Masked Soft k-Means	2.46%	**2.62%**

Table 1: Omniglot 1-shot Results

ProtoNet Model	1-shot Acc.	5-shot Acc.	1-shot Acc w/ D	5-shot Acc. w/ D
Supervised	43.36%	59.03%	43.36%	59.03%
Semi-Supervised Inference	48.68%	62.94%	46.16%	62.32%
Soft k-Means	48.25%	**65.72%**	46.72%	61.94%
Soft k-Means+Cluster	**50.87%**	63.75%	48.60%	61.51%
Masked Soft k-Means	50.57%	63.78%	**50.04%**	**62.50**%

Table 2: *mini*ImageNet 1/5-shot Results

expect our results to match the published numbers, which should instead be interpreted as an upper-bound for the performance of the semi-supervised models defined in this work.

Episode construction then is performed as follows. For a given dataset, we create a training episode by first sampling N classes uniformly at random from the set of training classes $\mathcal{C}_{\text{train}}$. We then sample K images from the labeled split of each of these classes to form the support set, and M images from the unlabeled split of each of these classes to form the unlabeled set. Optionally, when including distractors, we additionally sample H other classes from the set of training classes and M images from the unlabeled split of each to act as the distractors. These distractor images are added to the unlabeled set along with the unlabeled images of the N classes of interest (for a total of $MN + MH$ images). The query portion of the episode is comprised of a fixed number of images from the labeled split of each of the N chosen classes. Test episodes are created analogously, but with the N classes (and optionally the H distractor classes) sampled from $\mathcal{C}_{\text{test}}$. Note that we used $M = 5$ for training and $M = 20$ for testing, thus also measuring the ability of the models to generalize to a larger unlabeled set size. We also used $H = N = 5$, i.e. used 5 classes for both the labeled classes and the disctractor classes.

In each dataset we compare our three semi-supervised models with two baselines. The first baseline, referred to as Supervised in our tables, is an ordinary Prototypical Network that is trained in a purely supervised way on the labeled split of each dataset. The second baseline, referred to as Semi-Supervised Inference, uses the embedding function learned by this supervised Prototypical Network, but performs semi-supervised refinement of the prototypes at inference time using a step of Soft k-Means refinement. This is to be contrasted with our semi-supervised models that perform this refinement both at training time and at test time, therefore learning a different embedding function. We evaluate each model in two settings: one where all unlabeled examples belong to the classes of interest, and a more challenging one that includes distractors. Details of the model hyperparameters can be found in the appendix.

5.3 RESULTS

Results for Omniglot, *mini*ImageNet and *tiered*ImageNet are given in Tables 1, 2 and 3, respectively, while Figure 4 shows the performance of our models on *tiered*ImageNet (our largest dataset) using different values for M (number of items in the unlabeled set per class).

Across all three benchmarks, at least one of our proposed models outperform the baselines, demonstrating the effectiveness of our semi-supervised meta-learning procedure. In particular, Soft k-means+Cluster performs the best on 1-shot non-distractor settings, as the extra cluster seems to provide a form of regularization that pushes the clusters farther apart. Soft k-Means performs well on 5-shot non-distractor settings, as it considers the most unlabeled examples. Masked Soft k-Means shows the most robust performance in distractors settings, in both 1-shot and 5-shot tasks. For 5-

ProtoNet Model	1-shot Acc.	5-shot Acc.	1-shot Acc. w/ D	5-shot Acc. w/ D
Supervised	46.60%	67.18%	46.60%	67.18%
Semi-Supervised Inference	50.38%	70.26%	46.87%	68.38%
Soft k-Means	53.41%	**71.31%**	50.18%	68.83%
Soft k-Means+Cluster	**55.82%**	70.79%	49.87%	70.16%
Masked Soft k-Means	52.76%	70.08%	**50.93%**	**71.00%**

Table 3: *tiered*ImageNet 1/5-shot Results

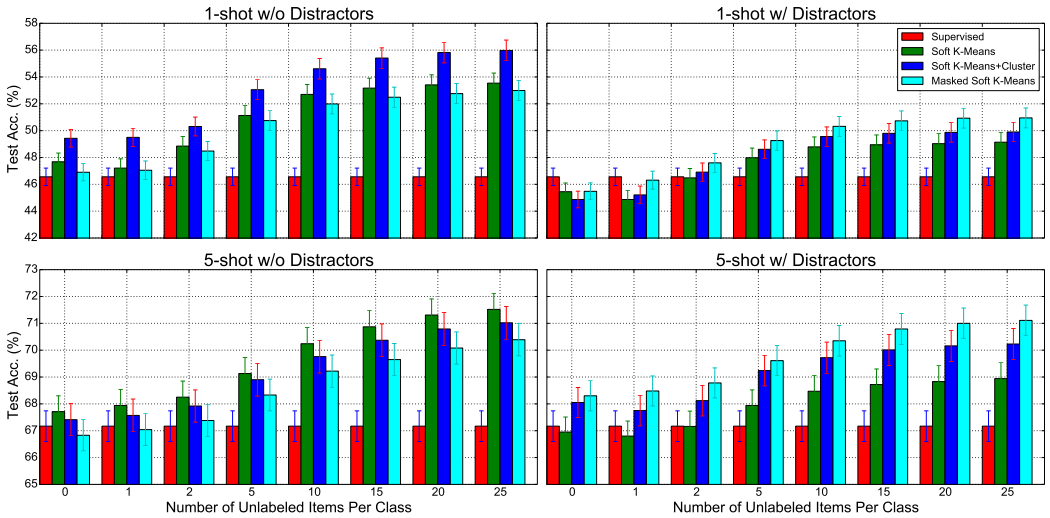

Figure 4: Model Performance on *tiered*ImageNet with different number of unlabeled items during test time.

shot, Masked soft k-Means reaches comparable performance compared to the upper bound of the best non-distractor performance.

From Figure 4, we observe clear improvement in test accuracy when the number grows from 0 to 25. Note that our models were trained with $M = 5$ and thus are showing an ability to extrapolate in generalization. This confirms that, through meta-training, the models learned to acquire a better representation that will be more helpful after semi-supervised refinement.

Note that the wins obtained in our semi-supervised learning are super-additive. Consider the case of the simple k-Means model on 1-shot without Distractors. Training only on labeled examples while incorporating the unlabeled set during test time produces an advantage of 3.8% (50.4-46.6), while incoporating the unlabeled set during training but not during test produces a win of 1.1% (47.7-46.6). Incorporating unlabeled examples during both training and test yields a win of 6.8% (53.4-46.6).

6 Conclusion

In this work, we propose a novel semi-supervised few-shot learning paradigm, where an unlabeled set is added to each episode. We also extend the setup to more realistic situations where the unlabeled set has classes not belonging to the labeled classes. To address the problem that current few-shot classification dataset is too small for a labeled vs. unlabeled split and does not have hierarchical levels of labels, we also introduce a new dataset, *tiered*ImageNet. We propose several novel extensions of Prototypical Networks, and they show consistent improvements under semi-supervised settings compared to our baselines. As future work, we are working on incorporating fast weights (Ba et al., 2016; Finn et al., 2017) into our framework so that examples can have different embedding representation given the contents in the episode.

REFERENCES

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4331–4339, 2016.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning algorithms for active learning. 2017.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 189–197. SIAM, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *34th International Conference on Machine Learning*, 2017.

Shalmoli Gupta, Ravi Kumar, Kefu Lu, Benjamin Moseley, and Sergei Vassilvitskii. Local search methods for k-means with outliers. *Proceedings of the VLDB Endowment*, 10(7):757–768, 2017.

Ville Hautamäki, Svetlana Cherednichenko, Ismo Kärkkäinen, Tomi Kinnunen, and Pasi Fränti. Improving k-means by outlier removal. In *Scandinavian Conference on Image Analysis*, pp. 978–987. Springer, 2005.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*, 2011.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *CoRR*, abs/1707.03141, 2017. URL `http://arxiv.org/abs/1707.03141`.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations*, 2017.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. One-shot learning with memory-augmented neural networks. In *33rd International Conference on Machine Learning*, 2016.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*, 2017.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pp. 3630–3638, 2016.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pp. 189–196. Association for Computational Linguistics, 1995.

Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

A *tiered*IMAGENET DATASET DETAILS

Each high-level category in *tiered*ImageNet contains between 10 and 30 ILSVRC-12 classes (17.8 on average). In the ImageNet hierarchy, some classes have multiple parent nodes. Therefore, classes belonging to more than one category were removed from the dataset to ensure separation between training and test categories. Test categories were chosen to reflect various levels of separation between training and test classes. Some test categories (such as "working dog") are fairly similar to training categories, whereas others (such as "geological formation") are quite different. The list of categories is shown below and statistics of the dataset can be found in Table 4. A visualization of the categories according to the ImageNet hierarchy is shown in Figure 5. The full list of classes per category will also be made public, however for the sake of brevity we do not include it here.

Table 4: Statistics of the *tiered*ImageNet dataset.

	Train	Test	Total
Categories	26	8	34
Classes	448	160	608
Images	572,956	206,209	779,165

Train Categories: `n02087551` (hound, hound dog), `n02098550` (sporting dog, gun dog), `n02092468` (terrier), `n02120997` (feline, felid), `n02370806` (ungulate, hoofed mammal), `n02469914` (primate), `n01726692` (snake, serpent, ophidian), `n01674216` (saurian), `n01524359` (passerine, passeriform bird), `n01844917` (aquatic bird), `n03405265` (furnishing), `n04081844` (restraint, constraint), `n03574816` (instrument), `n03738472` (mechanism), `n03800933` (musical instrument, instrument), `n03699975` (machine), `n03125870` (craft), `n03791235` (motor vehicle, automotive vehicle), `n04451818` (tool), `n03414162` (game equipment), `n03278248` (electronic equipment), `n03257877` (durables, durable goods, consumer durables), `n03419014` (garment), `n03297735` (establishment), `n02913152` (building, edifice), `n04014297` (protective covering, protective cover, protection).

Test Categories: `n02103406` (working dog), `n01473806` (aquatic vertebrate), `n02159955` (insect), `n04531098` (vessel), `n03839993` (obstruction, obstructor, obstructer, impediment, impedimenta), `n09287968` (geological formation, formation), `n00020090` (substance), `n15046900` (solid).

B EXTRA EXPERIMENTAL RESULTS

Figure 6 shows test accuracy values with different number of unlabeled items during test time. Figure 7 shows our mask output value distribution of the masked soft k-means model on Omniglot. The mask values have a bi-modal distribution, corresponding to distractor and non-distractor items.

C HYPERPARAMETER DETAILS

For Omniglot, we adopted the best hyperparameter settings found for ordinary Prototypical Networks in Snell et al. (2017). In these settings, the learning rate was set to 1e-3, and cut in half every 2K updates starting at update 2K. We trained for a total of 20K updates. For *mini*Imagenet and *tiered*ImageNet, we trained with a starting learning rate of 1e-3, which we also decayed. We started the decay after 25K updates, and every 25K updates thereafter we cut it in half. We trained for a total of 200K updates. We used ADAM (Kingma & Ba, 2014) for the optimization of our models. For the MLP used in the Masked Soft k-Means model, we use a single hidden layer with 20 hidden units with a tanh non-linearity for all 3 datasets. We did not tune the hyparameters of this MLP so better performance may be attained with a more rigorous hyperparameter search.

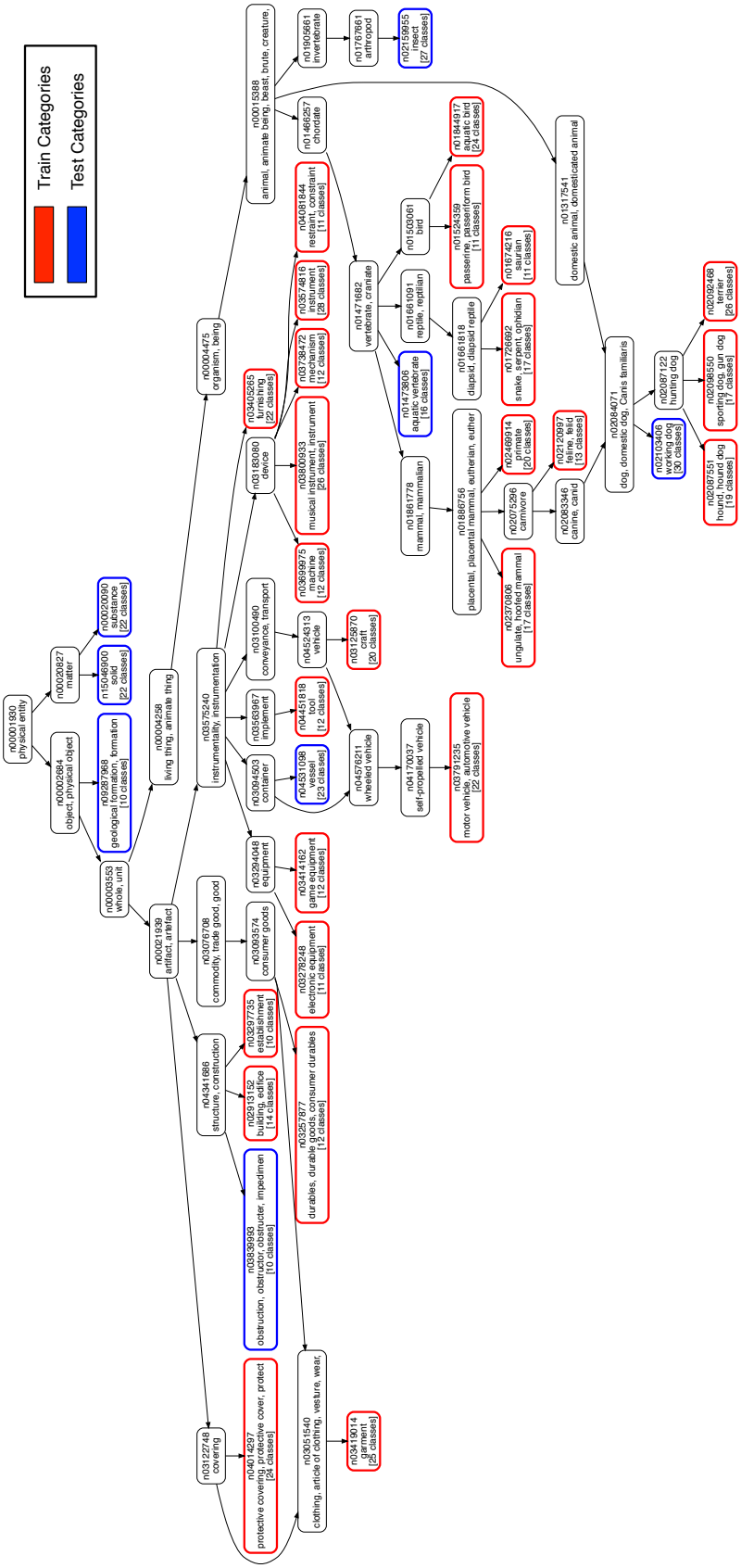

Figure 5: Hierarchy of *tiered*Imagenet categories. Training categories are highlighted in red and test categories in blue. Each category indicates the number of associated classes from ILSVRC-12. Best viewed zoomed-in on electronic version.

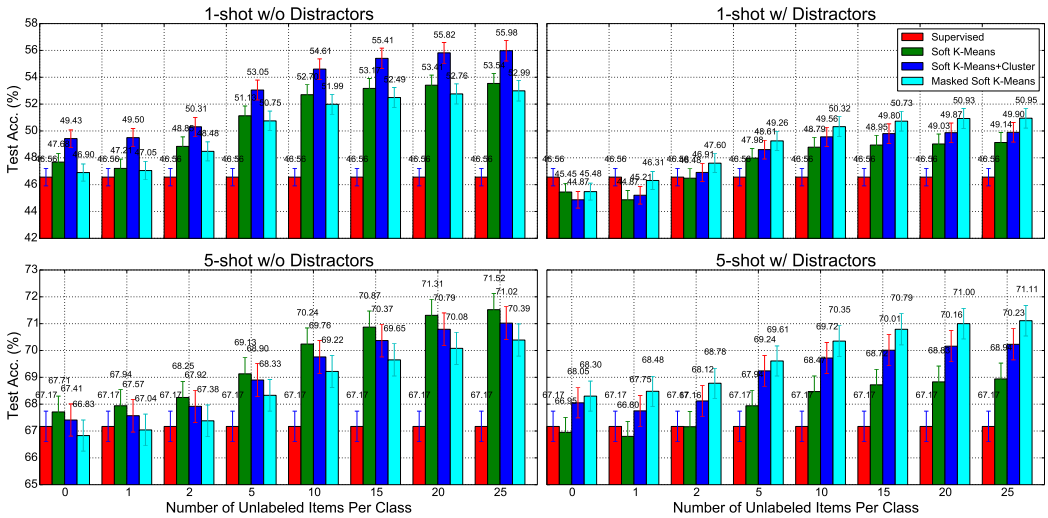

Figure 6: Model Performance on *tiered*ImageNet with different number of unlabeled items during test time. We include test accuracy numbers in this chart.

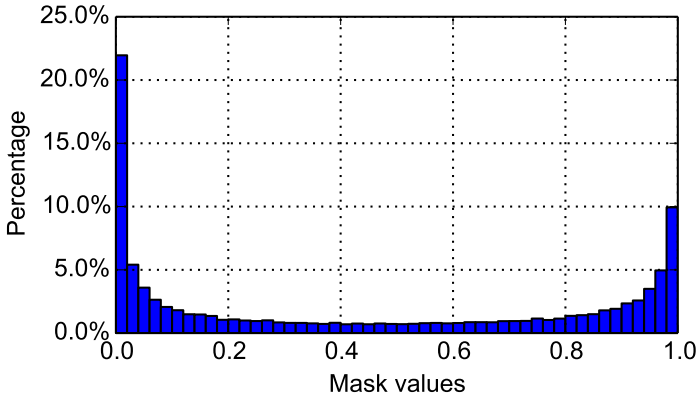

Figure 7: Mask values predicted by masked soft k-means on Omniglot.