NIPS-2017-prototypical-networks-for-few-shot-lear Paper

Thursday, 24 November 2022 00:25

Prototypical Networks for Few-shot Learning

Jake Snell
University of Toronto*
Vector Institute

Kevin Swersky
Twitter

Richard Zemel
University of Toronto
Vector Institute
Canadian Institute for Advanced Research

Abstract

We propose *Prototypical Networks* for the problem of few-shot classification, where a classifier must generalize to new classes not seen in the training set, given only a small number of examples of each new class. Prototypical Networks learn a metric space in which classification can be performed by computing distances to prototype representations of each class. Compared to recent approaches for few-shot learning, they reflect a simpler inductive bias that is beneficial in this limited-data regime, and achieve excellent results. We provide an analysis showing that some simple design decisions can yield substantial improvements over recent approaches involving complicated architectural choices and meta-learning. We further extend Prototypical Networks to zero-shot learning and achieve state-of-the-art results on the CU-Birds dataset.

1 Introduction

Few-shot classification [22, 18, 15] is a task in which a classifier must be adapted to accommodate new classes not seen in training, given only a few examples of each of these classes. A naive approach, such as re-training the model on the new data, would severely overfit. While the problem is quite difficult, it has been demonstrated that humans have the ability to perform even one-shot classification, where only a single example of each new class is given, with a high degree of accuracy [18].

Two recent approaches have made significant progress in few-shot learning. Vinyals et al. [32] proposed *Matching Networks*, which uses an attention mechanism over a learned embedding of the labeled set of examples (the *support set*) to predict classes for the unlabeled points (the *query set*).

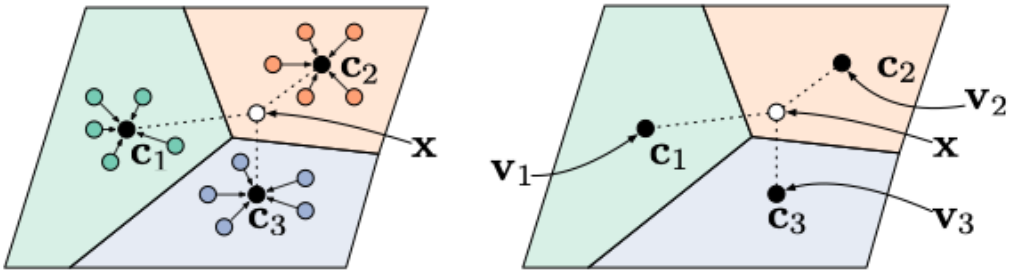
rning-

Matching Networks can be interpreted as a weighted nearest-neighbor classifier applied within an embedding space. Notably, this model utilizes sampled mini-batches called *episodes* during training, where each episode is designed to mimic the few-shot task by subsampling classes as well as data points. The use of episodes makes the training problem more faithful to the test environment and thereby improves generalization. Ravi and Larochelle [24] take the episodic training idea further and propose a meta-learning approach to few-shot learning. Their approach involves training an LSTM [11] to produce the updates to a classifier, given an episode, such that it will generalize well to a test-set. Here, rather than training a single model over multiple episodes, the LSTM meta-learner learns to train a custom model for each episode.

We attack the problem of few-shot learning by addressing the key issue of overfitting. Since data is severely limited, we work under the assumption that a classifier should have a very simple inductive bias. Our approach, *Prototypical Networks*, is based on the idea that there exists an embedding in which points cluster around a single prototype representation for each class. In order to do this, we learn a non-linear mapping of the input into an embedding space using a neural network and take a class's prototype to be the mean of its support set in the embedding space. Classification is then performed for an embedded query point by simply finding the nearest class prototype. We

*Initial work done while at Twitter.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



(a) Few-shot (b) Zero-shot

Figure 1: Prototypical Networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_\phi(y = k|\mathbf{x}) \propto \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))$.

follow the same approach to tackle zero-shot learning; here each class comes with meta-data giving a high-level description of the class rather than a small number of labeled examples. We therefore learn an embedding of the meta-data into a shared space to serve as the prototype for each class. Classification is performed, as in the few-shot scenario, by finding the nearest class prototype for an embedded query point.

In this paper, we formulate Prototypical Networks for both the few-shot and zero-shot settings. We draw connections to Matching Networks in the one-shot setting, and analyze the underlying distance function used in the model. In particular, we relate Prototypical Networks to clustering [4] in order to justify the use of class means as prototypes when distances are computed with a Bregman divergence, such as squared Euclidean distance. We find empirically that the choice of distance is vital, as Euclidean distance greatly outperforms the more commonly used cosine similarity. On several benchmark tasks, we achieve state-of-the-art performance. Prototypical Networks are simpler and more efficient than recent meta-learning algorithms, making them an appealing approach to few-shot and zero-shot learning.

2 Prototypical Networks

2 Prototypical Networks

2.1 Notation

In few-shot classification we are given a small support set of N labeled examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where each $\mathbf{x}_i \in \mathbb{R}^D$ is the D-dimensional feature vector of an example and $y_i \in \{1, \ldots, K\}$ is the corresponding label. S_k denotes the set of examples labeled with class k.

2.2 Model

Prototypical Networks compute an M-dimensional representation $\mathbf{c}_k \in \mathbb{R}^M$, or *prototype*, of each class through an embedding function $f_{\phi} : \mathbb{R}^D \rightarrow \mathbb{R}^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\phi}(\mathbf{x}_i) \tag{1}$$

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow [0, +\infty)$, Prototypical Networks produce a distribution over classes for a query point $\mathbf{x}$ based on a softmax over distances to the prototypes in the embedding space:

$$p_{\phi}(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_{k'}))} \tag{2}$$

Learning proceeds by minimizing the negative log-probability $J(\phi) = -\log p_{\phi}(y = k \mid \mathbf{x})$ of the true class k via SGD. Training episodes are formed by randomly selecting a subset of classes from the training set, then choosing a subset of examples within each class to act as the support set and a

Algorithm 1 Training episode loss computation for Prototypical Networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss J for a randomly generated training episode.

$V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples
  $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\phi}(\mathbf{x}_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_{\phi}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_{k'})) \right]$ ▷ Update loss
  end for
end for

subset of the remainder to serve as query points. Pseudocode to compute the loss $J(\phi)$ for a training

episode is provided in Algorithm 1.

2.3 Prototypical Networks as Mixture Density Estimation

For a particular class of distance functions, known as *regular Bregman divergences* [4], the Prototypical Networks algorithm is equivalent to performing mixture density estimation on the support set with an exponential family density. A regular Bregman divergence d_φ is defined as:

$$d_\varphi(\mathbf{z}, \mathbf{z}') = \varphi(\mathbf{z}) - \varphi(\mathbf{z}') - (\mathbf{z} - \mathbf{z}')^T \nabla\varphi(\mathbf{z}'), \tag{3}$$

where φ is a differentiable, strictly convex function of the Legendre type. Examples of Bregman divergences include squared Euclidean distance $\|\mathbf{z} - \mathbf{z}'\|^2$ and Mahalanobis distance.

Prototype computation can be viewed in terms of hard clustering on the support set, with one cluster per class and each support point assigned to its corresponding class cluster. It has been shown [4] for Bregman divergences that the cluster representative achieving minimal distance to its assigned points is the cluster mean. Thus the prototype computation in Equation (1) yields optimal cluster representatives given the support set labels when a Bregman divergence is used.

Moreover, any regular exponential family distribution $p_\psi(\mathbf{z}|\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ and cumulant function ψ can be written in terms of a uniquely determined regular Bregman divergence [4]:

$$p_\psi(\mathbf{z}|\boldsymbol{\theta}) = \exp\{\mathbf{z}^T\boldsymbol{\theta} - \psi(\boldsymbol{\theta}) - g_\psi(\mathbf{z})\} = \exp\{-d_\varphi(\mathbf{z}, \boldsymbol{\mu}(\boldsymbol{\theta})) - g_\varphi(\mathbf{z})\} \tag{4}$$

Consider now a regular exponential family mixture model with parameters $\boldsymbol{\Gamma} = \{\boldsymbol{\theta}_k, \pi_k\}_{k=1}^K$:

$$p(\mathbf{z}|\boldsymbol{\Gamma}) = \sum_{k=1}^K \pi_k p_\psi(\mathbf{z}|\boldsymbol{\theta}_k) = \sum_{k=1}^K \pi_k \exp(-d_\varphi(\mathbf{z}, \boldsymbol{\mu}(\boldsymbol{\theta}_k)) - g_\varphi(\mathbf{z})) \tag{5}$$

Given $\boldsymbol{\Gamma}$, inference of the cluster assignment y for an unlabeled point $\mathbf{z}$ becomes:

$$p(y = k|\mathbf{z}) = \frac{\pi_k \exp(-d_\varphi(\mathbf{z}, \boldsymbol{\mu}(\boldsymbol{\theta}_k)))}{\sum_{k'} \pi_{k'} \exp(-d_\varphi(\mathbf{z}, \boldsymbol{\mu}(\boldsymbol{\theta}_k)))} \tag{6}$$

For an equally-weighted mixture model with one cluster per class, cluster assignment inference (6) is equivalent to query class prediction (2) with $f_\phi(\mathbf{x}) = \mathbf{z}$ and $\mathbf{c}_k = \boldsymbol{\mu}(\boldsymbol{\theta}_k)$. In this case,

Prototypical Networks are effectively performing mixture density estimation with an exponential family distribution determined by d_φ. The choice of distance therefore specifies modeling assumptions about the class-conditional data distribution in the embedding space.

2.4 Reinterpretation as a Linear Model

A simple analysis is useful in gaining insight into the nature of the learned classifier. When we use Euclidean distance $d(\mathbf{z}, \mathbf{z}') = \|\mathbf{z} - \mathbf{z}'\|^2$, then the model in Equation (2) is equivalent to a linear model with a particular parameterization [21]. To see this, expand the term in the exponent:

$$-\|f_\phi(\mathbf{x}) - \mathbf{c}_k\|^2 = -f_\phi(\mathbf{x})^\top f_\phi(\mathbf{x}) + 2\mathbf{c}_k^\top f_\phi(\mathbf{x}) - \mathbf{c}_k^\top \mathbf{c}_k \tag{7}$$

The first term in Equation (7) is constant with respect to the class k, so it does not affect the softmax probabilities. We can write the remaining terms as a linear model as follows:

$$2\mathbf{c}_k^\top f_\phi(\mathbf{x}) - \mathbf{c}_k^\top \mathbf{c}_k = \mathbf{w}_k^\top f_\phi(\mathbf{x}) + b_k, \text{ where } \mathbf{w}_k = 2\mathbf{c}_k \text{ and } b_k = -\mathbf{c}_k^\top \mathbf{c}_k \tag{8}$$

We focus primarily on squared Euclidean distance (corresponding to spherical Gaussian densities) in this work. Our results indicate that Euclidean distance is an effective choice despite the equivalence to a linear model. We hypothesize this is because all of the required non-linearity can be learned within the embedding function. Indeed, this is the approach that modern neural network classification

systems currently use, e.g., [16, 31].

2.5 Comparison to Matching Networks

Prototypical Networks differ from Matching Networks in the few-shot case with equivalence in the one-shot scenario. Matching Networks [32] produce a weighted nearest neighbor classifier given the support set, while Prototypical Networks produce a linear classifier when squared Euclidean distance is used. In the case of one-shot learning, $\mathbf{c}_k = \mathbf{x}_k$ since there is only one support point per class, and Matching Networks and Prototypical Networks become equivalent.

A natural question is whether it makes sense to use multiple prototypes per class instead of just one. If the number of prototypes per class is fixed and greater than 1, then this would require a partitioning scheme to further cluster the support points within a class. This has been proposed in Mensink et al. [21] and Rippel et al. [27]; however both methods require a separate partitioning phase that is decoupled from the weight updates, while our approach is simple to learn with ordinary gradient descent methods.

Vinyals et al. [32] propose a number of extensions, including decoupling the embedding functions of the support and query points, and using a second-level, fully-conditional embedding (FCE) that takes into account specific points in each episode. These could likewise be incorporated into Prototypical Networks, however they increase the number of learnable parameters, and FCE imposes an arbitrary ordering on the support set using a bi-directional LSTM. Instead, we show that it is possible to achieve the same level of performance using simple design choices, which we outline next.

2.6 Design Choices

Distance metric Vinyals et al. [32] and Ravi and Larochelle [24] apply Matching Networks using cosine distance. However for both Prototypical Networks and Matching Networks any distance is permissible, and we found that using squared Euclidean distance can greatly improve results for both. For Protypical Networks, we conjecture this is primarily due to cosine distance not being a Bregman divergence, and thus the equivalence to mixture density estimation discussed in Section 2.3 does not hold.

Episode composition A straightforward way to construct episodes, used in Vinyals et al. [32] and Ravi and Larochelle [24], is to choose N_c classes and N_S support points per class in order to match the expected situation at test-time. That is, if we expect at test-time to perform 5-way classification and 1-shot learning, then training episodes could be comprised of $N_c = 5$, $N_S = 1$. We have found, however, that it can be extremely beneficial to train with a higher N_c, or "way", than will be used at test-time. In our experiments, we tune the training N_c on a held-out validation set. Another consideration is whether to match N_S, or "shot", at train and test-time. For Prototypical Networks, we found that it is usually best to train and test with the same "shot" number.

2.7 Zero-Shot Learning

Zero-shot learning differs from few-shot learning in that instead of being given a support set of training points, we are given a class meta-data vector $\mathbf{v}_k$ for each class. These could be determined in advance, or they could be learned from e.g., raw text [8]. Modifying Prototypical Networks to deal with the zero-shot case is straightforward: we simply define $\mathbf{c}_k = g_\vartheta(\mathbf{v}_k)$ to be a separate embedding of the meta-data vector. An illustration of the zero-shot procedure for Prototypical Networks as it relates to the few-shot procedure is shown in Figure 1. Since the meta-data vector and query point come from different input domains, we found it was helpful empirically to fix the prototype embedding g to have unit length, however we do not constrain the query embedding f.

3 Experiments

3 Experiments

For few-shot learning, we performed experiments on Omniglot [18] and the *mini*ImageNet version of ILSVRC-2012 [28] with the splits proposed by Ravi and Larochelle [24]. We perform zero-shot experiments on the 2011 version of the Caltech UCSD bird dataset (CUB-200 2011) [34].

3.1 Omniglot Few-shot Classification

Omniglot [18] is a dataset of 1623 handwritten characters collected from 50 alphabets. There are 20 examples associated with each character, where each example is drawn by a different human subject. We follow the procedure of Vinyals et al. [32] by resizing the grayscale images to 28×28 and augmenting the character classes with rotations in multiples of 90 degrees. We use 1200 characters plus rotations for training (4,800 classes in total) and the remaining classes, including rotations, for test. Our embedding architecture mirrors that used by Vinyals et al. [32] and is composed of four convolutional blocks. Each block comprises a 64-filter 3×3 convolution, batch normalization layer [12], a ReLU nonlinearity and a 2×2 max-pooling layer. When applied to the 28×28 Omniglot images this architecture results in a 64-dimensional output space. We use the same encoder for embedding both support and query points. All of our models were trained via SGD with Adam [13]. We used an initial learning rate of 10^{-3} and cut the learning rate in half every 2000 episodes. No regularization was used other than batch normalization.

We trained Prototypical Networks using Euclidean distance in the 1-shot and 5-shot scenarios with training episodes containing 60 classes and 5 query points per class. We found that it is advantageous to match the training-shot with the test-shot, and to use more classes (higher "way") per training episode rather than fewer. We compare against various baselines, including the Neural Statistician [7], Meta-Learner LSTM [24], MAML [9], and both the fine-tuned and non-fine-tuned versions of Matching Networks [32]. We computed classification accuracy for our models averaged over 1,000 randomly generated episodes from the test set. The results are shown in Table 1 and to our knowledge are competitive with state-of-the-art on this dataset.

Figure 2 shows a sample t-SNE visualization [20] of the embeddings learned by Prototypical Networks. We visualize a subset of test characters from the same alphabet in order to gain better insight, despite the fact that classes in actual test episodes are likely to come from different alphabets. Even though the visualized characters are minor variations of each other, the network is able to cluster the hand-drawn characters closely around the class prototypes.

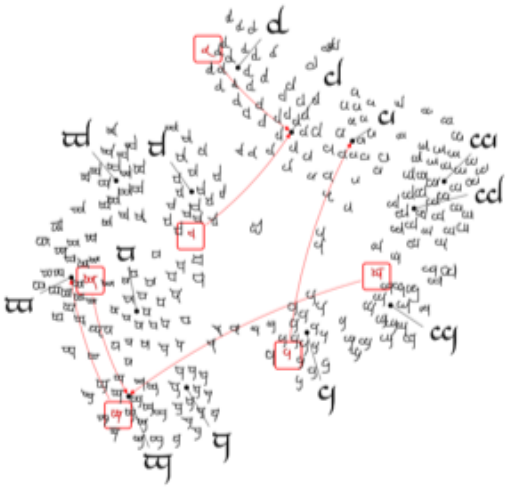


Figure 2: A t-SNE visualization of the embeddings learned by Prototypical networks on the Omniglot dataset. A subset of the Tengwar script is shown (an alphabet in the test set). Class prototypes are indicated in black. Several misclassified characters are highlighted in red along with arrows pointing to the correct prototype.

Table 1: Few-shot classification accuracies on Omniglot. *Uses non-standard train/test splits.

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot	20-way Acc. 1-shot	20-way Acc. 5-shot
Matching Networks [32]	Cosine	N	98.1%	98.9%	93.8%	98.5%

MATCHING NETWORKS [32]	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETWORKS [32]	Cosine	Y	97.9%	98.7%	93.5%	98.7%
NEURAL STATISTICIAN [7]	-	N	98.1%	99.5%	93.2%	98.1%
MAML [9]*	-	N	98.7%	**99.9%**	95.8%	**98.9%**
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	**98.8%**	99.7%	**96.0%**	**98.9%**

Table 2: Few-shot classification accuracies on *mini*ImageNet. All accuracy results are averaged over 600 test episodes and are reported with 95% confidence intervals. *Results reported by [24].

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot
BASELINE NEAREST NEIGHBORS*	Cosine	N	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
MATCHING NETWORKS [32]*	Cosine	N	$43.40 \pm 0.78\%$	$51.09 \pm 0.71\%$
MATCHING NETWORKS FCE [32]*	Cosine	N	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
META-LEARNER LSTM [24]*	-	N	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML [9]	-	N	$\mathbf{48.70 \pm 1.84\%}$	$63.15 \pm 0.91\%$
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	$\mathbf{49.42 \pm 0.78\%}$	$\mathbf{68.20 \pm 0.66\%}$

3.2 *mini*ImageNet Few-shot Classification

The *mini*ImageNet dataset, originally proposed by Vinyals et al. [32], is derived from the larger ILSVRC-12 dataset [28]. The splits used by Vinyals et al. [32] consist of 60,000 color images of size 84×84 divided into 100 classes with 600 examples each. For our experiments, we use the splits introduced by Ravi and Larochelle [24] in order to directly compare with state-of-the-art algorithms for few-shot learning. Their splits use a different set of 100 classes, divided into 64 training, 16 validation, and 20 test classes. We follow their procedure by training on the 64 training classes and using the 16 validation classes for monitoring generalization performance only.

We use the same four-block embedding architecture as in our Omniglot experiments, though here it results in a 1,600-dimensional output space due to the increased size of the images. We also use the same learning rate schedule as in our Omniglot experiments and train until validation loss stops improving. We train using 30-way episodes for 1-shot classification and 20-way episodes for 5-shot classification. We match train shot to test shot and each class contains 15 query points per episode. We compare to the baselines as reported by Ravi and Larochelle [24], which include a simple nearest neighbor approach on top of features learned by a classification network on the 64 training classes. The other baselines are two non-fine-tuned variants of Matching Networks (both ordinary and FCE) and the Meta-Learner LSTM. in the non-fine-tuned setting because the fine-tuning procedure as proposed by Vinyals et al. [32] is not fully described. As can be seen in Table 2, Prototypical Networks achieves state-of-the-art by a wide margin on 5-shot accuracy.

We conducted further analysis, to determine the effect of distance metric and the number of training classes per episode on the performance of Prototypical Networks and Matching Networks. To make the methods comparable, we use our own implementation of Matching Networks that utilizes the same embedding architecture as our Prototypical Networks. In Figure 3 we compare cosine vs. Euclidean distance and 5-way vs. 20-way training episodes in the 1-shot and 5-shot scenarios, with 15 query points per class per episode. We note that 20-way achieves higher accuracy than 5-way and conjecture that the increased difficulty of 20-way classification helps the network to generalize better, because it forces the model to make more fine-grained decisions in the embedding space. Also, using Euclidean distance improves performance substantially over cosine distance. This effect is even more pronounced for Prototypical Networks, in which computing the class prototype as the mean of embedded support points is more naturally suited to Euclidean distances since cosine distance is not a Bregman divergence.

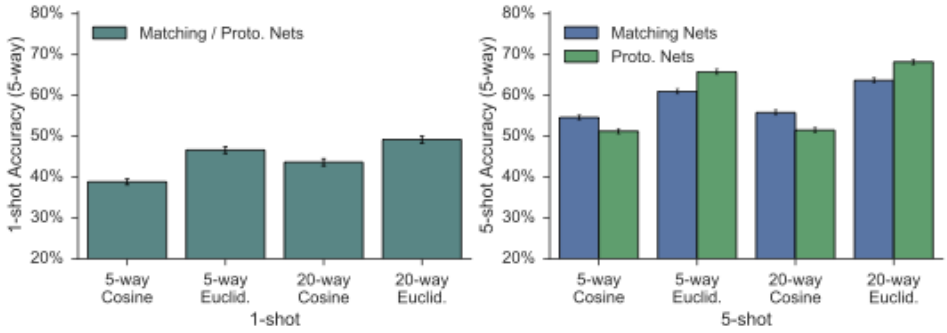


Figure 3: Comparison showing the effect of distance metric and number of classes per training episode on 5-way classification accuracy for both Matching Networks and Prototypical Networks on *mini*ImageNet. The x-axis indicates configuration of the training episodes (way, distance, and shot), and the y-axis indicates 5-way test accuracy for the corresponding shot. Error bars indicate 95% confidence intervals as computed over 600 test episodes. Note that Matching Networks and Prototypical Networks are identical in the 1-shot case.

Table 3: Zero-shot classification accuracies on CUB-200.

Model	**Image Features**	**50-way Acc.** 0-shot
ALE [1]	Fisher	26.9%
SJE [2]	AlexNet	40.3%
SAMPLE CLUSTERING [19]	AlexNet	44.3%
SJE [2]	GoogLeNet	50.1%
DS-SJE [25]	GoogLeNet	50.4%
DA-SJE [25]	GoogLeNet	50.9%
SYNTHESIZED CLASSIFIERS [6]	GoogLeNet	**54.7%**
PROTOTYPICAL NETWORKS (OURS)	GoogLeNet	**54.8%**
ZHANG AND SALIGRAMA [36]	VGG-19	**55.3% ± 0.8**

3.3 CUB Zero-shot Classification

In order to assess the suitability of our approach for zero-shot learning, we also run experiments on the Caltech-UCSD Birds (CUB) 200-2011 dataset [34]. The CUB dataset contains 11,788 images of 200 bird species. We closely follow the procedure of Reed et al. [25] in preparing the data. We use their splits to divide the classes into 100 training, 50 validation, and 50 test. For images we use 1,024-dimensional features extracted by applying GoogLeNet [31] to middle, upper left, upper right, lower left, and lower right crops of the original and horizontally-flipped image[2]. At test time we use only the middle crop of the original image. For class meta-data we use the 312-dimensional continuous attribute vectors provided with the CUB dataset. These attributes encode various characteristics of the bird species such as their color, shape, and feather patterns.

We learned a simple linear mapping on top of both the 1024-dimensional image features and the 312-dimensional attribute vectors to produce a 1,024-dimensional output space. For this dataset we found it helpful to normalize the class prototypes (embedded attribute vectors) to be of unit length, since the attribute vectors come from a different domain than the images. Training episodes were constructed with 50 classes and 10 query images per class. The embeddings were optimized via SGD with Adam at a fixed learning rate of 10^{-4} and weight decay of 10^{-5}. Early stopping on validation loss was used to determine the optimal number of epochs for retraining on the training plus validation set.

Table 3 shows that we achieve state-of-the-art results when compared to methods utilizing attributes as class meta-data. We compare our method to variety of zero-shot learning methods, including other embedding approaches such as ALE [1], SJE [2], and DS-SJE/DA-SJE [25]. We also compare to a recent clustering approach [19] which trains an SVM on a learned feature space obtained by fine-

[2]Features downloaded from `https://github.com/reedscot/cvpr2016`.

tuning AlexNet [16]. The Synthesized Classifiers approach of [6] is a manifold learning technique that aligns the class meta-data space with the visual model space, and the method of Zhang and Saligrama [36] is a structured prediction approach trained on top of VGG-19 features [30]. Since Zhang and Saligrama [36] is a randomized method, we include their reported error bars in Table 3. Our Protypical Networks outperform Synthesized Classifiers and are within error bars of Zhang and Saligrama [36], while being a much simpler approach than either.

We also ran an additional set of zero-shot experiments with stronger class meta-data. We extracted 1,024-dimensional meta-data vectors for each CUB-200 class using the pretrained Char CNN-RNN model of [25], then trained zero-shot Prototypical Networks using the same procedure described above except we used a 512-dimensional output embedding, as chosen via validation accuracy. We obtained test accuracy of 58.3%, compared to the 54.0% accuracy obtained by DS-SJE [25] with a Char CNN-RNN model. Moreover, our result exceeds the 56.8% accuracy attained by DS-SJE with even stronger Word CNN-RNN class-metadata representations. Taken together, these zero-shot classification results demonstrate that our approach is general enough to be applied even when the data points (images) are from a different domain relative to the classes (attributes).

4 Related Work

The literature on metric learning is vast [17, 5]; we summarize here the work most relevant to our proposed method. Neighborhood Components Analysis (NCA) [10] learns a Mahalanobis distance to maximize K-nearest-neighbor's (KNN) leave-one-out accuracy in the transformed space. Salakhutdinov and Hinton [29] extend NCA by using a neural network to perform the transformation. Large margin nearest neighbor (LMNN) classification [33] also attempts to optimize KNN accuracy but does so using a hinge loss that encourages the local neighborhood of a point to contain other points with the same label. The DNet-KNN [23] is another margin-based method that improves upon LMNN by utilizing a neural network to perform the embedding instead of a simple linear transformation. Of these, our method is most similar to the non-linear extension of NCA [29] because we use a neural network to perform the embedding and we optimize a softmax based on Euclidean distances in the transformed space, as opposed to a margin loss. A key distinction between our approach and non-linear NCA is that we form a softmax directly over *classes*, rather than individual points, computed from distances to each class's prototype representation. This allows each class to have a concise representation independent of the number of data points and obviates the need to store the entire support set to make predictions.

Our approach is also similar to the nearest class mean approach [21], where each class is represented by the mean of its examples. This approach was developed to rapidly incorporate new classes into a classifier without retraining, however it relies on a linear embedding and was designed to handle the case where the novel classes come with a large number of examples. In contrast, our approach utilizes neural networks to non-linearly embed points and we couple this with episodic training in order to handle the few-shot scenario. Mensink et al. [21] attempt to extend their approach to also perform non-linear classification, but they do so by allowing classes to have multiple prototypes. They find these prototypes in a pre-processing step by using k-means on the input space and then perform a multi-modal variant of their linear embedding. Prototypical Networks, on the other hand, learn a non-linear embedding in an end-to-end manner with no such pre-processing, producing a non-linear classifier that still only requires one prototype per class. In addition, our approach naturally generalizes to other distance functions, particularly Bregman divergences.

The center loss proposed by Wen et al. [35] for face recognition is similar to ours but has two main differences. First, they learn the centers for each class as parameters of the model whereas we compute protoypes as a function of the labeled examples within each episode. Second, they combine the center loss with a softmax loss in order to prevent representations collapsing to zero, whereas we construct a softmax loss from our prototypes which naturally prevents such collapse. Moreover, our approach is designed for the few-shot scenario rather than face recognition.

A relevant few-shot learning method is the meta-learning approach proposed in Ravi and Larochelle [24]. The key insight here is that LSTM dynamics and gradient descent can be written in effectively the same way. An LSTM can then be trained to itself train a model from a given episode, with the performance goal of generalizing well on the query points. MAML [9] is another meta-learning

approach to few-shot learning. It seeks to learn a representation that is easily fit to new data with few

steps of gradient descent. Matching Networks and Prototypical Networks can also be seen as forms of meta-learning, in the sense that they produce simple classifiers dynamically from new training episodes; however the core embeddings they rely on are fixed after training. The FCE extension to Matching Networks involves a secondary embedding that depends on the support set. However, in the few-shot scenario the amount of data is so small that a simple inductive bias seems to work well, without the need to learn a custom embedding for each episode.

Prototypical Networks are also related to the Neural Statistician [7] from the generative modeling literature, which extends the variational autoencoder [14, 26] to learn generative models of datasets rather than individual points. One component of the Neural Statistician is the "statistic network" which summarizes a set of data points into a statistic vector. It does this by encoding each point within a dataset, taking a sample mean, and applying a post-processing network to obtain an approximate posterior over the statistic vector. Edwards and Storkey [7] test their model for one-shot classification on the Omniglot dataset by considering each character to be a separate dataset and making predictions based on the class whose approximate posterior over the statistic vector has minimal KL-divergence from the posterior inferred by the test point. Like the Neural Statistician, we also produce a summary statistic for each class. However, ours is a discriminative model, as befits our discriminative task of few-shot classification.

With respect to zero-shot learning, the use of embedded meta-data in Prototypical Networks resembles the method of [3] in that both predict the weights of a linear classifier. The DS-SJE and DA-SJE approach of [25] also learns deep multimodal embedding functions for images and class meta-data. Unlike ours, they learn using an empirical risk loss. Neither [3] nor [25] uses episodic training, which allows us to help speed up training and regularize the model.

5 Conclusion

We have proposed a simple method called Prototypical Networks for few-shot learning based on the idea that we can represent each class by the mean of its examples in a representation space learned by a neural network. We train these networks to specifically perform well in the few-shot setting by using episodic training. The approach is far simpler and more efficient than recent meta-learning approaches, and produces state-of-the-art results even without sophisticated extensions developed for Matching Networks (although these can be applied to Prototypical Networks as well). We show how performance can be greatly improved by carefully considering the chosen distance metric, and by modifying the episodic learning procedure. We further demonstrate how to generalize Prototypical Networks to the zero-shot setting, and achieve state-of-the-art results on the CUB-200 dataset. A natural direction for future work is to utilize Bregman divergences other than squared Euclidean distance, corresponding to class-conditional distributions beyond spherical Gaussians. We conducted preliminary explorations of this, including learning a variance per dimension for each class. This did not lead to any empirical gains, suggesting that the embedding network has enough flexibility on its own without requiring additional fitted parameters per class. Overall, the simplicity and effectiveness of Prototypical Networks makes it a promising approach for few-shot learning.

Acknowledgements

We would like to thank Marc Law, Sachin Ravi, Hugo Larochelle, Renjie Liao, and Oriol Vinyals for helpful discussions. This work was supported by the Samsung GRP project and the Canadian Institute for Advanced Research.

References

[1] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *IEEE Computer Vision and Pattern Recognition*, pages 819–826, 2013.

[2] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *IEEE Computer Vision and Pattern Recognition*, 2015.

[3] Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *International Conference on Computer Vision*, pages 4247–4255, 2015.

[4] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6(Oct):1705–1749, 2005.

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

[6] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *IEEE Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.

[7] Harrison Edwards and Amos Storkey. Towards a neural statistician. *International Conference on Learning Representations*, 2017.

[8] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *International Conference on Computer Vision*, pages 2584–2591, 2013.

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

[10] Jacob Goldberger, Geoffrey E. Hinton, Sam T. Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2004.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[15] Gregory Koch. Siamese neural networks for one-shot image recognition. Master's thesis, University of Toronto, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[17] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[18] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *CogSci*, 2011.

[19] Renjie Liao, Alexander Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. *Advances in Neural Information Processing Systems*, 2016.

[20] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[21] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classifi-

cation: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[22] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *IEEE Computer Vision and Pattern Recognition*, volume 1, pages 464–471, 2000.

[23] Renqiang Min, David A Stanley, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang. A deep non-linear feature mapping for large-margin knn classification. In *IEEE International Conference on Data Mining*, pages 357–366, 2009.

[24] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[25] Scott Reed, Zeynep Akata, Bernt Schiele, and Honglak Lee. Learning deep representations of fine-grained visual descriptions. In *IEEE Computer Vision and Pattern Recognition*, 2016.

[26] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[27] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *International Conference on Learning Representations*, 2016.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[29] Ruslan Salakhutdinov and Geoffrey E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, pages 412–419, 2007.

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

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Computer Vision and Pattern Recognition*, pages 1–9, 2015.

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

[33] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2005.

[34] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[35] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[36] Ziming Zhang and Venkatesh Saligrama. Zero-shot recognition via structured prediction. In *European Conference on Computer Vision*, pages 533–548. Springer, 2016.