Large Margin Meta-Learning for Few-Shot Classification

Yong Wang[1], **Xiao-Ming Wu**[2]*, **Qimai Li**[2], **Jiatao Gu**[1], **Wangmeng Xiang**[2], **Lei Zhang**[2], **Victor O.K. Li**[1]*
[1]The University of Hong Kong, [2]The Hong Kong Polytechnic University
{wangyong, jiataogu, vli}@eee.hku.hk, {xiao-ming.wu}@polyu.edu.hk,
{csqmli, cswxiang, cslzhang}@comp.polyu.edu.hk

Abstract

This paper proposes a large margin principle to overcome the problem of data scarcity in training meta-learning models for few-shot classification. To realize it, we develop an efficient framework that can be easily incorporated in existing metric-based meta-learning models. We demonstrate that the large margin principle can improve the generalization capacity of state-of-the-art meta-learning methods and lead to consistent and considerable improvements on few-shot classification.

1 Introduction

Few-shot classification [3] is a very challenging problem as it aims to recognize new objects given only a few examples. One promising approach for tackling this problem is meta-learning. Unlike traditional supervised learning that requires a large labeled set for training, meta-learning trains a classifier that can generalize to new tasks by distilling knowledge from a large number of similar tasks and then transferring knowledge to quickly adapt to new tasks. Several directions have been explored for meta-learning, including learn to fine-tune [14, 4, 13, 9], sequence-based methods [16, 11], and metric-based methods [8, 1, 21, 18, 5, 20, 6, 10, 15].

The success of metric-based methods relies on learning a discriminative metric space in which similar samples are mapped close to each other and dissimilar ones distant such that a query can be easily classified. However, learning a good metric space is difficult due to data scarcity in training tasks. To overcome this problem, we propose a large margin principle for learning a more discriminative metric space. The key insight is that samples from different classes should be mapped as far apart as possible in the metric space to improve generalization and prevent overfitting. To this end, we develop a unified framework to impose the large margin constraint by augmenting the linear classification loss function of metric-based methods with a distance loss function – the triplet loss [17] to train a more discriminative model. Our framework is simple, robust, very easy to implement, and can be potentially applied to many metric-based meta-learning methods that adopts a linear classifier. Applications on two state-of-the-art metric-based methods – graph neural networks [6] and prototypical networks [18] show that the large margin constraint can substantially improve the generalization capacity of the original models with little computational overhead.

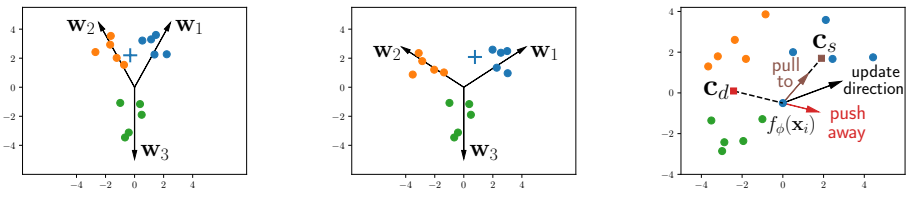
2 Large Margin Meta-Learning

2.1 Meta-Learning for Few-Shot Classification

Meta-learning consists of two phases: meta-training and meta-testing. In meta-training, a large amount of training data $\mathcal{D}_{\text{meta-train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ from a set of classes $\mathcal{C}_{\text{train}}$ are used for training a classifier, where $\mathbf{x}_i$ is the feature vector of an example, $y_i \in \mathcal{C}_{\text{train}}$ is the label, and N is the number

*corresponding authors

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.



(a) (b) (c)

Figure 1: Large margin meta-learning. (a) Classifier trained without the large margin constraint. (b) Classifier trained with the large margin constraint. (c) Gradient of the triplet loss.

of training examples. In meta-testing, a support set of S labeled examples $\mathcal{D}_s = \{(\mathbf{x}_j, y_j)\}_{j=1}^{S}$ from a set of new classes $\mathcal{C}_{\text{test}}$ is given, i.e., $y_j \in \mathcal{C}_{\text{test}}$ and $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{test}} = \emptyset$. The goal is to predict the labels of a query set $\mathcal{D}_q = \{(\mathbf{x}_j)\}_{j=S+1}^{S+Q}$, where Q is the number of queries. If the support set $\mathcal{D}_s$ contains K examples from each of C classes, i.e., $S = K \times C$, the problem is called C-way K-shot classification. Typically, K is a small number such as 1 or 5. To improve generalization, an episode-based training strategy [21] is commonly adopted to better exploit the training set $\mathcal{D}_{\text{meta-train}}$. In particular, in meta-training, a model is trained by support/query sets, where a support set is formed by sampling K examples from each of C classes from $\mathcal{C}_{train}$, and a query set is formed by sampling from the rest of the C classes' samples. The purpose is to mimic the test scenario during training.

2.2 Large Margin Principle

Metric-Based Meta-Learning. To learn quickly from only a couple of examples whose classes are unseen in meta-training, a model should acquire some transferable knowledge in meta-training. In metric-based few-shot learning [8, 6, 18], the basic idea is to learn a nonlinear mapping $f_\phi(\cdot)$ which can model the class relationship among data samples, i.e., similar samples are mapped to nearby points in the metric space while dissimilar ones are mapped far apart. Usually, the mapping $f_\phi(\cdot)$ embeds a sample $\mathbf{x}_i$ into a relatively low dimensional space, and the embedded point $f_\phi(\mathbf{x}_i)$ is then classified by a linear classifier, e.g., the softmax classifier. Note that the softmax classifier can be considered as the last fully connected layer in a neural network. Both the mapping $f_\phi(\cdot)$ and the classifier parameters are learned by minimizing the cross entropy loss: $\mathcal{L}_{\text{softmax}} = \frac{1}{N}\sum_i -\log(\frac{e^{\mathbf{w}_{y_i}^\top f_\phi(\mathbf{x}_i)}}{\sum_j e^{\mathbf{w}_j^\top f_\phi(\mathbf{x}_i)}})$, where $\mathbf{w}_j$ is a classifier weight vector corresponding to the j-th column of the weight matrix W of the softmax classifier. Without loss of generality, we omit the bias b to simplify the analysis. Note that $\mathbf{w}_j$ can be considered as the class center of samples in class j in the embedding space.

After learning $f_\phi(\cdot)$ and W, the model can be used for testing. Fig.1(a) shows a 3-way 5-shot test case, where the support samples are indicated by dots and the query sample is indicated by a cross. Samples in the same class are indicated by the same color. We can see that samples from each class are mapped to cluster around the corresponding classifer weight vector $\mathbf{w}_j$. However, the query sample, which belongs to class 1, may be wrongly classified to class 2, due to the small margin between $\mathbf{w}_1$ and $\mathbf{w}_2$.

How Can It Work Better? As each training episode consists of very few samples of each class, the standard error of the sample mean is high [12]. In other words, the class average may be a poor estimate of the true class center, and some samples may not represent its own class very well. To overcome this problem and improve the model generalization capacity on new classes, we propose to enforce a large margin between the classifier weight vectors (or class centers). The idea is that samples from different classes should be mapped as far apart as possible in the metric space. As shown in Fig. 1(b), the query sample can be correctly classified by enlarging the margin between $\mathbf{w}_1$ and $\mathbf{w}_2$. It is worth noting that the large margin principle distributes the classifier weight vectors in a balanced manner (Fig.1(b)), which leads to balanced decision boundaries.

2.3 Model

To enforce the large margin constraint, we propose to add a large margin loss function to the classification loss function, and the total loss is given by:

$$\mathcal{L} = \mathcal{L}_{\text{softmax}} + \lambda * \mathcal{L}_{\text{large-margin}}, \tag{1}$$

where λ is a balancing parameter. We choose the triplet loss [17] to be the large margin distance function, which acts on the embeddings of training samples in the metric space:

$$\mathcal{L}_{\text{large-margin}} = \frac{1}{N_t} \sum_{i=1}^{N_t} \left[\| f_\phi(\mathbf{x}_i^a) - f_\phi(\mathbf{x}_i^p) \|_2^2 - \| f_\phi(\mathbf{x}_i^a) - f_\phi(\mathbf{x}_i^n) \|_2^2 + m \right]_+, \tag{2}$$

where $m \in \mathbb{R}_{++}$ is a parameter for the margin. $(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)$ forms a triplet, where $\mathbf{x}_i^a$ is called the anchor sample, $\mathbf{x}_i^p$ is the positive sample w.r.t. the anchor sample, and $\mathbf{x}_i^n$ is the negative sample w.r.t. the anchor sample. $\mathbf{x}_i^a$ and $\mathbf{x}_i^p$ have the same labels, and $\mathbf{x}_i^a$ and $\mathbf{x}_i^n$ have different labels. N_t is the number of triplets. Any sample in the training set can be chosen as anchor. Intuitively, the additional triplet loss will help train a more discriminative mapping $f_\phi(\cdot)$, which embeds samples of the same class closer to each other and those of different classes farther apart in the metric space.

2.4 Analysis

We analyze the influence of the triplet loss on the embeddings by studying the gradient of $\mathcal{L}_{\text{large-margin}}$ with respect to $f_\phi(\mathbf{x}_i)$, the embedding of sample $\mathbf{x}_i$, during back propagation.

To find the gradient of $\mathcal{L}_{\text{large-margin}}$ with respect to $f_\phi(\mathbf{x}_i)$, we need to find the terms in which $\mathbf{x}_i$ is the anchor sample, the positive sample, or the negative sample. We partition the samples into three multisets. The first set S_s contains all samples that are paired with $\mathbf{x}_i$ and have the same label as $\mathbf{x}_i$. The second set S_d contains all samples that are paired with $\mathbf{x}_i$ but have different labels with $\mathbf{x}_i$. The third set contains samples that are not paired with $\mathbf{x}_i$. The multiplicity of an element in these multisets is the number of triplets in which the element is paired with $\mathbf{x}_i$. Note that if a sample $\mathbf{x}_s \in S_s$, the distance $\| f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_s) \|_2^2$ is added to the loss, while if a sample $\mathbf{x}_d \in S_d$, the distance is subtracted from the loss. After some rearrangements, (2) can be written as:

$$\mathcal{L}_{\text{large-margin}} = \frac{1}{N_t} \sum_i \left(\sum_{\mathbf{x}_s \in S_s} \| f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_s) \|_2^2 - \sum_{\mathbf{x}_d \in S_d} \| f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_d) \|_2^2 \right) + \text{const}, \tag{3}$$

where const denotes a constant independent of $\mathbf{x}_i$. Then the gradient of $\mathcal{L}_{\text{large-margin}}$ with respect to $f_\phi(\mathbf{x}_i)$ can be derived as:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{large-margin}}}{\partial f_\phi(\mathbf{x}_i)} &= -\frac{2|S_s|}{N_t} \left(\frac{1}{|S_s|} \sum_{\mathbf{x}_s \in S_s} f_\phi(\mathbf{x}_s) - f_\phi(\mathbf{x}_i) \right) - \frac{2|S_d|}{N_t} \left(f_\phi(\mathbf{x}_i) - \frac{1}{|S_d|} \sum_{\mathbf{x}_d \in S_d} f_\phi(\mathbf{x}_d) \right) \\ &= \underbrace{-\frac{2|S_s|}{N_t} \left(\mathbf{c}_s - f_\phi(\mathbf{x}_i) \right)}_{\text{pull towards its own class}} \underbrace{- \frac{2|S_d|}{N_t} \left(f_\phi(\mathbf{x}_i) - \mathbf{c}_d \right)}_{\text{push away from other classes}}, \end{aligned} \tag{4}$$

where $\mathbf{c}_s$ is the center of embedded points in S_s, and $\mathbf{c}_d$ is the center of embedded points in S_d. By (4), the gradient consists of two parts. The first part is a vector pointing from $f_\phi(\mathbf{x}_i)$ to the center of embedded points in S_s, which pulls $f_\phi(\mathbf{x}_i)$ towards its own class during training, as indicated by the brown arrow in Fig. 1(c). The other part is a vector pointing from the center of embedded points in S_d to $f_\phi(\mathbf{x}_i)$, which pushes $f_\phi(\mathbf{x}_i)$ away from other classes, as indicated by the red arrow in Fig. 1(c). This shows that the augmented triplet loss can effectively enforce the large margin constraint.

3 Experimental Results

In this section, we apply the proposed large margin framework on graph neural networks (GNN) [6] and prototypical networks (PN) [18]. We add the triplet loss eq. (2) to their respective objective functions to train large margin graph neural networks (L-GNN) and large margin prototypical networks (L-PN). For the strategy of triplet selection, please refer to appendix. The experiments are conducted on the benchmark Mini-Imagenet dataset, which is composed of 100 classes with 600 samples from each class.

Parameter Setup. To make sure our method can really work in practice, we use fixed hyper-parameters in all experiments. We set $\lambda = 1$ in (1). For the triplet loss, we set the margin as $\frac{1}{2N_b} \sum_{i=1}^{N_b} \| f_\phi(\mathbf{x}_i) \|_2$ with randomly initialized model parameters, where N_b is the number of all samples in one mini-batch. All testing results are averaged over 10 runs. For each run, we perform 2,000 tests for GNN and 600 tests for PN. All the results are averaged with 95% confidence intervals.

Few-Shot Classification Results. Table 1 shows the results of few-shot classification. We can see that L-GNN and L-PN consistently improve GNN and PN, respectively, on *every* learning task,

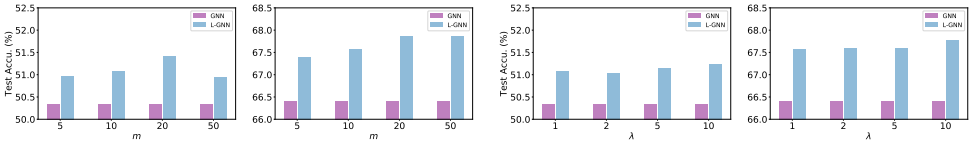
Model	Dist	5-Way 1-shot	5-Way 5-shot	10-Way 1-shot	10-Way 5-shot
GNN	-	$50.33 \pm 0.36\%$	$66.41 \pm 0.63\%$	$33.16 \pm 0.65\%$	$49.14 \pm 0.68\%$
L-GNN	-	$\mathbf{51.08 \pm 0.69\%}$	$\mathbf{67.57 \pm 0.66\%}$	$\mathbf{34.53 \pm 0.64\%}$	$\mathbf{51.48 \pm 0.69\%}$
PN	Cosine	$43.23 \pm 0.24\%$	$51.29 \pm 0.22\%$	$26.99 \pm 0.13\%$	$32.57 \pm 0.12\%$
L-PN	Cosine	$\mathbf{50.10 \pm 0.25\%}$	$\mathbf{66.94 \pm 0.21\%}$	$\mathbf{33.51 \pm 0.15\%}$	$\mathbf{50.86 \pm 0.14\%}$
PN	Euclid.	$47.98 \pm 0.25\%$	$66.72 \pm 0.21\%$	$31.91 \pm 0.15\%$	$51.50 \pm 0.14\%$
L-PN	Euclid.	$\mathbf{49.47 \pm 0.25\%}$	$\mathbf{66.83 \pm 0.21\%}$	$\mathbf{32.60 \pm 0.15\%}$	$\mathbf{51.72 \pm 0.14\%}$

Table 1: Few-shot classification on Mini-Imagenet.

Model	Trained only with labeled samples 20%-labeled	Trained only with labeled samples 40%-labeled	Semi-supervised 20%-labeled	Semi-supervised 40%-labeled
GNN	$50.33 \pm 0.36\%$	$56.91 \pm 0.42\%$	$52.45 \pm 0.88\%$	$58.76 \pm 0.86\%$
L-GNN	$\mathbf{51.08 \pm 0.69\%}$	$\mathbf{57.90 \pm 0.68\%}$	$\mathbf{54.51 \pm 0.69\%}$	$\mathbf{60.47 \pm 0.67\%}$

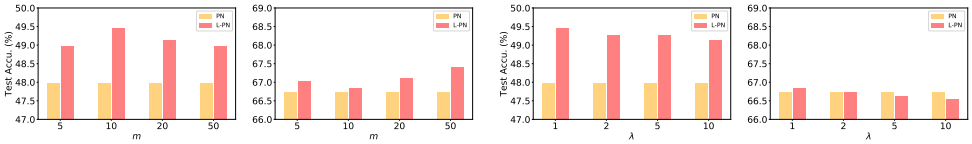
Table 2: 5-way 5-shot semi-supervised few-shot classification on Mini-Imagenet. "Trained only with labeled samples" means training and testing without using unlabeled samples; "Semi-supervised" means with unlabeled samples.

demonstrating the benefit of incorporating the large margin loss in training. For PN with cosine distance, the improvements are striking. Table 2 shows the results of semi-supervised few-shot classification. We can see that L-GNN significantly outperforms GNN. This shows that adding the large margin constraint helps learn a better embedding space for both labeled and unlabeled data.



(a) Test accuracy as m changes. (b) Test accuracy as λ changes.

Figure 2: 5-way learning using GNN on Mini-Imagenet. (For (a) and (b), Left: 1-shot, right: 5-shot)



(a) Test accuracy as m changes. (b) Test accuracy as λ changes.

Figure 3: 5-way learning using PN on Mini-Imagenet. (For (a) and (b), Left: 1-shot, right: 5-shot)

Parameter Sensitivity. We also study the sensitivity of the balancing parameter λ and the margin m. The experimental results for 5-way 1-shot and 5-shot learning are shown in Fig. 2 and 3. Our method consistently improves GNN and PN for a wide range of λ and m, demonstrating its robustness.

Running Time. The computational overhead of our method is very small. We evaluate the running time on a platform of GeForce GTX 1080 Ti. For 5-way 5-shot learning on Mini-Imagenet, one update of L-GNN takes 0.237s versus 0.228s of GNN and one update of L-PN takes 0.109s versus 0.105s of PN, incurring only 3.9% and 3.8% computational overhead.

4 Conclusions

This paper demonstrates the effectiveness of the large margin principle for metric-based meta-learning for the first time and shows that it can be implemented easily with little overhead. Our framework is simple, efficient, and can be applied to improve existing and new meta-learning methods. Future works include developing theoretical guarantees for large margin meta-learning and applying our method to solve real-word problems.

Acknowledgements. This research was supported in part by the Facebook Low Resource Neural Machine Translation Award. This work also received support from the grant 1-ZVJJ funded by the Hong Kong Polytechnic University.

References

[1] Luca Bertinetto, João F. Henriques, Jack Valmadre, and Philip H. S. Torrand Andrea Vedaldi. Learning feed-forward one-shot learners. In NIPS, 2016.

[2] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In arXiv:1801.07698, 2018.

[3] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. In TPAMI, 2006.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In ICML, 2017.

[5] Stanislav Fort. Gaussian prototypical networks for few-shot learning on omniglot. In NIPS workshop, 2017.

[6] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In ICLR, 2018.

[7] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In CVPR, 2006.

[8] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In ICML, 2015.

[9] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. In arXiv:1707.09835, 2017.

[10] Akshay Mehrotra and Ambedkar Dukkipati. Generative adversarial residual pairwise networks for one shot learning. In arXiv:1703.08033, 2017.

[11] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In ICLR, 2018.

[12] Alexander M. Mood and Franklin A. Graybill. Introduction to the theory of statistics. In McGraw Hill: New York, 1974.

[13] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In ICML, 2017.

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

[15] Mengye Ren, Eleni Triantafillou, Sachin Ravia, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In ICLR, 2018.

[16] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In ICML, 2016.

[17] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In CVPR, 2015.

[18] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In NIPS, 2017.

[19] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In NIPS, 2014.

[20] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In CVPR, 2018.

[21] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra. Matching networks for one shot learning. In NIPS, 2016.

[22] Feng Wang, Xiang Xiang, Jian Cheng, and Alan L. Yuille. Normface: L2 hypersphere embedding for face verification. In ACM MM, 2017.

[23] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In CVPR, 2018.

A Appendix

A.1 Effectiveness of Large Margin Prior

When Does It Work? The working assumption of the triplet loss is that the similarity/dissimilarity between the embedded points can be measured by Euclidean distance. If the embedded points lie on a nonlinear manifold in the metric space, Euclidean distance cannot reflect their similarity. This indicates that the embedded points should be linearly separable in the metric space, which suggests that the triplet loss should work with a linear model such as the softmax classifier in the metric space.

Computational Overhead. The computational overhead of the added triplet loss lies in two aspects: triplet selection and loss computation. It is well known that online selection of triplets could be very time-consuming when the training set is large. However, in few-shot learning, the number of samples for each class is fixed in each update iteration, so we can use an offline strategy for triplet selection. In fact, we only need to form the triplets once, and then store the parings and use them for indexing the embeddings in each update. In this way, the computational overhead for triplet selection is negligible. For the loss computation, the time complexity for each update is $\mathcal{O}(N_t d)$ where N_t is the number of triplets and d is the dimensionality of embeddings. Hence, the computation is efficient if the embeddings are low-dimensional.

A.2 Large Margin Graph Neural Networks

A.2.1 Graph Neural Networks.

In the training of GNN, each mini-batch consists of multiple episodes, and the number of episodes is the batch size. For each episode, the meta-train samples are given as:

$$\mathcal{D}_{\text{meta-train}} = \{\{(\mathbf{x}_1, y_1), ...(\mathbf{x}_S, y_S)\}, \{\overline{\mathbf{x}}_{S+1}, \overline{y}_{S+1}\}, \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_r\} : y_i, \overline{y}_{S+1} \in \{1, ..., C\}\}, \tag{5}$$

where $\{(\mathbf{x}_1, y_1), ...(\mathbf{x}_S, y_S)\}$ is the support set with labels and $\{\overline{\mathbf{x}}_{S+1}, \overline{y}_{S+1}\}$ is the query set with only one query. In the semi-supervised setting for few-shot learning, $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_r\}$ is the set of samples without labels. For few-shot learning, r is 0; and for semi-supervised few-shot learning, $r \geq 1$. For each sample, the initial feature vector is given by:

$$\mathbf{x}_i^{(0)} = (\theta(\mathbf{x}_i); h(y_i)),$$

where θ is a trainable convolutional neural network, and $h(\cdot)$ is a one-hot vector encoding the label information. Denote by $\mathbf{X}^{(0)} = [\mathbf{x}_1^{(0)}, ..., \mathbf{x}_S^{(0)}, \overline{\mathbf{x}}_{S+1}^{(0)}, \tilde{\mathbf{x}}_1^{(0)}, ..., \tilde{\mathbf{x}}_r^{(0)}]^\top$ the initial feature matrix.

A graph is constructed by taking each sample as a node, and the adjacency matrix $\tilde{A}^{(0)}$ is updated by:

$$\tilde{A}_{i,j}^{(0)} = \text{MLP}_{\tilde{\theta}}(\text{abs}(\mathbf{x}_i^{(0)} - \mathbf{x}_j^{(0)})),$$

where $\text{MLP}_{\tilde{\theta}}(\cdot)$ is a multilayer perceptron with learnable parameters $\tilde{\theta}$ and $\text{abs}(\cdot)$ is the absolute value function. The operator family is formed by $\mathcal{A} = \{\tilde{A}^{(0)}, \mathbf{1}\}$, where $\mathbf{1}$ is an all-1's matrix. The new features will be obtained by combining the features of its adjacent nodes and then projected by a projection layer:

$$\mathbf{X}^{(1)} = \sigma(\sum_{B \in \mathcal{A}} B\mathbf{X}^{(0)}\theta_B^{(0)}),$$

where $\theta_B^{(0)} \in \mathbb{R}^{d_0 \times d_1}$ is the learnable parameters of the projection layer, d_0 is the length of $\mathbf{x}_i^{(0)}$, and d_1 is the length of $\mathbf{x}_i^{(1)}$. This process can be repeated multiple times. After M iterations, we obtain the final embedding matrix $\mathbf{X}^{(M)}$.

To classify the embeddings, GNN uses a parametric softmax classifier. Namely, for C-way learning, in each training episode, the probability of the query being classified as the k-th class is:

$$P(k|\overline{\mathbf{x}}_{S+1}) = \frac{\exp(\mathbf{w}_k^\top f_\phi(\overline{\mathbf{x}}_{S+1}))}{\sum_{j=1}^{C} \exp(\mathbf{w}_j^\top f_\phi(\overline{\mathbf{x}}_{S+1}))},$$

where $f_\phi(\overline{\mathbf{x}}_{S+1}) = \overline{\mathbf{x}}_{S+1}^{(M)}$. For all episodes in a mini-batch, the softmax loss is:

$$\mathcal{L}_{\text{softmax}} = -\sum_k \overline{y}_{S+1,k} \log P(Y_* = \overline{y}_{S+1,k} | \overline{\mathbf{x}}_{S+1,k}),$$

where Y_* is the predicted label for the query in the k-th episode, $\overline{\mathbf{x}}_{S+1,k}$ is the feature of the query, and $\overline{y}_{S+1,k}$ is the ground truth label.

A.2.2 Triplet Selection.

In the training of GNN, each mini-batch consists of multiple episodes. For C-way K-shot learning, the support set of each episode is formed by sampling K examples from each of the C classes, resulting in $C \times K$ examples. The label representation of each example is a C-dimensional one-hot vector. For any two examples, if their label representations are the same, they should be mapped to the same class, and vice versa. To form the triplets, for each example (anchor) in the support set, we sample 5 positive examples from the training batch which have the same label representation with the anchor; and for each positive example, we sample 5 negative examples which have different label representation with the anchor. Hence, for each anchor example, 25 triplets are formed. For 5-way 5-shot learning, with a batch size of 40, there will be a total of $25,000$ $(25 \times 25 \times 40)$ triplets. In our experiments on Mini-Imagenet, it only takes 0.331s to form the triplets. Since the selection of triplets only needs to be done once at the beginning of training, it incurs almost no computational overhead.

A.3 Large Margin Prototypical Networks

A.3.1 Prototypical Networks.

PN is constructed based on the following steps. A training set with N labeled examples $\mathcal{D}_{\text{meta-train}} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}, y_i \in \{1, ..., C_{\text{train}}\}$ is given. First, randomly sample N_c classes from $\{1, ..., C_{\text{train}}\}$, and denote by C_{N_c} the set of class indices. Denote by c_k the k-th element in C_{N_c} and by $\mathcal{D}_{c_k}$ the subset of all training samples with $y_i = c_k$. For each class in C_{N_c}, randomly select some samples from $\mathcal{D}_{c_k}$ to form $\mathcal{S}_k$ which is a subset of the support set; and randomly select some other samples from $\mathcal{D}_{c_k}$ to form $\mathcal{Q}_k$ which is a subset of the query set, and make sure $\mathcal{S}_k \cap \mathcal{Q}_k = \emptyset$. Then for each class, compute the prototype $\mathbf{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_k} f_\phi(\mathbf{x}_i)$ where the mapping $f_\phi(\cdot)$ is typically a convolutional neural network with learnable parameters ϕ.

PN uses a non-parametric softmax classifier. Namely, for a query sample $\mathbf{x}_i$ in $\mathcal{Q}_k$, the probability of it being classified to the k-th class is:

$$P(k|\mathbf{x}_i \in \mathcal{Q}_k) = \frac{\exp(-d(f_\phi(\mathbf{x}_i), \mathbf{c}_k))}{\sum_{j=1}^{N_c} \exp(-d(f_\phi(\mathbf{x}_i), \mathbf{c}_j))}, \tag{6}$$

where $d(\cdot, \cdot)$ is a metric measuring the distance between any two vectors, which can be cosine distance or Euclidean distance. For all query samples in an episode, the classification loss is:

$$\mathcal{L}_{\text{softmax}} = -\frac{1}{N_c} \sum_{k=1}^{N_c} \frac{1}{|\mathcal{Q}_k|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{Q}_k} P(y_i|\mathbf{x}_i). \tag{7}$$

If $d(\cdot, \cdot)$ is Euclidean distance, PN is actually a linear model in the embedding space [18]. Since

$$\begin{aligned} - \parallel f_\phi(\mathbf{x}) - \mathbf{c}_k \parallel^2 &= -f_\phi(\mathbf{x})^T f_\phi(\mathbf{x}) + 2\mathbf{c}_k^T f_\phi(\mathbf{x}) - \mathbf{c}_k^T \mathbf{c}_k \\ &= \mathbf{w}_k^T f_\phi(\mathbf{x}) + b_k + \text{const}, \end{aligned}$$

where $\mathbf{w}_k = 2\mathbf{c}_k$ and $b_k = -\mathbf{c}_k^T \mathbf{c}_k$, (6) can be considered as a linear classifier.

A.3.2 Triplet Selection.

The implementation of prototypical networks does not use mini-batch in an update iteration. Take 5-shot learning for example, in one update iteration, for each class, the general practice [18] is to sample 5 support examples and 15 additional query examples, so the number of samples in each class is 20. For each sample (anchor) in the support set and the query set, we sample 10 positive examples

Model	Dist	5-Way 1-shot	5-Way 5-shot
PN	Euclid.	$47.98 \pm 0.25\%$	$66.72 \pm 0.21\%$
L-PN ($m = 10$)	Euclid.	$49.47 \pm 0.25\%$	$66.83 \pm 0.21\%$
PN+normalized triplet	Euclid.	$50.30 \pm 0.25\%$	$67.13 \pm 0.21\%$
GNN	–	$50.33 \pm 0.36\%$	$66.41 \pm 0.63\%$
L-GNN ($m = 10$)	–	$51.08 \pm 0.69\%$	$67.57 \pm 0.66\%$
GNN+normalized triplet	–	$50.73 \pm 0.70\%$	$67.28 \pm 0.64\%$
GNN+normalized contrastive	–	$50.81 \pm 0.69\%$	$67.49 \pm 0.64\%$
GNN+normface	–	$51.41 \pm 0.68\%$	$67.81 \pm 0.64\%$
GNN+cosface ($m = 0.2$)	–	$51.49 \pm 0.69\%$	$66.72 \pm 0.65\%$
GNN+arcface ($m = 0.1$)	–	$\times$	$67.21 \pm 0.64\%$

Table 3: 5-way few-shot learning on Mini-Imagenet. '$\times$': fail to converge.

from the class of anchor; and for each positive sample, we sample 10 negative examples from other classes. Hence, for each sample, 100 triplets are formed. For 5-way 5-shot learning, with $N_c = 20$ classes, a total of $40,000$ $(100 \times 20 \times 20)$ triplets are formed. In our experiments on Mini-Imagenet, it only takes 0.118s to form the triplets. Since the selection of triplets only needs to be done once at the beginning of training, it incurs almost no computational overhead.

A.4 Alternative Loss Functions

We implement and compare several of the aforementioned loss functions for large margin few-shot learning, including the normalized triplet loss, the normalized contrastive loss [7, 19], the normface loss [22], the cosface loss [23], and the arcface loss [2]. We test these models for 1-shot and 5-shot learning on Mini-Imagenet, and the results are summarized in Table 3.

To summarize, the experiments show that all large margin losses can substantially improve the original few-shot learning model, which demonstrates the benefits of the large margin principle. Compared with other loss functions, the unnormalized triplet loss has two clear advantages. First, it is more general and can be easily incorporated into metric-based few-shot learning methods. The normface loss, the cosface loss, and the arcface loss can not be directly applied to few-shot learning methods using non-parametric classifiers. Second, it is more robust than other loss functions such as the normalized triplet loss, the normalized contrastive loss, the cosface loss, and the arcface loss. On the running time, the computational overheads of these loss functions are all small, similar to that of the unnormalized triplet loss.