Non-Probabilistic Cosine Similarity Loss for Few-Shot Image Classification

Joonhyuk Kim[1]
juhkim@rit.kaist.ac.kr
Inug Yoon[1]
iuyoon@rit.kaist.ac.kr
Gyeong-Moon Park[2]
gmpark@etri.re.kr
Jong-Hwan Kim[1]
johkim@rit.kaist.ac.kr

[1] Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea

[2] Electronics and Telecommunications Research Institute (ETRI)
Daejeon, Republic of Korea

Abstract

A few-shot image classification problem aims to recognize previously unseen objects with a small amount of data. Many works have been offered to solve the problem, while a simple transfer learning method with the cosine similarity based cross-entropy loss is still powerful compared with other methods. To improve the performance, we propose a novel *Non-Probabilistic Cosine similarity (NPC) loss* for few-shot classification that can replace the cross-entropy loss with the cosine similarity. A key difference of NPC loss is that it uses values of inputs instead of their probabilities. By simply changing the loss function, our model avoids overfitting on a training set and performs well on few-shot tasks. Experimental results show that the model with NPC loss clearly outperforms those with other loss functions and also achieves excellent performance compared with state-of-the-art algorithms on Mini-Imagenet and CUB-200-2011 datasets.

1 Introduction

Traditional deep learning approaches for image classification have achieved great success [8, 19], however, a lack of data for training is fatal. Because one of the significant factors for the rapid development of this field is an increase in the amount of accessible data. In this regard, it is desirable to deal well with new classes with few data. Humans have an inherited ability to recognize previously unseen objects only by facing these few times. For example, a child, who has never seen any dogs before, can quickly learn what dogs are from a few experiences. Therefore, it is natural to research on advanced models that are fit to *few-shot image classification tasks* just as humans do.

To address few-shot image classification tasks, many previous works have been proposed based on a meta-learning approach [1, 6, 11, 12, 13, 15, 16, 17, 18, 20, 21, 23, 26]. Meta-learning aims to extract common knowledge from previous tasks, which can generalize to new tasks. An episodic training strategy is commonly used as a specific solution. In the episodic training, a large training dataset is split into many few-shot tasks, each of which

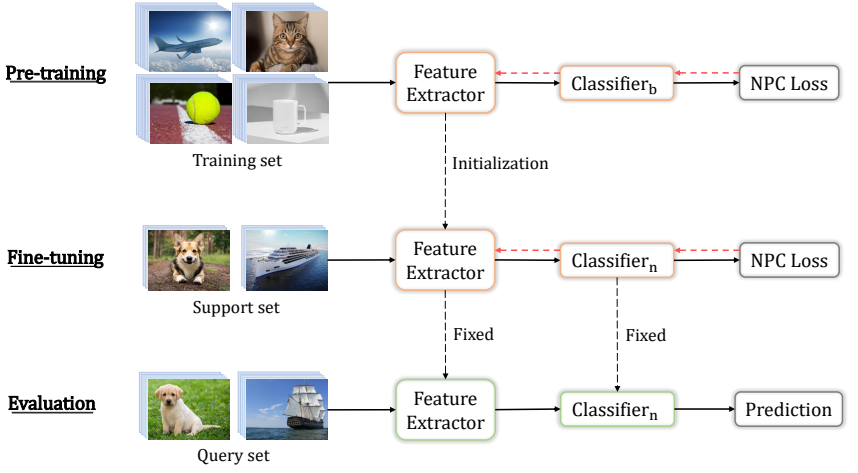


Figure 1: Our training process for few-shot image classification. The modules with orange color are trainable, and those with green color are fixed. The red dotted arrows indicate the backpropagation process. First, the feature extractor and the classifier for base classes ($Classifier_b$) are pre-trained by the training set, which contains a large number of images. Then the feature extractor from the pre-training stage and the classifier for novel classes ($Classifier_n$) are further trained by the support set, which has disjoint classes with the training set and contains only a few data per class (usually, less than or equal to 5 images per class). Finally, the model is evaluated by using the query set.

contains a few-shot training set (a support set) and its corresponding test set (a query set). One few-shot task is called one episode and the model is trained episodically.

On the other hand, recent research [2, 5, 7] showed that a standard transfer learning based approach with the cross-entropy loss could also reach comparable (or even better) results with other sophisticated models based on the meta-learning approach. In this approach, the models are pre-trained by the large training set which contains images from base classes and then fine-tuned on the few-shot support set from a test set which contains images from novel classes. Finally, classification accuracy is evaluated by the query set from the test set. Besides, in most cases, transfer learning models with the cosine similarity achieved better performance than those with the dot product [2, 7]. Inspired by these studies, we propose a new loss function with the cosine similarity and our model with the new loss achieves excellent performance by using a simple transfer learning method (see Figure 1).

In this paper, we propose a new loss function, named *Non-Probabilistic Cosine similarity (NPC) loss* for few-shot image classification, which induces to classify images by the values of the cosine similarity, not by their probabilities. Since the input of the cross-entropy loss should satisfy the probability condition, the softmax function is usually used together with it. It means that the loss function depends on the relative difference of values of the inputs. Thus, it is effective for training and predicting on images from trained classes since the feature space of the inputs is perfectly fit to the trained classes. However, it is not guaranteed that unseen classes have a similar distribution with the trained classes: the model trained with the cross-entropy loss might lead to overfitting on the training set. To solve this problem, we

design NPC loss function that classifies by the values themselves with a proper criterion to prevent such overfitting.

Our contribution is three-fold.

- To the best of our knowledge, we first address the overfitting issue when using probability-based loss functions in few-shot learning.
- We propose NPC loss that alleviates the overfitting problem in few-shot learning.
- NPC loss is easy to implement, but powerful. The model trained with NPC performs well compared to many other state-of-the-art models, and especially in 5-shot cases for Mini-ImageNet [23] and CUB-200-2011 [24] dataset, it achieves the best performance.

2 Related Work

Previous works for few-shot image classification can be grouped into two main groups; meta-learning based and transfer learning based models.

Meta-learning based: The meta-learning based models can be further categorized into gradient based and metric-learning based models.

i) Gradient based: These models typically learn model parameters from previous tasks and then fine-tune on new tasks. Ravi and Larochelle [17] presented an LSTM-based meta-learner that learns the parameters of a learner. Finn *et al*. [6] used an average of model parameters on previous tasks as initial parameters on new tasks. However, the model is unstable because the purpose of the model was to find sensitive points that can adapt to unseen tasks quickly. To overcome this drawback, Nichol *et al*. [15] simplified the computation by only using the first derivative of its loss, and Antoniou *et al*. [1] introduced various technical ways to train the model of [6]. Recently, Rusu *et al*. [18] introduced a model that embeds the features of images to low-dimensional space. This latent embedding was data-dependent and eventually resulted in fast optimization.

ii) Metric-learning based: A large number of previous models are metric-learning based. The goal of these models is to find a metric and a feature embedding that works well on few-shot tasks. Vinyals *et al*. [23] embedded the support set and the query set by different LSTM-based embedding modules. Then, they used the cosine distance as the metric to distinguish each image. Snell *et al*. [20] used just one embedding module for both the support set and the query set, and used Euclidean distance as the metric. Sung *et al*. [21] also used one embedding module as in [20], but searched for a proper metric by a suggested relation module. Li *et al*. [12] applied the Naive-Bayes Nearest-Neighbor algorithm after the images passing through their embedding module, and demonstrated the effectiveness of their approach.

Transfer learning based: Meta-learning based models are all achieved by a unique training method, *e.g*., episodic training. Unlike those models, models based on the traditional transfer learning method are trained by the whole large dataset at once. Chen *et al*. [2] showed that the model pre-trained on the large dataset and then fine-tuned on the few-shot dataset achieves comparable performance with other existing models. In addition to this, Gidaris *et al*. [7] devised an attention-based weight generator for the few-shot dataset. They used a linear combination of pre-trained weights and averaged features of the few-shot data as new weights of the few-shot data. Recently, Dhillon *et al*. [5] achieved high performance

by combining pre-training on the large dataset and the transductive fine-tuning on the few-shot dataset. They utilized unlabeled data while fine-tuning, just like in the semi-supervised learning field. The most remarkable aspect is that above models commonly applied the cosine similarity instead of the dot product, and achieved high performance.

Our model is also based on transfer learning and uses the cosine similarity. However, unlike other models, we utilize the novel loss function proposed in this paper. Our model is trained as in [2], and it shows excellent performance compared to others.

3 Approach

3.1 Problem Definition

We first formalize the few-shot classification problem by defining following notations. Let a pair $(\mathbf{x}, y)$ denote an image and its corresponding ground-truth label. Consider two datasets, a training set $\mathscr{D}_{train} = \{(\mathbf{x}_i, y_i) \mid y_i \in \mathscr{Y}_{train}\}_{i=1}^{N_{train}}$, and a test set $\mathscr{D}_{test} = \{(\mathbf{x}_i, y_i) \mid y_i \in \mathscr{Y}_{test}\}_{i=1}^{N_{test}}$, where N_{train} and N_{test} are the number of total images in each dataset, and $\mathscr{Y}_{train}$ and $\mathscr{Y}_{test}$ are sets of the ground-truth labels for images in each dataset, respectively. Note that label spaces of each dataset are disjoint with each other, *e.g.*, $\mathscr{Y}_{train} \cap \mathscr{Y}_{test} = \emptyset$.

The goal is to train a model on the training set and applying it to few-shot tasks sampled from the test set. One few-shot task (one episode) consists of two datasets, the support set and the query set. Thus, let us define subsets of the test set, the support set $D_{support} = \{(\mathbf{x}_i, y_i) \mid y_i \in \mathscr{Y}_{sampled}\}_{i=1}^{N_{support}}$, and the query set $D_{query} = \{(\mathbf{x}_i, y_i) \mid y_i \in \mathscr{Y}_{sampled}\}_{i=1}^{N_{query}}$, where $\mathscr{Y}_{sampled}$ is a sampled subset of $\mathscr{Y}_{test}$. $N_{support} = |\mathscr{Y}_{sampled}| \times K_{support}$ and $N_{query} = |\mathscr{Y}_{sampled}| \times K_{query}$ are the number of images in each dataset, which means there are $K_{support}$ and K_{query} images for each class, respectively, where $|\cdot|$ indicates the number of elements in the set. Generally, this task is called the $|\mathscr{Y}_{sampled}|$-way $K_{support}$-shot problem.

The feature extractor of the model is denoted as $F(\cdot\,|\,\phi)$ with the network parameters ϕ, and the classifier is denoted as $C(\cdot\,|\,W)$, where W is a set of weight vectors for each class. By the feature extractor trained on $\mathscr{D}_{train}$ and $D_{support}$, and the classifier trained on $D_{support}$, the model predicts the labels of images in D_{query}. For an image $\mathbf{x}$, the predicted label can be represented as $\hat{y} = C(F(\mathbf{x}\,|\,\phi^*)\,|\,W^*)$, where the $*$ symbol indicates the optimized parameters.

3.2 NPC Loss

If the ground-truth label of the i-th image $\mathbf{x}_i$ is n, *e.g.*, $y_i = n$, then, for a feature vector $\mathbf{z}_i = F(\mathbf{x}_i\,|\,\phi)$, our NPC loss is expressed as

$$L_{NPC} = \frac{1}{N}\sum_{i=1}^{N}[(e^{s\,(1-cos(\mathbf{z}_i,\mathbf{w}_n))} - 1) + \lambda \sum_{m=1,m\neq n}^{|\mathscr{Y}|} (e^{s\,(cos(\mathbf{z}_i,\mathbf{w}_m)-cos\,\varepsilon)} - 1)_+], \tag{1}$$

where N is a batch size, $\mathbf{w}_n$ is a weight vector of the classifier for class n, and $\mathscr{Y}$ is a set of the ground-truth labels. For vectors $\mathbf{a}$ and $\mathbf{b}$, $cos(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})/(\|\mathbf{a}\|_2\|\mathbf{b}\|_2)$, and $(\cdot)_+$ denotes $max(\cdot, 0)$. λ is a hyperparameter that adjusts the ratio of the first and second terms. ε and s are also hyperparameters, which are discussed in the following.

The concept of NPC loss is depicted in Figure 2. Let us assume that there are two weight vectors and one feature vector. Weight vectors are representatives of classes n and m, respectively, and the ground-truth label of the feature vector is n. NPC loss makes the

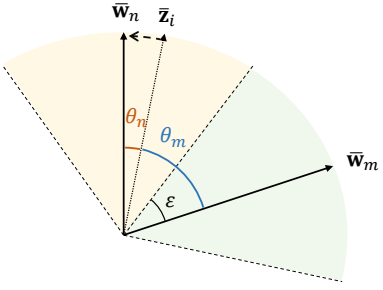


Figure 2: $\bar{\mathbf{w}}_n$ and $\bar{\mathbf{w}}_m$ denotes the normalized weight vectors for classes n and m, respectively. $\bar{\mathbf{z}}_i$ is the normalized feature vector which has the ground-truth label of n. By NPC loss, an angular distance between $\bar{\mathbf{w}}_n$ and $\bar{\mathbf{z}}_i$, θ_n approaches to zero, and the one between $\bar{\mathbf{w}}_m$ and $\bar{\mathbf{z}}_i$, θ_m becomes larger than the criterion ε.

angular distance between the weight vector of the ground-truth label and the corresponding feature vector small and the one between the weight vector and the feature vector belonging to another class large. The former is caused by the first term of Equation 1, and the latter by the second term of it. The first term is smaller as the value of $cos(\mathbf{z}_i, \mathbf{w}_n)$ increases, and zero when the cosine value equals to 1. That is, it induces the two vectors to be close to each other. The second term, on the other hand, is smaller as the value of $cos(\mathbf{z}_i, \mathbf{w}_m)$ decreases, and is zero when $cos(\mathbf{z}_i, \mathbf{w}_m) \leq cos\ \varepsilon$. It means it makes the angle between the two vectors to be larger than ε.

ε can be viewed as a criterion to be set in advance. As ε increases, the inter-class variance becomes larger; however, since the importance of the first term in Equation 1 decreases, the intra-class variance also becomes larger. And both the inter-class and the intra-class variances become smaller vice versa. Therefore, it is crucial to decide the proper value of ε in advance. We assume that there is an optimal value of ε for each dataset. Thus, we set the value of ε by tuning on the validation set of the datasets. Further analysis for ε is described in Section 4.5.

s is a re-scale factor that is also frequently used by other loss functions with the cosine similarity [2, 4, 7, 14, 16, 25]. Since the value of the cosine similarity is between -1 and 1, the range of the exponential function with this is also limited. In the case of the exponential function, when the exponents are greater than or equals to zero, the ratio of its values is bigger as the exponents increase. For example, $e^2/e^1 \ll e^{20}/e^{10}$. Therefore, there is an effect of enhancing the discriminative power of the model by multiplying the re-scale factor.

3.3 Comparison with Cross-entropy Loss

For a feature vector $\mathbf{z}_i = F(\mathbf{x}_i|\phi)$, the Cosine-similarity-based Cross-entropy (CC) loss with the softmax function is expressed as

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\,cos(\mathbf{z}_i,\mathbf{w}_n)}}{\sum_{m=1}^{|\mathscr{Y}|} e^{s\,cos(\mathbf{z}_i,\mathbf{w}_m)}}. \tag{2}$$

The cross-entropy loss also gathers the same class together and pushes the other classes away. In this respect, it is similar to NPC loss. However, the most significant difference is whether the loss function utilizes the probability or not. The inputs of the negative logarithmic function should be between 0 and 1 for the function always to be positive and monotonically decreasing. The softmax function satisfies this condition. And since the total sum of it is always 1, the relative values of the inputs are naturally adjusted. Because of these characteristics, it leads the model to have high discriminative power.

However, this can be a problem in the case of few-shot image classification. The fact that the relative values of the inputs automatically optimize the model implies that there is a possibility of overfitting on the training set. Such a model is likely to extract the detailed features for the base classes; thus, if the model is applied for the novel classes, these features will be disturbing. Hence, we aim to train the model that extracts features that are also suited for the novel classes to ensure the high discriminative power for those. We hypothesize that this could be achievable by not using probabilities, which leads the model to extract less detailed features for base classes. And finally, by fine-tuning, the model can be further trained to extract detailed features for the novel classes. However, there is no guarantee that the less detailed base classes' features will be applied to novel classes. Thus, we introduce ε to adjust this discrepancy. The effectiveness of the proposed approach for few-shot classification is demonstrated in Section 4.

3.4 Training Strategy

Our model follows the procedure of the network pre-training, fine-tuning, and finally, evaluation. Figure 1 illustrates the whole procedure.

Pre-training: In the pre-training stage, we train the feature extractor $F(\cdot|\phi)$ and the classifier $C(\cdot|W_{train})$, where W_{train} is a set of the weight vectors of each class in $\mathscr{D}_{train}$. By minimizing NPC loss on examples in the training set, we find the optimal values for ϕ and W_{train} from scratch.

Fine-tuning: We train the feature extractor $F(\cdot|\phi)$ and the classifier $C(\cdot|W_{sampled})$, where $W_{sampled}$ is a set of the weight vectors of each class in $D_{support}$. Unlike in the pre-training stage, ϕ and $W_{sampled}$ are initialized by specific values. We use the feature extractor from the pre-training stage, *e.g.*, the initial value of ϕ is a result of the pre-training stage. For the classifier, the feature vectors belonging to each class are averaged. For the feature vectors having the ground-truth label of n, the initial value of the weight vector for class n is

$$\mathbf{w}_n^{init} = \frac{1}{K_{support}} \sum_{k=1}^{K_{support}} \bar{\mathbf{z}}_k^n, \tag{3}$$

where $\bar{\mathbf{z}}_k^n$ denotes the k-th normalized feature vector which has the ground-truth label of n. To recap, the feature extractor and the classifier from the specific initial values are trained on examples in $D_{support}$.

Evaluation: Finally, we predict the label of examples in D_{query} by the feature extractor and the classifier from the fine-tuning stage. For the feature vector $\mathbf{z}_i$ obtained from the i-th image of D_{query}, the predicted label is

$$\hat{y}_i = \arg\max_n cos(\mathbf{z}_i, \mathbf{w}_n), \tag{4}$$

for $n = 1, ..., |\mathscr{Y}_{sampled}|$.

4 Experimental Results

In this section, the experimental results are delineated for few-shot image classification problems. First, the proposed NPC loss is compared with other losses while using the same models. Next, our model depicted in Figure 1 is compared with the state-of-the-art algorithms on benchmark datasets. And then, visualizations of the feature vectors are provided. Finally, an analysis about the hyperparameter ε is conducted.

4.1 Implementation Details

Here we describe details about the datasets and experimental settings for our experiments.

Datasets: We used two benchmark datasets which are Mini-ImageNet and CUB-200-2011 (CUB). Mini-ImageNet dataset is a subset of ImageNet dataset [3]. There are 100 classes sampled from ImageNet and each class contains 600 images. Vinyals *et al.* [23] first presented this dataset, however, recent works utilized the setting provided by Ravi and Larochelle [17]. Thus, we used the same setting which splits 100 classes into 64 training, 16 validation, and 20 test classes. CUB dataset contains 11,788 images of birds from 200 classes. It was firstly presented by Wah *et al.* [24], and we split the dataset into 100 training, 50 validation, and 50 test classes, which is the setting provided by Hilliard *et al.* [9].

Network architecture: For the feature extractor $F(\cdot|\phi)$, we used two different architectures: $\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$ and ResNet-18. The former one refers to four conjugated convolution modules. Each module consists of 3×3 convolution filters and followed by batch normalization, ReLU nonlinearity, and 2×2 max-pooling. The first two modules have 64 channels, and the last two have 128 channels. The latter one refers to the ResNet [8] like architecture used in [2] (Refer to the supplementary material for the details). The input image size is 84×84 for $\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$ and 224×224 for ResNet-18. For the classifier $C(\cdot|W)$, we used one linear layer with the cosine similarity, where the output dimension of the $Classifier_b$ in Figure 1 depends on the number of the training classes of each dataset, and the one of the $Classifier_n$ depends on the number of ways of few-shot tasks.

Experimental settings: For the both datasets, we used train classes for the pre-training, and test classes for the evaluation. The validation classes were used for tuning hyperparameters. A standard data augmentation including random crop, left-right flip, color jitter was applied in the pre-training stage. We implemented 1-shot, and 5-shot cases and evaluated with 15 query shots, *e.g.*, $K_{support} = 1, 5$, and $K_{query} = 15$. For the pre-training stage, the Adam optimizer [10] with a fixed learning rate of 10^{-3} was used. ε was set to 90° for Mini-ImageNet dataset, and 30° for CUB dataset. For the fine-tuning stage, we trained the model with the Adam optimizer with a fixed learning rate of 5×10^{-4} for 50 epochs. λ and s were set to 1/15 and 30, respectively, and the batch size of 128 was used for the both stages.

4.2 Comparison with Other Losses

First, we compared NPC loss with the CC loss (Equation 2), ArcFace (AF) [4], and CosFace (CF) [25]. We selected the AF and CF for comparison since they show higher discriminative power than other margin added cosine similarity losses as well as the cross-entropy loss. For the i-th feature vector $\mathbf{z}_i = F(\mathbf{x}_i|\phi)$, these losses are expressed as

$$L = -\frac{1}{N}\sum_{i=1}^{N} log \frac{e^{s\,(cos(arccos(cos(\mathbf{z}_i,\mathbf{w}_n))+M_1)-M_2)}}{e^{s\,(cos(arccos(cos(\mathbf{z}_i,\mathbf{w}_n))+M_1)-M_2)} + \sum_{m=1,m\neq n}^{|\mathscr{Y}|} e^{s\,cos(\mathbf{z}_i,\mathbf{w}_m)}}, \tag{5}$$

Loss function	Architecture	5-way		20-way	
		1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
CC	$\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$	**52.78 ± 0.77**	68.76 ± 0.68	**24.43 ± 0.27**	39.38 ± 0.24
AF		**52.62 ± 0.77**	68.82 ± 0.66	23.70 ± 0.25	38.52 ± 0.23
CF		**52.32 ± 0.81**	69.15 ± 0.66	**24.71 ± 0.27**	40.18 ± 0.25
NPC		**52.72 ± 0.86**	**70.79 ± 0.70**	**24.78 ± 0.26**	**42.10 ± 0.25**
CC	ResNet-18	50.02 ± 0.84	66.60 ± 0.68	23.30 ± 0.27	37.03 ± 0.24
AF		47.42 ± 0.79	63.64 ± 0.69	21.21 ± 0.27	33.88 ± 0.25
CF		50.45 ± 0.83	66.96 ± 0.69	23.23 ± 0.27	36.64 ± 0.24
NPC		**57.51 ± 0.85**	**75.37 ± 0.65**	**28.51 ± 0.31**	**48.30 ± 0.25**

Table 1: The results of few-shot image classification on the test set of Mini-ImageNet dataset without fine-tuning. The results are average accuracies over 600 test episodes with the 95% confidence intervals.

Method	Architecture	Mini-ImageNet		CUB	
		1-shot (%)	5-shot (%)	1-shot (%)	5-shot (%)
Matching Networks [23]	$\text{conv}(64)_{\times 4}$	46.6	60.0	-	-
Matching Networks*	$\text{conv}(64)_{\times 4}$	48.14 ± 0.78	63.48 ± 0.66	60.52 ± 0.88	75.29 ± 0.75
LSTM Meta-learner [20]	$\text{conv}(64)_{\times 4}$	43.44 ± 0.77	60.60 ± 0.71	-	-
Prototypical Networks [21]	$\text{conv}(64)_{\times 4}$	49.42 ± 0.78	68.20 ± 0.66	-	-
Prototypical Networks*	$\text{conv}(64)_{\times 4}$	44.42 ± 0.84	64.24 ± 0.72	50.46 ± 0.88	76.39 ± 0.64
MAML [6]	$\text{conv}(32)_{\times 4}$	48.70 ± 1.84	63.11 ± 0.92	-	-
MAML*	$\text{conv}(32)_{\times 4}$	46.47 ± 0.82	62.71 ± 0.71	54.73 ± 0.97	75.75 ± 0.76
Relation Network [22]	conv(64-96-128-256)	50.44 ± 0.82	65.32 ± 0.70	-	-
Relation Network*	conv(64-96-128-256)	49.31 ± 0.85	66.60 ± 0.69	**62.34 ± 0.94**	77.84 ± 0.68
Dynamic Few-shot [9]	$\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$	55.95 ± 0.84	73.00 ± 0.64	-	-
Baseline++ [9]	$\text{conv}(64)_{\times 4}$	48.24 ± 0.75	66.43 ± 0.63	60.53 ± 0.83	79.34 ± 0.61
DN4 [14]	$\text{conv}(64)_{\times 4}$	51.24 ± 0.74	71.02 ± 0.64	53.15 ± 0.84	**81.90 ± 0.60**
NPC	$\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$	52.73 ± 0.83	71.26 ± 0.70	60.65 ± 0.89	**81.30 ± 0.59**
Baseline++ [9]	ResNet-18	51.87 ± 0.77	75.68 ± 0.63	-	-
Dynamic Few-shot [9]	ResNet-12	55.45 ± 0.89	70.13 ± 0.68	-	-
TADAM [15]	ResNet-12	58.5 ± 0.3	76.7 ± 0.3	-	-
MetaOpt SVM [11]	ResNet-12	**62.64 ± 0.61**	78.63 ± 0.46	-	-
LEO† [18]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	-	-
NPC	ResNet-18	60.98 ± 0.87	**80.17 ± 0.65**	-	-

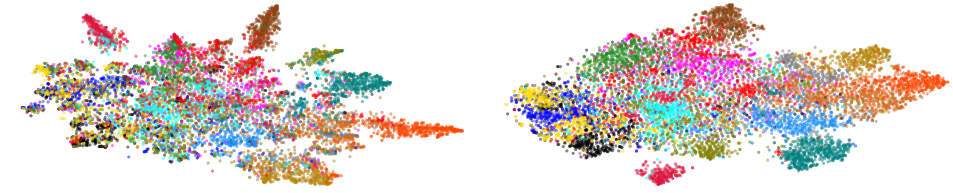
* Results from [9]. † Pre-trained on both the training and validation sets. - Not reported.

Table 2: The mean accuracies over 600 test episodes with the 95% confidence intervals of 5-way image classification task. The test sets of each dataset are used. The second column indicates the architecture of the feature extractor. All results are cited from the original works except for those with the $*$ symbol.

where M_1 and M_2 are an angular margin. $M_1 = 0$ for the CF, and $M_2 = 0$ for the AF.

For a fair comparison, we used the same model architectures, training procedures and only changed the loss functions. Table 1 shows the results of 5-way and 20-way image classifications on the test set of Mini-ImageNet dataset without fine-tuning. For the CC, the AF, and the CF, we set s to 10 and trained the model by the Adam optimizer with early stopping. M_1 and M_2 was set to 0.1 and 0.35 for the AF and the CF, respectively.

When the shallow architecture ($\text{conv}(64)_{\times 2}$-$(128)_{\times 2}$) was used, the model trained with NPC loss achieves better performance only when the 5-shot cases. However, when the deep architecture (ResNet-18) was used, the model trained with NPC loss overwhelms those with other probabilistic losses. Note that the accuracies of the probability-based losses decrease when the feature extractor architecture is deepened. Generally, the use of deep architectures improves performance but also aggravates overfitting issues. The CC, AF, and CF suffer performance degradation as the architecture deepened, but in the case of NPC, performance



(a) (b)

Figure 3: t-SNE visualizations of the feature vectors of the test classes of Mini-ImageNet dataset extracted from the ResNet-18 architecture before fine-tuning. Each figure indicates the feature visualization of (a) the model trained with the CC loss and (b) the one trained with NPC loss.

increased. Through this experiment, we proved our hypothesis that overfitting has a negative effect on few-shot learning, and NPC offsets this effect.

4.3 Results on Benchmark Datasets

We conducted standard few-shot classification experiments on Mini-ImageNet and CUB datasets and compared with the state-of-the-art models. Table 2 shows the results of 5-way image classification on both datasets. $\text{conv}(k)_{\times l}$ denotes that the convolution module with k channels are conjugated l times. ResNet-12 indicates residual network [8] modules used in [16] and [11]. WRN-28-10 denotes a wide residual network [27] with a depth of 28 and a widening factor of 10.

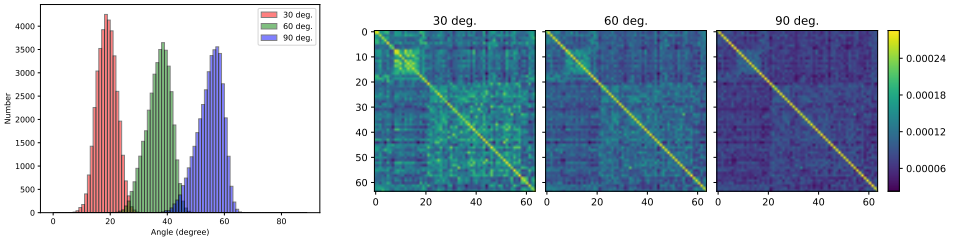
Among the models using four convolution modules as their feature extractors, our model with fine-tuning achieved the best performance except for Dynamic Few-shot model on Mini-ImageNet dataset. Moreover, our model with fine-tuning showed the best accuracy for the 5-shot case on CUB dataset. When using ResNet-18 architecture, our model after fine-tuning overwhelmed other models with the conv architectures; however, for a fair comparison, we compared our results on Mini-ImageNet with some other models with deep structures. Our model outperformed most of the models and achieved the best performance for the 5-shot case. Note that Dynamic Few-shot model with the deep structure suffered from the performance degradation probably caused by overfitting.

Our model with NPC loss achieved excellent performances on two benchmark datasets, and compared with other models, it is easy to implement because we simply replace the loss function of the standard transfer learning method.

4.4 Feature Visualization

To show that the model trained with NPC loss has high discriminative power for the novel classes, we compared the t-SNE visualizations [22] for the feature vectors of Mini-ImageNet's test classes extracted from each extractor: one trained by the CC loss, and the other trained by NPC loss. Here, ResNet-18 architecture was used as the feature extractor.

As can be seen from Figure 3, when the model is trained with the CC loss, most of the classes are hard to distinguish from each other. The fact that some classes are well-clustered indicates that there are some similar classes between the base and novel classes. However, when the model is trained with NPC loss, most of the classes are well-clustered. Comparing well-clustered classes from Figure 3(a) with (b), intra-class variance of each class is smaller



(a) (b)

Figure 4: An analysis of the feature vectors of 64 training classes in Mini-ImageNet dataset after the pre-training stage. (a) The distribution of angles between 64 weight vectors and feature vectors belonging to each class. As ε increases, the angle distribution moves to the right. (b) A visualization of covariance matrices of the weight vectors. The distance between each weight vector becomes larger as ε increases.

in the case of Figure 3(a). It indicates the model trained with NPC loss extracts less detailed features of the base classes but can be shared across the novel classes.

4.5 Hyperparameter Analysis

Here, we further analyzed ε in Equation 1. It plays a crucial role in determining the distribution of the feature vectors as mentioned in Section 3.2. Both the intra-class and the inter-class variances increase as ε becomes larger, and vise versa. Figure 4 illustrates this characteristic. After the pre-training stage, we measured angles between the weight vectors in W_{train} and corresponding feature vectors. The left part of the figure shows the distribution of these angles when ε is 30°, 60°, and 90°. As ε becomes larger, the angle distribution moves to the right, and it means that the intra-class variance is getting bigger. The right part of the figure is a visualization of covariance matrices of W_{train}. The larger the ε, the smaller the values of elements of the covariance matrix except for the diagonal elements. This indicates that inter-class variance is getting bigger as ε increases. Since the intra-class variance should be small and the inter-class variance should be large for excellent discriminative power, it is essential to adjust ε to find the proper balance of the distributions of the feature vectors.

We could roughly set the initial value of ε as follows: In a fine-grained dataset such as CUB, the intra-class variance should be small since both the base and novel classes are likely to share detailed features, which means ε should be small. Datasets such as Mini-ImageNet are less likely to share such features, thus, requiring large ε. We observed that our model achieved the best performances when ε was set to 30° for CUB dataset, and set to 90° for Mini-ImageNet dataset.

5 Conclusion

In this paper, we proposed a novel NPC loss function for few-shot image classification. We addressed the overfitting problem when using the probabilistic loss functions in this task. We applied our loss function to a simple transfer learning method and obtained excellent classification performance. Our observation showed that proposed NPC loss could alleviate the negative effect of the overfitting issue. Future work could focus on using NPC loss for other models with different network architectures or meta-learning strategies.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00440, Development of Artificial Intelligence Technology that Continuously Improves Itself as the Situation Changes in the Real World).

References

[1] A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. In *Proc. ICLR*, 2019.

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

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proc. CVPR*, 2019.

[5] G.S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto. A baseline for few-shot image classification. In *Proc. ICLR*, 2020.

[6] C. Finn, P. Abbeel, and S. Levin. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017.

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

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

[9] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C.D. Corley, and N.O. Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

[10] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proc. CVPR*, 2019.

[12] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, and J. Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proc. CVPR*, 2019.

[13] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning. In *Proc. ICML*, 2018.

[14] W. Liu, Y. Wen, Z. Yu, M. Li, and B. Raj L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proc. CVPR*, 2017.

[15] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. In *Proc. ICML*, 2018.

[16] B.N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Proc. NIPS*, 2018.

[17] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *Proc. ICLR*, 2017.

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

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

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

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

[22] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[23] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one-shot learning. In *Proc. NIPS*, 2016.

[24] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

[25] H. Wang, Y. Wang, Z. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proc. CVPR*, 2018.

[26] S.W. Yoon, J. Seo, and J. Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proc. ICML*, 2019.

[27] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.