A Closer Look at Prototype Classifier for Few-shot Image Classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

The prototypical network is a prototype classifier based on meta-learning and is widely used for few-shot learning because it classifies unseen examples by constructing class-specific prototypes without adjusting hyper-parameters during meta-testing. Interestingly, recent research has attracted a lot of attention, showing that training a new linear classifier, which does not use a meta-learning algorithm, performs comparably with the prototypical network. However, the training of a new linear classifier requires the retraining of the classifier every time a new class appears. In this paper, we analyze how a prototype classifier works equally well without training a new linear classifier or meta-learning. We experimentally find that directly using the feature vectors, which is extracted by using standard pre-trained models to construct a prototype classifier in meta-testing, does not perform as well as the prototypical network and training new linear classifiers on the feature vectors of pre-trained models. Thus, we derive a novel generalization bound for a prototypical classifier and show that the transformation of a feature vector can improve the performance of prototype classifiers. We experimentally investigate several normalization methods for minimizing the derived bound and find that the same performance can be obtained by using the L2 normalization and minimizing the ratio of the within-class variance to the between-class variance without training a new classifier or meta-learning.

1 Introduction

Few-shot learning is used to adapt quickly to new classes with low annotation cost. Meta-learning is a standard training procedure to tackle the few-shot learning problem and the *prototypical network* [1], a.k.a ProtoNet is a widely used meta-learning algorithm for few-shot learning. In ProtoNet, we use a prototype classifier based on meta-learning to predict the classes of unobserved objects by constructing class-specific prototypes without adjusting the hyper-parameters during meta-testing.

ProtoNet has two advantages. (1) Since the nearest neighbor method is applied on query data and class prototypes during the meta-test phase, no hyper-parameters are required in the meta-test phase. (2) The classifiers can quickly adapt to new environments because they do not have to be re-trained for the support set in the meta-test phase when new classes appear. Moreover, the generalization bound of ProtoNet in relation to the number of shots in a support set has been studied [2]. The bound suggests that the performance of ProtoNet depends on the ratio of the within-class variance to the between-class variance of the features extracted using the meta-trained model.

There have been studies on training a new linear classifier on the features extracted using a pre-trained model without meta-learning, which can perform comparably with the meta-learned models [3, 4]. We call this approach the linear-evaluation-based approach. In these studies, the models are trained with the standard classification problem, i.e., models are trained with cross-entropy loss after linear

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

projection from the embedding space to the class-probability space. The linear-evaluation-based approach has three advantages over meta-learning. (1) Training converges faster than meta-learning. (2) Implementation is simpler. (3) Meta-learning decreases in performance if the number of shots does not match between meta-training and meta-testing [2]; however, the linear-evaluation-based approach does not need to take this into account. However, it requires retraining a linear classifier every time a new class appears.

To avoid meta-learning during the training phase and the linear evaluation during the testing phase, we focus on using a prototype classifier in the testing phase and training models in a standard classification manner. As we discuss in section 4, we found that when we directly constructed prototypes from the feature vectors extracted using pre-trained models and applied the nearest neighbor method as in the testing phase in ProtoNet , this does not perform as well as the linear-evaluation-based approach. We hypothesize that the reason is the difference between the loss function in ProtoNet and pre-trained models. As described in section 3, if we consider a prototype as a pseudo sample average of the features in each class, the loss function of ProtoNet can be considered as having a regularizing effect that makes it closer to the sample average of a class. Since standard classification training computes cross-entropy loss with additional linear projection to make the features linearly separable, the loss function does not have such an effect and can cause large within-class variance. Figure 1 shows a scatter plot of the features extracted using a neural network with two dimension output trained on CIFAR-10 with ProtoNet(1(a)) and cross-entropy loss with a linear projection layer(1(b)). This figure indicates that the features extracted using a model trained in a standard classification manner get distributed away from the origin and cause large within-class variance along the direction of the class mean vectors, while those of ProtoNet are more centered to its class means. This phenomenon is also observed in face recognition literature [5–8].

We now focus on a theoretical analysis for a prototype classifier. A recent study [2] analyzed an upper bound of risk by using a prototype classifier. The bound depends on the number of shots of a support set, between-class variance, and within-class variance. However, it has two drawbacks. The analysis requires the class-conditioned distribution of features to be Gaussian and to have the same covariance matrix among classes. Moreover, since the bound does not depend on the norm of the feature vectors, it is not clear from the bound what feature-transformation method can lead to performance improvement. Thus, we need to derive a novel bound for a prototype classifier.

Our contributions are threefold.

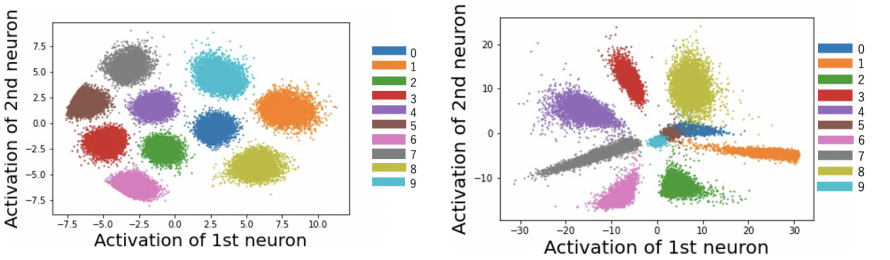
1. We relax the existing assumption; specifically, the bound does not require that the features be distributed in Gaussian distribution, and each covariance matrix does not have to be the same among classes.

2. We show that our generalization bound consists of three terms: (1) the variance of the norm of feature vectors, (2) the difference in the distribution shape constructed from each class embedding, and (3) the ratio of the within-class variance to the between-class variance.

3. From our theoretical analysis and empirical investigation, we found that reducing two terms is a critical factor to improve the performance of prototype classifiers: the variance of the norm of feature vectors and the ratio of the within-class variance to the between-class variance.

While it is empirically known that the $L2$-normalization of a feature vector and minimize the ratio of the within-class variance to the between-class variance can improve the performance of a prototype classifier, our work is the first to theoretically show that these transformation can improve the performance. The bound of Cao et al. [2] cannot explain the performance improvement of $L2$-normalization because $L2$-normalization does not reduce the ratio of the within-class variance to the between-class variance as shown in Section 4.

A prototype classifier can be applied to any trained feature extractor. It is simpler than linear-evaluation methods such as logistic regression and support vector machine (SVM) for three reasons. (1) A prototype classifier does not require training in the test phase. The training requires time and memory consumption. (2) Since the nearest neighbor method is applied on query data and class prototypes, the computation in the test phase does not depend on the number of classes while linear-evaluation methods do. (3) A prototype classifier requires fewer hyperparameters than linear-evaluation methods. Note that our goal is not to replace linear-evaluation methods with a prototype



(a) ProtoNet

(b) cross-entropy loss with linear projection layer

Figure 1: Distribution of features extracted using a neural network with two dimensional final layer trained on CIFAR-10 with (a): ProtoNet loss and (b): cross-entropy loss with linear projection layer. The ProtoNet features get distributed closer to its class center than the features extracted using the model trained on cross-entropy loss with linear projection layer.

classifier, but to show that a prototype classifier can be used as a counterpart of the linear-evaluation methods and can be a practically useful first step in tackling few-shot learning problems.

2 Related Work

We summarize related work by describing a prototype classifier with meta-learning, linear-evaluation-based approach without meta-learning, a prototype classifier without meta-learning and theoretical analysis related to the few-shot learning problem.

Prototype classifier with meta-learning On the basis of the hypothesis that features well distinguished in the training phase are also useful for classifying new classes, constructing one or multiple prototypes for classifying unseen examples is a widely used approach [1, 9–15]. Certain algorithms compute similarities between multiple prototypes and unseen examples by using their own modules, such as an attention mechanism [9, 14], a relation network [12], reweighting mechanisms by taking into account between-class and within-class interaction [11], and latent clusters [13]. Another line of research is transforming the space of extracted features to a better distinguishable space [16–19] or taking the variance of features into account [20] .

Update-based meta-learning In contrast to the prototype classifier with meta-learning, update-based meta-learning adjusts the model parameters to adapt to new classes. Model-agnostic meta-learning (MAML) and its variants [21–23] search for good initialization parameters that adapt to new classes with a few labeled data and a few update steps. These approaches require additional learning of hyper-parameters and training time; thus, they prevent quick adaptation to new classes.

Linear-evaluation-based approach without meta-learning Interestingly, recent studies have shown that training a new linear classifier with features extracted using a model trained with cross-entropy loss on base-dataset performs comparably with meta-learning based methods [3, 24]. A more effective method for training a new classifier in few-shot settings has been proposed [25, 26], such as calibrating distribution generated by a support set [25] , self-supervised learning [26], and distilling knowledge to obtain better embeddings [4]. However, similar to update-based meta-learning, these studies require additional hyper-parameters and training for application to new classes; thus if we tackle these points, we would have an alternative method that is easier and more convenient to use.

A prototype classifier without meta-learing Recent studies have shown that using a prototype classifier without meta-learning or a linear classifier can lead to comparable performance [24, 27]. Wang et al. [24] has empirically shown that Centering and normalizing the feature vectors can improve the performance of a prototype classifier. Another study empirically shows that applying NCA loss in the training phase can improve the performance of a prototype classifier [27]. However, their study requires a model to be trained with NCA loss while our studies do not require a model to be trained

with any specific loss. These studies do not show any theoretical insights on the performance of a prototype classifier and they do not compare with linear-evaluation-based methods.

Theoretical analysis of few-shot learning Even though much improvement has been empirically made on few-shot learning, theoretical analysis is scarce. In the context of meta-learning, Du et al. [28] provided a risk bound on the meta-testing phase that is related to the number of meta-training data and meta-testing data. Lucas et al. [29] derived information-theoretic lower-bounds on min-max rates of convergence for algorithms that are trained on data from multiple sources and tested on novel data. Cao et al. [2] derived a new bound on a prototype classifier and theoretically demonstrated that the mismatch regarding the number of shots in a support set between meta-training and meta-testing degrades the performance of prototypical networks, which has been only experimentally observed. However, their bound depends on two assumptions: the class-conditional distributions of features are Gaussian and have the same covariance matrix among classes. In contrast, we derive a novel bound that does not depend on any specific distribution.

3 Theoretical Analysis of Prototype Classifier in Terms of Variance of Norm of Feature Vectors

In this section, we first formulate our problem settings and explain the recent theoretical analysis on a prototype classifier. Next we provide our novel bound for a prototype classifier with the bound related to the variance of the norm of the feature vectors. Finally, we list several methods that can improve the performance of a prototype classifier based on our bound.

3.1 Problem Setting

Let $\mathcal{Y}$ be a space of a class, τ a probability distribution over $\mathcal{Y}$, $\mathcal{X}$ a space of input data, $\mathcal{D}$ a probability distribution over $\mathcal{X}$, $\mathcal{D}_y$ a probability distribution over $\mathcal{X}$ given a class y. We define $\mathcal{D}^{\otimes nk}$ and $\mathcal{D}_y^{\otimes n}$ by $\mathcal{D}_y^{\otimes n} = \Pi_{i=1}^{n}\mathcal{D}_y$ and $\mathcal{D}^{\otimes nk} \equiv \Pi_{i=1}^{n}\mathcal{D}_i^{\otimes k}$, respectively. We sample N classes from τ to form the N-way classification problem. Denote by K the number of annotated data in each class and $x \in \mathcal{X}, y \in \mathcal{Y}$ as input data and its class respectively. We define a set of support data of class c sampled from τ as $S_c = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \cap y_i = c\}_{i=1}^{K}$ and a set of support data in the N-way K-shot classification problem as $S = \bigcup_{c=1}^{N} S_c$. Suppose a feature extractor computes a function $\phi : \mathcal{X} \to \mathbb{R}^D$, where D is the number of the embedding dimensions. $\overline{\phi(S_c)}$ is defined by $\overline{\phi(S_c)} = \frac{1}{K}\sum_{\boldsymbol{x}\in S_c}\phi(\boldsymbol{x})$. Let Φ be a space of the extractor function ϕ. Denote by $\mathcal{M} : \Phi \times \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^{NK} \to \mathbb{R}^N$ a prototype classifier function that computes the probability of input x belonging to class c as follows.

$$\mathcal{M}(\phi, \boldsymbol{x}, S)_c = p_{\mathcal{M}}(y = c|\boldsymbol{x}, S, \phi) = \frac{\exp\left(-\|\phi(\boldsymbol{x}) - \overline{\phi(S_c)}\|^2\right)}{\sum_{l=1}^{N}\exp\left(-\|\phi(\boldsymbol{x}) - \overline{\phi(S_l)}\|^2\right)}, \tag{1}$$

where $\|v\|^2 = \sum_{d=1}^{D}\left(v^{(d)}\right)^2$, and $v^{(d)}$ is the d-th dimension of vector v. The prediction of an input x, denoted by $\hat{y} \in \mathcal{Y}$, is computed by taking argmax for $\mathcal{M}(\phi, \boldsymbol{x}, S)$, i.e., $\hat{y} = \mathrm{argmax}\mathcal{M}(\phi, \boldsymbol{x}, S)$. We denote by $\mathbb{E}_{z\sim q(z)}[g(z)]$ an operation to take the expectation of $g(z)$ over z distributed as $q(z)$, and we simply denote $\mathbb{E}_{z\sim q(z)}[g(z)]$ as $\mathbb{E}_z[g(z)]$ when z is obviously distributed on $q(z)$. We define $\mathrm{Var}_{z\sim q(z)}[g(z)]$ as an operation to take the variance of $g(z)$ over z distributed as $q(z)$. With $\mathbb{I}$ denoting the indicator function, we define the expected risk $\mathrm{R}_{\mathcal{M}}$ of a prototype classifier as

$$\mathrm{R}_{\mathcal{M}}(\phi) = \mathbb{E}_{S\sim\mathcal{D}^{\otimes nk}}\mathbb{E}_{c\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\mathbb{I}[\mathrm{argmax}\mathcal{M}(\phi, \boldsymbol{x}, S) \neq c]. \tag{2}$$

For simplicity, we now discuss the binary classification setting. We show a case of multi-class classification in Appendix A.5 due to lack of space.

Let c_1 and c_2 denote any pair of classes sampled from τ. We consider that a query data point x belongs to class c_1 and support sets S consist of class c_1's support set and c_2's support set. Then, equation 2 is written as follows.

$$\mathrm{R}_{\mathcal{M}}(\phi) = \mathbb{E}_{S,c_1,\boldsymbol{x}}[\mathbb{I}[\mathrm{argmax}\mathcal{M}(\phi, \boldsymbol{x}, S_{c_1} \cup S_{c_2}) \neq c_1]], \tag{3}$$
$$where \quad \mathbb{E}_{S,c_1,\boldsymbol{x}} = \mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}.$$

3.2 What Feature-Transformation Method Is Expected to Be Effective?

The current theoretical analysis for a prototype classifier [2] has the following two drawbacks (see Appendix A.2 for the details). The first is that the modeling assumption requires a class-conditioned distribution of the features to follow a Gaussian distribution with the same covariance matrix among classes. For example, when we use the ReLU activation function in the last layer, it is not normally distributed and the class-conditioned distribution does not have the same covariance matrix as shown in Figure 1. The second drawback is that it is not clear what feature-transformation method can reduce the upper bound. A feature-transformation method that maximizes the between-class variance and minimizes the within-class variance such as linear discriminant analysis (LDA) [30] and Embedding Space Transformation (EST) [2] can be expected to improve performance; however, it is not clear how the second term of the denominator changes.

From Figure 1, the distribution of each class feature stretches in the direction of its class-mean feature vector. This property is also observed in metric learning literatures [5, 6]. A model trained in the cross-entropy loss after linear projection from the embedding space to the class-probability space computes a probability of input x belonging to class c with a weight matrix $W \in \mathbb{R}^{D\times K}$ given by

$$p(y=c|\boldsymbol{x}, W, \phi) = \frac{\exp\left(\phi(\boldsymbol{x})^\top W_c\right)}{\sum_{j=1}^{N}\exp\left(\phi(\boldsymbol{x})^\top W_j\right)}. \tag{4}$$

Comparing equation 4 with equation 1, we found that equation 4 does not have a regularization term similar to the one appearing in equation 1 that forces the features to be close to its class-mean feature vector. This implies that the features extracted using a model trained with the cross-entropy loss using equation 4 are less close to its class-mean feature vector than the ProtoNet loss on equation 1. Through this observation and the property mentioned above, we hypothesize that normalizing the norm of the feature vectors can push the features to each class-mean feature vector and can boost the performance of a prototype classifier trained with cross-entropy loss on equation 4.

3.3 Relating Variance of Norm to Upper Bound of Expected Risk

To understand what effect the variance of the norm of the feature vectors has on the performance of a prototype classifier, we analyze how variance contributes to the expected risk when an embedding function ϕ is fixed. The following theorem provides a generalization bound for the expected risk of a prototype classifier in terms of the variance of the norm of the feature vectors computed using a feature extractor.

Theorem 1. *Let $\mathcal{M}$ be an operation of a prototype classifier for binary classification defined by equation 1. For $\mu_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\phi(\boldsymbol{x})]$, $\Sigma_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[(\phi(\boldsymbol{x})-\mu_c)(\phi(\boldsymbol{x})-\mu_c)^\top]$, $\mu = \mathbb{E}_{c\sim\tau}[\mu_c]$,$\Sigma = \mathbb{E}_{c\sim\tau}[(\mu_c-\mu)(\mu_c-\mu)^\top]$, and $\Sigma_\tau = \mathbb{E}_{c\sim\tau}[\Sigma_c]$, if $\phi(\boldsymbol{x})$ has the variance of its norm, then the misclassification risk of the prototype classifier for binary classification $\mathrm{R}_{\mathcal{M}}$ satisfies*

$$\mathrm{R}_{\mathcal{M}}(\phi) \leq 1 - \frac{4(\mathrm{Tr}(\Sigma))^2}{\mathbb{E}\mathrm{V}[h_{L2}(\phi(\boldsymbol{x}))] + \mathrm{V}_{\mathrm{Tr}}(\Sigma_{c_1}) + \mathrm{V}_{\mathrm{wit}}(\Sigma_\tau, \Sigma, \boldsymbol{\mu}) + \mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_{c_1}, \boldsymbol{\mu}_{c_2})}, \tag{5}$$

$$\textit{where} \quad \mathbb{E}\mathrm{V}[h_{L2}(\phi(\boldsymbol{x}))] = \frac{4}{K}\mathbb{E}_{c\sim\tau}\left[\mathrm{Var}_{\boldsymbol{x}\sim\mathcal{D}_c}\left[\|\phi(\boldsymbol{x})\|^2\right]\right], \tag{6}$$

$$\mathrm{V}_{\mathrm{Tr}}(\Sigma_{c_i}) = \left(\frac{4}{K} + \frac{2}{K^2}\right)\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_i}\right)\right], \tag{7}$$

$$\begin{aligned}\mathrm{V}_{\mathrm{wit}}\left(\Sigma_\tau, \Sigma, \boldsymbol{\mu}\right) = &\frac{8}{K}\left(\mathrm{Tr}(\Sigma_\tau)\right)\left(\mathrm{Tr}(\Sigma) + \boldsymbol{\mu}^\top\boldsymbol{\mu}\right) + 4\left(\mathrm{Tr}(\Sigma) + \boldsymbol{\mu}^\top\boldsymbol{\mu}\right)^2 \\ &+ 4\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_1}\right)\right],\end{aligned} \tag{8}$$

$$\mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_{c_1}, \boldsymbol{\mu}_{c_2}) = \mathbb{E}_{c_1,c_2}\left[\left(\left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)^\top\left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)\right)^2\right]. \tag{9}$$

Remark. *The term $\mathbb{E}\mathrm{V}[h_{L2}(\phi(\boldsymbol{x}))]$ is the variance of the norm of the feature vectors. The term $\mathrm{V}_{\mathrm{Tr}}(\Sigma_{c_1})$ is the variance of the summation with the diagonal element of the covariance matrix from each class embedding; it can be interpreted as the difference in the distribution shape constructed*

from each class embedding. The term $\mathrm{V}_{\mathrm{wit}}(\Sigma_\tau, \Sigma, \boldsymbol{\mu})$ *is related to the within-class variance. The last term of equation 8 is approximately linear with* $\frac{\mathrm{Tr}(\Sigma_{c_1})}{\mathrm{Tr}(\Sigma)}$ *because* $\frac{(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})}{\mathrm{Tr}(\Sigma)}$ *is supposed to be constant. Thus* $\frac{\mathrm{V}_{\mathrm{wit}}(\Sigma_\tau,\Sigma,\boldsymbol{\mu})}{\mathrm{Tr}(\Sigma)}$ *is a secondary expression for the ratio of the within-class variance to the between-class variance. The term* $\mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_{c_1}, \boldsymbol{\mu}_{c_2})$ *is the expectation of the Euclidean distance between the class-mean vectors .*

This bound has the following properties.

1. Its derivation does not require the features to be distributed in Gaussian distribution and the class-conditioned covariance matrix does not have to be the same among classes
2. The bound can decrease when any of four statistics decreases with fixed between-class variance (Σ): (i) the variance of the norm of the feature vectors, as discussed in Section 3.2, (ii) the difference in the distribution shape constructed from each class embedding, (iii) the ratio of the within-class variance (Σ_τ) to the between-class variance (Σ), and (iv) the Euclidean distance between the class-mean vectors.

As a result, our bound loosens the modeling assumption of Theorem2 in Appendix A.2 and has its property. We show the proof in Appendix A.4.

3.4 Feature-Transformation Methods

We hypothesize from Theorem 1 that in addition to a feature-transformation method related to equation 6, lowering the ratio of between-class variance to within-class variance can improve the performance of a prototype classifier. As shown in Figure 2 left in experimantal analysis, the value of equation 7 is relatively small compared to equation 6 and equation 8. Regarding equation 9, the ratio of the Euclidean distance between the class mean vectors and the between-class variance is supposed to be constant. Thus, we focus on transformation methods related to equation 6 and equation 8. We analyze four feature-transformation methods: L_2-normalization (L_2-norm), EST,LDA, EST+L_2-norm and LDA+L_2-norm. We give the details on each method in Section A.1 due to lack of space.

4 Experimental Evaluation

In this section, we experimentally analyzed the effectiveness of the feature-transformation methods mentioned in Section3.4 under two scenarios: (1) standard object recognition and (2) cross-domain adaptation. The center loss [5] and affinity loss [31] have been proposed to efficiently pull the features of the same class to their centers in the training phase; however,our goal is not to achieve SOTA but, through comparison with existing studies, show that we can achieve the same level of performance without fine-tuning and thus we focus on a widely used pre-trained model in the experiments following the line of studies Chen et al. [3] and Tian et al. [4].

4.1 Datasets and Evaluation Protocol

For standard object recognition, we use the *mini*ImageNet dataset, *tiered*ImageNet dataset, the CIFARFS dataset, and the FC100 dataset. **miniImageNet.** The *mini*ImageNet dataset [9] is a standard bench-mark for few-shot learning algorithms in recent studies. It contains 100 classes randomly sampled from ImageNet. Each class contains 600 images. We follow a widely used splitting protocol [32] to split the dataset into $64/16/20$ for training/validation/testing respectively.

tieredImageNet The *tiered*ImageNet dataset [33] is another subset of ImageNet but has 608 classes. These classes are grouped into 34 higher level categories in accordance with the ImageNet hierarchy and these categories are split into 20 training (351 classes), 6 validation (97 classes), 8 testing categories (160 classes). This splitting protocol ensures that the training set is distinctive enough from the testing set and makes the problem more realistic than miniImageNet since we generally cannot assume that test classes will be similar to those seen in training.

CIFAR-FS The CIFAR-FS dataset [34] is a recently proposed fewshot image classification bench-mark, consisting of all 100 classes from CIFAR-100 [35]. The classes are randomly split into 64, 16, and 20 for meta-training, meta-validation, and meta-testing respectively.

Table 1: Classification accuracies with ResNet12 on *mini*ImageNet, *tiered*ImageNet, CIFARFS, and FC100 of ProtoNet, linear-evaluation-based methods [3], centering with L_2-norm [24], and ours. The Baseline without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline are in bold. Regarding to Baseline++, Baseline++ without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline++ are in bold. All the methods are our reimplementation .

	*mini*ImageNet		*tiered*ImageNet		CIFAR-FS		FC100	
	1-shot	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
methods with meta-learning or linear-evaluation								
ProtoNet[1]	53.48 ± 0.62	73.56 ± 0.65	55.40 ± 0.98	77.67 ± 0.70	63.26 ± 0.99	79.91 ± 0.72	35.56 ± 0.77	51.12 ± 0.71
Baseline [3]	54.54 ± 0.80	76.50 ± 0.62	61.67 ± 0.92	81.62 ± 0.64	60.62 ± 0.85	79.79 ± 0.70	39.72 ± 0.68	56.04 ± 0.76
A prototype classifier with feature-transformation methods								
Baseline-w/o-linear , centering+L_2-norm[24]	**58.96 ± 0.83**	**75.98 ± 0.62**	**64.88 ± 0.86**	80.42 ± 0.64	**62.51 ± 0.86**	**79.35 ± 0.66**	**41.58 ± 0.74**	**55.82 ± 0.76**
Baseline-w/o-linear	46.36 ± 0.58	73.97 ± 0.62	50.60 ± 0.87	78.10 ± 0.67	45.91 ± 0.89	75.81 ± 0.79	31.60 ± 0.61	52.50 ± 0.74
Baseline-w/o-linear , L_2-norm	**59.60 ± 1.12**	**77.12 ± 0.44**	**64.24 ± 0.89**	**81.93 ± 0.66**	**63.78 ± 0.87**	**80.14 ± 0.75**	**40.34 ± 0.71**	**56.61 ± 0.71**
Baseline-w/o-linear , EST	51.40 ± 0.83	73.47 ± 0.62	50.94 ± 0.93	78.47 ± 0.68	56.08 ± 0.94	76.64 ± 0.76	**45.90 ± 0.85**	**64.32 ± 0.78**
Baseline-w/o-linear , EST+L_2-norm	**57.34 ± 0.84**	**76.90 ± 0.62**	**64.14 ± 0.91**	**81.30 ± 0.64**	**65.19 ± 0.93**	**81.13 ± 0.69**	**49.54 ± 0.91**	**64.12 ± 0.69**
Baseline-w/o-linear , LDA	51.40 ± 0.83	73.47 ± 0.62	50.94 ± 0.93	78.47 ± 0.68	56.08 ± 0.94	76.80 ± 0.76	**45.90 ± 0.85**	**64.32 ± 0.78**
Baseline-w/o-linear , LDA+L_2-norm	**60.29 ± 0.80**	**75.99 ± 0.66**	**64.16 ± 0.90**	**81.96 ± 0.65**	**65.57 ± 0.89**	**80.80 ± 0.68**	**50.97 ± 0.81**	**64.91 ± 0.76**
methods with meta-learning or linear-evaluation								
Baseline++ [3]	56.33 ± 0.81	74.62 ± 0.60	63.02 ± 0.91	81.07 ± 0.69	65.43 ± 0.95	80.18 ± 0.75	36.01 ± 0.64	50.73 ± 0.72
A prototype classifier with feature-transformation methods								
Baseline++-w/o-linear , centering+L_2-norm[24]	**57.50 ± 0.81**	74.00 ± 0.60	**63.31 ± 0.91**	79.19 ± 0.68	**65.76 ± 0.90**	**79.95 ± 0.73**	**37.51 ± 0.71**	50.38 ± 0.72
Baseline++-w/o-linear	41.18 ± 0.76	69.48 ± 0.66	46.52 ± 0.87	73.85 ± 0.70	48.28 ± 0.89	72.79 ± 0.74	29.72 ± 0.57	46.85 ± 0.67
Baseline++-w/o-linear , L_2-norm	**57.96 ± 0.80**	**75.38 ± 0.61**	**65.36 ± 0.90**	**81.08 ± 0.69**	**66.52 ± 0.92**	**80.23 ± 0.70**	**37.16 ± 0.67**	**51.10 ± 0.71**
Baseline++-w/o-linear , EST	47.11 ± 0.81	69.01 ± 0.63	52.01 ± 0.90	68.36 ± 0.82	52.72 ± 0.95	72.10 ± 0.74	**37.66 ± 0.68**	**53.34 ± 0.72**
Baseline++-w/o-linear , EST+L_2-norm	**58.32 ± 0.81**	**75.19 ± 0.63**	**64.51 ± 0.96**	**80.38 ± 0.70**	**66.01 ± 0.90**	78.78 ± 0.71	**42.98 ± 0.80**	**58.13 ± 0.72**
Baseline++-w/o-linear , LDA	47.42 ± 0.80	68.43 ± 0.69	47.76 ± 0.90	74.41 ± 0.75	52.19 ± 0.97	70.91 ± 0.76	**40.40 ± 0.75**	**57.23 ± 0.75**
Baseline++-w/o-linear , LDA+L_2-norm	**58.26 ± 0.87**	**75.13 ± 0.66**	**65.10 ± 0.92**	**80.39 ± 0.69**	**67.31 ± 0.96**	**79.69 ± 0.72**	**44.83 ± 0.83**	**58.67 ± 0.74**

Table 2: Classification accuracies with ResNet12 on *mini*ImageNet, *tiered*ImageNet, CIFARFS, and FC100 of methods in current studies and ours. The best performing methods and any other runs within 95% confidence margin are in bold. Methods with † show the results of our reimplementation following Section 4.2

	*mini*ImageNet		*tiered*ImageNet		CIFAR-FS		FC100	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
methods with meta-learning or linear-evaluation								
MetaOptNet [36]†	59.65 ± 0.60	75.53 ± 0.62	63.01 ± 0.72	80.01 ± 0.64	66.03 ± 0.77	79.80 ± 0.54	40.1 ± 0.63	53.3 ± 0.62
TapNet [17]	**61.65 ± 0.15**	76.36 ± 0.10	63.08 ± 0.15	80.26 ± 0.12	-	-	-	-
A prototype classifier with feature-transformation methods								
Baseline-w/o-linear , L_2-norm	59.60 ± 1.12	**77.12 ± 0.44**	64.24 ± 0.89	**81.93 ± 0.66**	63.78 ± 0.87	80.14 ± 0.75	40.34 ± 0.71	56.61 ± 0.71
Baseline-w/o-linear , EST+L_2-norm	57.34 ± 0.84	**76.90 ± 0.62**	64.14 ± 0.91	81.30 ± 0.64	65.19 ± 0.93	**81.13 ± 0.69**	49.54 ± 0.91	**64.12 ± 0.69**
Baseline-w/o-linear , LDA+L_2-norm	60.29 ± 0.80	75.99 ± 0.66	64.16 ± 0.90	**81.96 ± 0.65**	65.57 ± 0.89	**80.80 ± 0.68**	**50.97 ± 0.81**	**64.91 ± 0.76**
Baseline++-w/o-linear , L_2-norm	57.96 ± 0.80	75.38 ± 0.61	**65.36 ± 0.90**	81.08 ± 0.69	66.52 ± 0.92	80.23 ± 0.70	37.16 ± 0.67	51.10 ± 0.71
Baseline++-w/o-linear , EST+L_2-norm	58.32 ± 0.81	75.19 ± 0.63	64.51 ± 0.96	80.38 ± 0.70	66.01 ± 0.90	78.78 ± 0.71	42.98 ± 0.80	58.13 ± 0.72
Baseline++-w/o-linear , LDA+L_2-norm	58.26 ± 0.87	75.13 ± 0.66	**65.10 ± 0.92**	80.39 ± 0.69	**67.31 ± 0.96**	79.69 ± 0.72	44.83 ± 0.83	58.67 ± 0.74

FC100 The FC100 dataset [36] is another dataset derived from CIFAR-100 [35], containing 100 classes which are grouped into 20 categories. These categories are split into 12 categories for training (60 classes), from 4 categories for validation (20 classes), 4 categories for testing (20 classes). This splitting protocol is similar to the protocol used in tieredImageNet so that the training set is distinctive enough from the testing set and makes the problem more realistic than CIFAR-FS.

For cross-domain scenario, we use the *mini*ImageNet dataset during pre-training stage and we use the CUB-200-2011 dataset [37], a.k.a CUB during testing stage.

CUB The CUB dataset contains 200 classes and 11,788 images in total. Following the evaluation protocol of Chen et al. [3], we split the dataset into $64/16/20$ for training/validation/testing respectively.

4.2 Implementation Details

We compared the feature-transformation methods against ProtoNet [1], Baseline, and Baseline++ [3]. For Baseline and Baseline++, we trained the linear projection layer on a support set. The difference between Baseline and Baseline++ is that the norm of the linear projection layer and features in Baseline++ are normalized to be constant. We call Baseline and Baseline++ as "linear evaluation" methods. We also compared with the feature-transformation method proposed in a previous study [24]. In that study they transformed the features so that the mean of all features was the origin before L_2-normalization and then used a prototype classifier without training a new linear classifier. We call this operation centering+L_2-norm. We re-implemented these methods following the training

procedure in a previous study [3]. In the pre-training stage, where the cross-entropy loss was used and meta-learning was not used, we trained 400 epochs with a batch size of 16. In the meta-training stage for ProtoNet, we trained $60,000$ episodes for 1-shot and $40,000$ episodes for 5-shot tasks. We used the validation set to select the training episodes with the best accuracy. We evaluated the model every 5 epochs, not every epoch as experimented in Chen et al. [3] , and we performed early stopping when the validation score does not improve for 50 epochs. In each episode, we sampled N classes to form N-way classification (in meta-training N=20 and meta-testing N=5 following the original study for ProtoNet [1]). For each class, we selected K labeled instances as our support set and 16 instances for the query set for a K-shot task.

In the linear evaluation or meta-testing stage for all methods, we averaged the results over 600 trials. In each trial, we randomly sampled 5 classes from novel classes, and in each class, we also selected K instances for the support set and 16 for the query set. For Baseline and Baseline++, we used the entire support set to train a new classifier for 100 iterations with a batch size of 4. With ProtoNet, we used the models trained in the same shots as meta-testing stage since a mismatch in the number of shots between meta-training and meta-testing degrades performance [2]. All methods were trained from scratch, and the Adam optimizer with an initial learning rate of 10^{-3} was used. We applied standard data augmentation including random crop, horizontal-flip, and color jitter in both the training stages. We used a ResNet12 network and ResNet18 network following the previous study [3, 17].

For LDA we set $\lambda = 0.0001$ for equation 11 in Appendix A.1 and for EST we set the dimensions of the projected space to 60 following the settings of the original study [2]. Following the procedure of EST, we calculate the transformation matrix of LDA based on the features of the training dataset.

4.3 Results

We present the experimental results of standard object recognition in Table 1 on the basis of backbones with ResNet12 for a comprehensive comparison. We show the result of backbones with ResNet18 in Table 3. We present the discussion on cross-dataset scenario and the result of the scenario in A.7.

Comparison of feature-transformation methods with ProtoNet, linear evaluation methods, and centering+L_2-norm From Table 1, we can observe that the prototype classifier with L_2-norm, EST+L_2-norm, LDA+L_2-norm performs comparably with ProtoNet and the linear-evaluation-based approach in all settings. Comparing the feature-transformation methods described in Section 3.4 with centering+L_2-norm, centering+L_2-norm can slightly improve the performance of the prototype classifier in several 1-shot settings . However, in 5-shot settings, the boost decreases and even performs worse than linear-evaluation methods, e.g. *mini*ImageNet and *tiered*ImageNet.

Comparison among feature-transformation methods In the 1-shot setting, although EST falls short of ProtoNet and the linear-evaluation-based approach, it also improves the performance of a prototype classifier. The performance gain of both L_2-norm, EST, LDA, EST+L_2-norm and LDA + L_2-norm decrease when the number of shots increases. This phenomenon can be explained through Theorem 1. The term relating to the variance of the norm and the ratio of the within-class variance to the between-class variance depends on K. Since the term diminishes as K increases, the performance gain of the feature-transformation methods decreases.

Comparison of feature-transformation methods with current studies Table 2 shows the performance of MetaOptNet [36], TapNet [17] and feature-transformation methods that performs comparable with ProtoNet or linear-evaluation-based methods in Table 1. We can observe that a prototype classifier can achieve comparable performance with current studies in most of the settings with the feature transformations. Moreover, we can further boost the performance of a prototype classifier by combining L_2-norm and the methods to reduce the ratio of the within-class variance to the between-class variance, e.g. CIFAR-FS and FC100.

Discussion on feature-transformation methods LDA and EST in the 5-shot setting do not improve the performance so much compared with L_2-norm variants while in the 1-shot settings, LDA and EST improve the performance. This is because, in the 5-shot settings, the values computed from equation 8 get smaller with larger K, and the effect of equation 8 on the risk of a prototype classifier in the 1-shot settings is larger. Especially, EST and LDA outperform L_2-norm in FC100. We found from Figure 2 that the features of FC100 show the largest ratio of the within-class to between-class variance

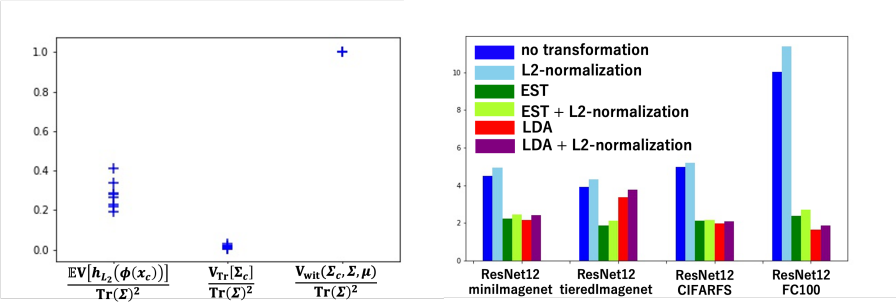


Figure 2: Left: We plotted the values shown in equations 6,7,8 divided by $\mathrm{Tr}(\Sigma)^2$. The values are computed from test-split of each dataset with ResNet12 and ResNet18. We scaled the values so that each dataset's $\frac{\mathrm{V}_{\mathrm{wit}}(\Sigma_\tau,\Sigma,\boldsymbol{\mu})}{\mathrm{Tr}(\Sigma)^2}$ to be 1 for simplicity. Right: We plotted the ratio of the within-class variance to the between-class variance ($\frac{\mathrm{Tr}(\Sigma_\tau)}{\mathrm{Tr}(\Sigma)}$) before and after applying L_2-norm, EST, LDA, EST + L_2-norm and LDA+L_2-norm of each dataset.

among all datasets. Thus the method of reducing the ratio works better with FC100 features than with any other dataset's features. As discussed in Section 4.3, the combination of feature-transformation methods that reduce the variance of the norm and the ratio of the within-class variance to the between-class variance can further boost the performance of a prototype classifier. This indicates that reducing the term equation 6 and equation 8 is an important factor for the performance of a prototype classifier.

4.4 The comparison of the ratio of the within-class variance to the between-class variance before and after applying L_2norm, EST and LDA

We show the ratio of the within-class variance to the between-class variance before and after applying L_2-norm, EST, LDA, EST + L_2-norm, LDA + L_2-norm in Figure 2 right. We calculated the ratio of the between-class variance to each class's variance of each dataset. This figure shows that L_2-normalization slightly changes the ratio while EST and LDA can reduce the ratio. The combination of EST and L_2-norm or LDA and L_2-norm can reduce the ratio while decreasing the variance of the norm of feature vectors to 0. Therefore, EST + L_2-norm and LDA + L_2-norm performs better than L_2-norm. Moreover, L_2-norm does not affect the bound of Cao et al. [2] because their bound depends on the ratio of the between-class variance to each class's variance as shown in 2. Thus the bound of Cao et al. [2] cannot explain the performance improvement of a prototype classifier with L_2-norm while our bound can.

5 Conclusion

We theoretically and experimentally analyzed how the variance of the norm of feature vectors affects the performance of a prototype classifier. We derived a generalization bound that does not require the features to be distributed in Gaussian distribution and the class-conditioned covariance matrix does not have to be the same among classes. We found that using EST+L_2-norm makes the classifier comparable with ProtoNet and the linear-evaluation-based approach. Our experiments show that when the number of shots in a support set increases, the performance gain from a feature-transformation method decreases, which is consistent with the results of theoretical analysis.

One limitation of this paper is that, in the first place, the prototype classifier works well under the assumption that the data distribution for each class is unimodal and isotropic modeling. However, we consider that this assumption may be sound in practice because the multimodality of the data distribution in a class is typically caused by the way the data is annotated. Since the number of a support set is small in few-shot settings (for example, in 5-way 5-shot, the number of labeled data is 25), we can easily check the labels and re-label them so that they don't become multi-modal.

Broader impact. We believe that our work shows that a prototypical classifier is expected to be a practically useful first step in tackling few-shot learning problems because of its simplicity. The progress in few-shot learning can impact importnant problems such as medical images and re-identification problems. We also recognize our work might constitute a threat that authoritarian entities deploy few-shot learning algorithms for surveillance.

References

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

[2] Tiansi Cao, Marc T.Law, and Sanja Fidler. A theoretical analysis of the number of shots in few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2019.

[4] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision (ECCV)*, 2020.

[5] Yandong Wen, Kaipeng Zhang, Zhifeng Li, , and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision (ECCV)*, 2016.

[6] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[7] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2018.

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019.

[9] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[10] Frederik Pahde, Mihai Puscas, Tassilo Klein, and Moin Nabi. Multimodal prototypical networks for few-shot learning. In *Winter Conference on Applications of Computer Vision(WACV)*, pages 2644–2653, 2021.

[11] Zhong Ji, Xingliang Chai, Yunlong Yu, and Zhongfei Zhang. Reweighting and information-guidance networks for few-shot learning. In *Neurocomputing*, volume 423, pages 13–23, 2021.

[12] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] Kelsey R. Allen, Evan Shelhamer, Hanul Shin, and Joshua B. Tenenbaum. Infinite mixture prototypes for few-shot learning. *PMLR*, pages 232–241, 2019.

[14] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *Advances in Neural Information Processing Systems*, pages 4003–4014, 2020.

[15] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5822–5830, 2018.

[16] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020.

[17] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[18] Debasmit Das, J. H. Moon, and C. S. George Lee. Few-shot image recognition with manifolds. In *International Symposium on Visual Computing (ISVC)*, pages 3–14, 2020.

[19] Debasmit Das and C.S. George Lee. A two-stage approach to few-shot learning for image recognition. In *IEEE Transactions on Image Processing*, volume 29, 2020.

[20] Peyman Bateni, Raghav Goyal, Frank Wood Vaden Masrani, , and Leonid Sigal. Improved few-shot visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3–14, 2020.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135. PMLR, 2017.

[22] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 9516–9527, 2018.

[23] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradient. *PMLR*, pages 1126–1135, 2017.

[24] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. volume abs/:1911.04623, 2019.

[25] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. In *International Conference on Learning Representations(ICLR)*, 2021.

[26] Cheng Perng Phoo and Bharath Hariharan. Self-training for few-shot transfer across extreme task difference. In *International Conference on Learning Representations(ICLR)*, 2021.

[27] Steinar Laenen and Luca Bertinetto. On episodes, prototypical networks, and few-shot learning. In *Advances in Neural Information Processing Systems*, 2021.

[28] Simon Shaolei Du, Wei Hu, Sham M. Kakade, Jason D. Lee, and Qi Lei. Few-shot learning via learning the representation, provably. In *International Conference on Learning Representations(ICLR)*, 2021.

[29] James Lucas, Mengye Ren, Irene Raissa KAMENI KAMENI, Toniann Pitassi, and Richard Zemel. Theoretical bounds on estimation error for meta-learning. In *International Conference on Learning Representations(ICLR)*, 2021.

[30] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., 1990.

[31] Munawar Hayat, Salman Khan, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Max-margin class imbalanced learning with gaussian affinity. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6469–6479, 2019.

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

[33] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised fewshot classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[34] Luca Bertinetto, João Henriques, Philip H.S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

[35] Alex Krizhevsky, Vinod Nair, , and Geoffrey Hinton. Cifar-100.

[36] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10657–10665, 2019.

[37] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[38] Douglas P. Wiens. On moments of quadratic forms in non-spherically distributed variables. In *A Journal of Theoretical and Applied Statistics*, volume 23, pages 265–270, 1992.

[39] Michael D. Perlman. Jensen's inequality for a convex vector-valued function on an infininite-dimensional space. In *J Multivar Anal*, pages 52–65, 1974.

Checklist

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See contribution part in Section 1.

(b) Did you describe the limitations of your work? [Yes] See Section 5

(c) Did you discuss any potential negative societal impacts of your work? [Yes] Seee Section 5

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] See Theorem 1

(b) Did you include complete proofs of all theoretical results? [Yes] See Section A.4

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We only show the implementation details.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.2

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section A.6

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We omit these information because total amount of compute is not so large and not our intrests.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.

(b) Did you mention the license of the assets? [No] We can confirm the license in the cited paper.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We used widely used benchmark dataset.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Detail of Feature-Transforming Methods

L_2 normalization (L_2-norm) From equation 6, normalizing the norm of the feature vectors can improve the performance of a prototype classifier. We denote a function normalizing the norm as ψ_{L2} given by

$$\psi_{L2}(\phi(\boldsymbol{x})) = \frac{\phi(\boldsymbol{x})}{\|\phi(\boldsymbol{x})\|}. \tag{10}$$

LDA From equation 8, decreasing the ratio of the within-class variance to the between-class variance can improve the performance of a prototype classifier. LDA [30] is widely used to search for a projection space that maximizes the between-class variance and minimizes the within-class variance. It computes the eigenvectors of a matrix $\hat{\Sigma_\tau}^{-1}\hat{\Sigma}$, where $\hat{\Sigma}$ is the covariance matrix of prototypes and $\hat{\Sigma_\tau}$ is the class-conditioned covariance matrix. Since the number of data is small in few-shot settings, $\hat{\Sigma_\tau}^{-1}$ cannot be estimated stably and we add a regularizer term to $\hat{\Sigma_\tau}$ and define it as $\hat{\Sigma_{\tau\mathrm{reg}}}$.

$$\hat{\Sigma_{\tau_{\mathrm{reg}}}} = \hat{\Sigma_\tau} + \lambda I, \tag{11}$$

where $\lambda \in R^1, I \in R^{D\times D}$ is identity matrix.

EST Since computation of LDA is unstable, we also analyze the effect of EST [2]. EST computes eigenvectors of a matrix $\hat{\Sigma} - \rho\hat{\Sigma_\tau}$: the difference between the covariance matrix of the class mean vectors and the class mean covariance matrix with weight parameter ρ. Similar to LDA, EST also searches for the projection space that maximizes the Σ and minimizes the Σ_τ.

EST+L_2-norm We hypothesize that the combination of the transforming methods can improve the performance of a prototype classifier independently from each other. Specifically, we focus on reducing equation 6 and equation 8 by combining **EST** and **L_2-norm**. We first apply EST to reduce equation 8 and after that we apply L_2-norm to reduce equation 6. At the end of the operation we want the variance of the norm to be 0, thus we apply EST and after that we apply L_2-norm.

LDA+L_2-norm We focus on reducing equation 8 and equation 7 by combining **LDA** and **L_2-norm**. Following the similar procedure of **EST+L_2-norm**, we first apply LDA and after that we apply L_2-norm.

A.2 Existing Upper Bound on Expected Risk for Prototype Classifier

To analyze the behavior of a prototype classifier, we start from the current study [2]. The following theorem is the upper bound of the expected risk of prototypical networks with the next conditions.

- The probability distribution of an extracted feature $\phi(\boldsymbol{x})$ given its class $y = c$ is Gaussian i.e $\mathcal{D}_y = \mathcal{N}(\mu_c, \Sigma_c)$, where $\mu_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\phi(\boldsymbol{x})]$ and $\Sigma_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[(\phi(\boldsymbol{x}) - \mu_c)(\phi(\boldsymbol{x}) - \mu_c)^\top]$.
- All class-conditioned distributions have the same covariance matrix, i.e., $\forall(c, c'), \Sigma_c = \Sigma_{c'}$.

Theorem 2 (Cao et al. [2])**.** *Let $\mathcal{M}$ be an operation of a prototype classifier on binary classification defined by equation 1. Then for $\mu = \mathbb{E}_{c\sim\tau}[\mu_c]$ and $\Sigma = \mathbb{E}_{c\sim\tau}[(\mu_c - \mu)(\mu_c - \mu)^\top]$, the misclassification risk of the prototype classifier on binary classification $\mathrm{R}_\mathcal{M}$ satisfies*

$$\mathrm{R}_\mathcal{M}(\phi) \le 1 - \frac{4\,\mathrm{Tr}(\Sigma)^2}{8(1+1/k)^2\,\mathrm{Tr}\,(\Sigma_c^2) + 16(1+1/k)\,\mathrm{Tr}\,(\Sigma\Sigma_c) + \mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_{c_1}, \boldsymbol{\mu}_{c_2})}, \tag{12}$$

where $\mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_{c_1}, \boldsymbol{\mu}_{c_2}) = \mathbb{E}_{c_1,c_2}\left[((\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})\top(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}))^2\right]$.

We show the detail of the derivation in Appendix A.3.

A.3 Reviewing Derivation Details of Theorem 2 (Cao et al. [2])

We briefly review the derivation of Theorem 2. In prototype classifier, from equation 1 and equation 3, $R_{\mathcal{M}}$ is written with sigmoid function σ as follows:

$$\begin{aligned} R_{\mathcal{M}}(\phi) =& \Pr_{c_1,c_2\sim\tau,\boldsymbol{x}\sim\mathcal{D}_{c_1},S\sim\mathcal{D}\otimes 2K}\left(\sigma(\|\phi(\boldsymbol{x})-\overline{\phi(S_{c_2})}\|-\|\phi(\boldsymbol{x})-\overline{\phi(S_{c_1})}\|)\leq\frac{1}{2}\right) \\ =& \Pr(\alpha<0), \end{aligned} \tag{13}$$

where $\alpha \triangleq \|\phi(\boldsymbol{x})-\overline{\phi(S_{c_2})}\|-\|\phi(\boldsymbol{x})-\overline{\phi(S_{c_1})}\|$. we bound equation 13 with expectation and variance of α by following proposition.

Proposition 1. *From the one-sided Chebyshev's inequality, it immediately follows that:*

$$\mathrm{R}_{\mathcal{M}}(\phi)=\Pr(\alpha<0)\leq 1-\frac{\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]^2}{\mathrm{Var}_{S,c_1,c_2,\boldsymbol{x}}[\alpha]+\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]^2}. \tag{14}$$

equation 13 can be further write down as follows by Law of Total Expectation

$$\begin{aligned} \mathrm{Var}_{S,c,\boldsymbol{x}}(\alpha) &= \mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\alpha^2]-\left(\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]\right)^2 \\ &= \mathbb{E}_{c_1,c_2}\mathbb{E}_{\boldsymbol{x},S}[\alpha^2|c_1,c_2]-\left(\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]\right)^2 \\ &= \mathbb{E}_{c_1,c_2}[\mathrm{Var}_{\boldsymbol{x},S}(\alpha|c_1,c_2)+\mathbb{E}_{\boldsymbol{x},S}[\alpha|c_1,c_2]^2]-\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}\mathbb{E}_{S\sim\mathcal{D}_{\otimes 2K}}[\alpha]^2. \end{aligned}$$

Therefore,

$$\Pr(\alpha<0)\leq 1-\frac{\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\alpha]^2}{\mathbb{E}_{c_1,c_2}[\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S}[\alpha]+\mathbb{E}_{\boldsymbol{x}\sim D_{c_1},S}[\alpha]^2]}. \tag{15}$$

We write down the expection and variance of α with following Lemmas 3 and 4.

Lemma 3. *Under the same notation and assumptions as Theorem 2, then,*

$$\begin{aligned} \mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha] &= (\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2}) \\ \mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2K}}[\alpha] &= 2\mathrm{Tr}(\Sigma). \end{aligned}$$

Lemma 4. *Under the same notation and assumptions as Theorem 2, then,*

$$\mathbb{E}_{c_1,c_2}\left[\mathrm{Var}_{\boldsymbol{x},S}\left[\alpha|c_1,c_2\right]\right]\leq 8\left(1+\frac{1}{K}\right)\mathrm{Tr}\left(\Sigma_\tau((1+\frac{1}{K})\Sigma_\tau+2\Sigma)\right). \tag{16}$$

The proofs of the above lemmas are in the current study [2].

With Proposition 1 and Lemma 3, Lemma 4, we obtain Theorem 2.

A.4 Derivation Details of Theorem 1

We will describe the detailed derivation of Theorem 1 in this section. Our derivation is different from Cao et al. [2]'s study in the following points.

1. We re-derived Lemma 3 because the term of the difference between the trace of the class covariance matrices is erased in the lemma. This term cannot omit in our derivation since we do not assume the class covariance matrix to be the same among classes.
2. We re-derived the bound on the variance of squared Euclidean distance of two vectors, e.g Lemma 4. The derivation of Cao et al. [2] uses the property of quadratic forms of normally distributed random variables and the fact that the sum of normally distributed random variables is also distributed in Gaussian distribution. The calculation of the variance of squared L_2-norm without depending on the property of some distributions is not straightforward [38]. We divide the variance of squared Euclidean distance of two vectors into the variance of the norm of the feature vectors and the variance of the inner-product of vectors. Then we apply Cauchy–Schwarz inequality to the inner-product.

We start the proof from equation 15. We first prove the following Lemma 5 related to the expectation statistics of α in equation 15.

Lemma 5. *Under the same notations and assumptions as Theorem 1, then,*

$$\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha] = \frac{1}{K}\left(\mathrm{Tr}(\Sigma_{c_2}) - \mathrm{Tr}(\Sigma_{c_1})\right) + (\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})\top(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})$$
$$\mathbb{E}_{c_1,c_2\sim\tau}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}\mathbb{E}_{S\sim\mathcal{D}_{\otimes 2K}}[\alpha] = 2\mathrm{Tr}(\Sigma).$$

Proof. First, from the definition of α, we split $\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]$ in to two parts and examine them seperately.

$$\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha] = \underbrace{\mathbb{E}\left[\left\|\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right\|^2\right]}_{(i)} - \underbrace{\mathbb{E}\left[\left\|\phi(\boldsymbol{x}) - \overline{\phi(S_{c_1})}\right\|^2\right]}_{(ii)}. \tag{17}$$

In regular conditions, for random vector X, the expectation of the norm is

$$\mathbb{E}[X^\top X] = \mathrm{Tr}(\mathrm{Var}(X)) + \mathbb{E}[X]^\top\mathbb{E}[X], \tag{18}$$

and the variance of the vector is

$$\mathrm{Var}(X) = \mathbb{E}[XX^\top] - \mathbb{E}[X]E[X]^\top \tag{19}$$
$$\Sigma_{c_i} \triangleq \mathrm{Var}_{\boldsymbol{x}\sim\mathcal{D}_{c_i}}(\phi(\boldsymbol{x})). \tag{20}$$

Hence,

$$\begin{aligned}(i) &= \mathbb{E}_{\boldsymbol{x}\sim D_{c_1}}\mathbb{E}_S\left[\left\|\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right\|^2\right] \\ &= \mathrm{Tr}\left(\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S}\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right]\right) + \mathbb{E}_{\boldsymbol{x}}\mathbb{E}_S\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right]^\top\mathbb{E}_{\boldsymbol{x}}\mathbb{E}_S\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right],\end{aligned} \tag{21}$$

where the first term inside the trace can be expanded as:

$$\begin{aligned}\mathrm{Var}\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right] &= \mathbb{E}\left[\left(\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right)\left(\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right)^\top\right] - (\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})^\top \\ &= \mathrm{Var}\left[\phi(\boldsymbol{x})\right] + \mathbb{E}\left[\phi(\boldsymbol{x})\right]\mathbb{E}\left[\phi(\boldsymbol{x})\right]^\top + \mathrm{Var}\left[\overline{\phi(S_{c_2})}\right] + \mathbb{E}\left[\overline{\phi(S_{c_2})}\right]\mathbb{E}\left[\overline{\phi(S_{c_2})}\right]^\top \\ &\quad - \boldsymbol{\mu}_{c_2}\boldsymbol{\mu}_{c_1}^\top - \boldsymbol{\mu}_{c_1}\boldsymbol{\mu}_{c_2}^\top - (\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})^\top \\ &= \Sigma_{c_1} + \frac{1}{K}\Sigma_{c_2} \quad \text{(Last terms cancel out).}\end{aligned} \tag{22}$$

The second term in equation 21 is simply as follows.

$$\mathbb{E}_{\boldsymbol{x}\sim D_{c_1}}\mathbb{E}_S\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right] = \boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}. \tag{23}$$

From equation 22 and equation 23 we obtain

$$(i) = \mathrm{Tr}(\Sigma_{c_1}) + \frac{1}{K}\mathrm{Tr}(\Sigma_{c_2}) + (\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}). \tag{24}$$

Similarly for ii,

$$\begin{aligned}(ii) &= \mathrm{Tr}\left(\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S}\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_1})}\right]\right) + \mathbb{E}_{\boldsymbol{x}}\mathbb{E}_S\left[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_1})}\right]^\top\mathbb{E}_{\boldsymbol{x}}\mathbb{E}_S[\phi(\boldsymbol{x}) - \overline{\phi(S_{c_1})}] \\ &= \mathrm{Tr}(\Sigma_{c_1}) + \frac{1}{K}\mathrm{Tr}(\Sigma_{c_1}).\end{aligned} \tag{25}$$

From (i) and (ii), and equation 17

$$\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha] = \frac{1}{K}\left(\mathrm{Tr}\left(\Sigma_{c_2}\right) - \mathrm{Tr}\left(\Sigma_{c_1}\right)\right) + \left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)^\top\left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right). \tag{26}$$

Since $\mathbb{E}_{c_1,c_2,\boldsymbol{x},S}[\alpha] = \mathbb{E}_{c_1,c_2\sim\tau}[\mathbb{E}_{S\sim\mathcal{D}^{\otimes 2k}}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1}}[\alpha]]$,

$$\begin{aligned}
\mathbb{E}_{c_1,c_2,\boldsymbol{x},S}[\alpha] &= \mathbb{E}_{c_1,c_2\sim\tau}\left[\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2}) - \mathrm{Tr}(\Sigma_{c_1})) + \left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)^\top \left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)\right] \\
&= \mathbb{E}_{c_1,c_2\sim\tau}\left[\left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)^\top \left(\boldsymbol{\mu}_{c_1} - \boldsymbol{\mu}_{c_2}\right)\right] \\
&= \mathbb{E}_{c_1,c_2\sim\tau}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1} + \boldsymbol{\mu}_{c_2}^\top\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_2}^\top\boldsymbol{\mu}_{c_1}\right] \\
&= \mathrm{Tr}\left(\Sigma\right) + \boldsymbol{\mu}^\top\boldsymbol{\mu} + \mathrm{Tr}\left(\Sigma\right) + \boldsymbol{\mu}^\top\boldsymbol{\mu} - 2\boldsymbol{\mu}^\top\boldsymbol{\mu} \\
&= 2\mathrm{Tr}\left(\Sigma\right).
\end{aligned} \tag{27}$$

□

Next we prove the following Lemma 6 related to the conditioned variance of α.

Lemma 6. *Under the same notation and assumptions as Theorem 1,*

$$\mathbb{E}_{c_1,c_2}\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S\sim\mathcal{D}^{\otimes 2N}}[\alpha] \leq \frac{4}{K}\mathbb{E}_{c\sim\tau}\mathrm{Var}\left[\|\phi(\boldsymbol{x})\|^2\right] + \frac{4}{K}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}(\Sigma_c)\right] + \mathrm{V}_{\mathrm{wit}}(\Sigma_\tau, \Sigma, \boldsymbol{\mu}), \tag{28}$$

where

$$\begin{aligned}
\mathrm{V}_{\mathrm{wit}}\left(\Sigma_\tau, \Sigma, \boldsymbol{\mu}\right) = &\frac{8}{K}\left(\mathrm{Tr}(\Sigma_\tau)\right)\left(\mathrm{Tr}(\Sigma) + \boldsymbol{\mu}^\top\boldsymbol{\mu}\right) + 4\left(\mathrm{Tr}(\Sigma) + \boldsymbol{\mu}^\top\boldsymbol{\mu}\right)^2 \\
&+ 4\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2} - \boldsymbol{\mu}_{c_1}\right)\right].
\end{aligned} \tag{29}$$

Proof. We start with the inequality between the variance of 2 random variables. We define $\mathrm{Cov}(A, B)$ as covariance of 2 random variables A, B.

$$\begin{aligned}
\mathrm{Var}[A+B] &= \mathrm{Var}[A] + \mathrm{Var}[B] + 2\mathrm{Cov}(A, B) \\
&\leq \mathrm{Var}[A] + \mathrm{Var}[B] + 2\sqrt{\mathrm{Var}[A]\mathrm{Var}[B]} \\
&\leq \mathrm{Var}[A] + \mathrm{Var}[B] + 2\cdot\frac{\mathrm{Var}[A] + \mathrm{Var}[B]}{2} \\
&= 2\mathrm{Var}[A] + 2\mathrm{Var}[B].
\end{aligned} \tag{30}$$

For $\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S}[\alpha]$,

$$\begin{aligned}
\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S}[\alpha] &= \mathrm{Var}\left[\left\|\phi(\boldsymbol{x}) - \overline{\phi(S_{c_2})}\right\|^2 - \left\|\phi(\boldsymbol{x}) - \overline{\phi(S_{c_1})}\right\|^2\right] \\
&= \mathrm{Var}\left[\left\|\overline{\phi(S_{c_1})}\right\|^2 - \left\|\overline{\phi(S_{c_2})}\right\|^2 - 2\phi(\boldsymbol{x})^\top\left(\overline{\phi(S_{c_2})} - \overline{\phi(S_{c_1})}\right)\right] \\
&\leq 2\mathrm{Var}\left[\left\|\overline{\phi(S_{c_1})}\right\|^2 - \left\|\overline{\phi(S_{c_2})}\right\|^2\right] \\
&\qquad + 4\mathrm{Var}\left[\phi(\boldsymbol{x})^\top\left(\overline{\phi(S_{c_2})} - \overline{\phi(S_{c_1})}\right)\right] \quad (\because \textit{equation } 30) \\
&= 2\mathrm{Var}\left[\left\|\overline{\phi(S_{c_1})}\right\|^2\right] + 2\mathrm{Var}\left[\left\|\overline{\phi(S_{c_2})}\right\|^2\right] + 4\mathrm{Var}\left[\phi(\boldsymbol{x})^\top(\overline{\phi(S_{c_2})} - \overline{\phi(S_{c_1})})\right].
\end{aligned} \tag{31}$$

From 3rd line to 4th line we decompose the variance of $\left\|\overline{\phi(S_{c_1})}\right\|^2 - \left\|\overline{\phi(S_{c_2})}\right\|^2$ use the independence of $\overline{\phi(S_{c_1})}$ and $\overline{\phi(S_{c_2})}$ with their class given.

Next we focus on $\mathrm{Var}\left[\phi(\boldsymbol{x})^\top(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})})\right]$.

$$\begin{aligned}
\mathrm{Var}\left[\phi(\boldsymbol{x})^\top(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})})\right] &= \mathbb{E}\left[\left(\phi(\boldsymbol{x})^\top\left(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right)\right)^2\right] \\
&\quad - \left(\mathbb{E}\left[\phi(\boldsymbol{x})\right]^\top \mathbb{E}\left[\left(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right)\right]\right)^2 \\
&\leq \mathbb{E}\left[\left(\|\phi(\boldsymbol{x})\|^2\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right\|\right)^2\right] - (\boldsymbol{\mu}_{c_1}^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}))^2 \\
&= \mathbb{E}\left[\|\phi(\boldsymbol{x})\|^2\right]\mathbb{E}\left[\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})})\right\|^2\right] - \left(\boldsymbol{\mu}_{c_1}^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right)^2.
\end{aligned} \tag{32}$$

From the 2nd line to the 3rd line we use Cauchy–Schwarz inequality.

For $\mathbb{E}\left[\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right\|^2\right]$, with equation 18

$$\mathbb{E}\left[\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right\|^2\right] = \frac{1}{K}\left(\mathrm{Tr}\left(\Sigma_{c_1}\right)+\mathrm{Tr}\left(\Sigma_{c_2}\right)\right)+\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right). \tag{33}$$

Thus $\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S}\left[\|\phi(\boldsymbol{x})\|^2\right]\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S}\left[\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right\|^2\right]\right]$ is calculated as follows.

$$\begin{aligned}
&\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S}\left[\|\phi(\boldsymbol{x})\|^2\right]\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S}\left[\left\|\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})}\right\|^2\right]\right] \\
&= \mathbb{E}_{c_1,c_2\sim\tau}\left[(\mathrm{Tr}(\Sigma_{c_1})+\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1})\left(\frac{1}{K}\left(\mathrm{Tr}(\Sigma_{c_1})+\mathrm{Tr}(\Sigma_{c_2})\right)+(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})\right)\right] \\
&= \mathbb{E}_{c_1,c_2\sim\tau}\left[\frac{1}{K}\left(\mathrm{Tr}\left(\Sigma_{c_1}\right)^2+\mathrm{Tr}(\Sigma_{c_1})\mathrm{Tr}\left(\Sigma_{c_2}\right)\right)\right] \\
&\quad + \mathbb{E}_{c_1,c_2\sim\tau}\left[\frac{2}{K}\left(\mathrm{Tr}\left(\Sigma_\tau\right)\right)\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}+\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&\quad + \mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&= \frac{1}{K}\left(\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}(\Sigma_{c_1})^2\right]+\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}(\Sigma_{c_1})\right]^2\right) \\
&\quad + \frac{2}{K}\left(\mathrm{Tr}\left(\Sigma_\tau\right)\right)\left(\mathrm{Tr}\left(\Sigma\right)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+\mathbb{E}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})\right] \\
&\quad + \mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&= \frac{1}{K}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}\left(\Sigma_c\right)\right] \\
&\quad + \frac{2}{K}\mathrm{Tr}\left(\Sigma_\tau\right)^2+\frac{2}{K}\left(\mathrm{Tr}\left(\Sigma_\tau\right)\right)\left(\mathrm{Tr}\left(\Sigma\right)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+\mathbb{E}_{c_1,c_2}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&\quad + \mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&= \frac{1}{K}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}(\Sigma_c)\right] \\
&\quad + \frac{2}{K}\mathrm{Tr}(\Sigma_\tau)\left(\mathrm{Tr}(\Sigma_\tau)+\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+\mathbb{E}_{c_1,c_2}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right] \\
&\quad + \mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right].
\end{aligned} \tag{34}$$

Now we take into account the term $-(\boldsymbol{\mu}_{c_1}^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}))^2$ in equation 32,

$$\begin{aligned}
&\mathbb{E}_{c_1,c_2}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1})-(\boldsymbol{\mu}_{c_1}^\top(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}))^2\right]\\
&\quad=\mathbb{E}_{c_1,c_2}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}\boldsymbol{\mu}_{c_2}^\top\boldsymbol{\mu}_{c_2}-(\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_2})^2\right]\\
&\quad\le\mathbb{E}_{c_1,c_2}\left[\boldsymbol{\mu}_{c_1}^\top\boldsymbol{\mu}_{c_1}\boldsymbol{\mu}_{c_2}^\top\boldsymbol{\mu}_{c_2}\right]\\
&\quad=\left(\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)^2.
\end{aligned}\tag{35}$$

Thus $\mathbb{E}_{c_1,c_2}\left[\mathrm{Var}[\phi(\boldsymbol{x})^\top(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})})]\right]$ is calculated as follows.

$$\begin{aligned}
&\mathbb{E}_{c_1,c_2}\left[\mathrm{Var}[\phi(\boldsymbol{x})^\top(\overline{\phi(S_{c_2})}-\overline{\phi(S_{c_1})})]\right]\\
&=\frac{1}{K}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}(\Sigma_c)\right]+\frac{2}{K}\mathrm{Tr}(\Sigma_\tau)\left(\mathrm{Tr}(\Sigma_\tau)+\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+(\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu})^2\\
&\quad+\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right].
\end{aligned}\tag{36}$$

Regarding $\left\|\overline{\phi(S_c)}\right\|^2$, since the function computing square norm is convex, next equation holds with D-dimensional Jensen's inequlality [39]:

$$\begin{aligned}
\left\|\overline{\phi(S_c)}\right\|^2&=\left\|\frac{1}{K}\sum_{\substack{i=0\\x\in S_c}}\phi(\boldsymbol{x})\right\|^2\\
&\le\frac{1}{K}\left\|\sum_{\substack{i=0\\x\in S_c}}\phi(\boldsymbol{x})\right\|^2.
\end{aligned}\tag{37}$$

Combining equation 31, equation 36, and equation 37 we obtain

$$\begin{aligned}
&\mathbb{E}_{c_1,c_2}\mathrm{Var}_{\boldsymbol{x}\sim D_{c_1},S\sim\mathcal{D}^{\otimes 2N}}[\alpha]\\
&\le\frac{4}{K}\mathbb{E}_{c\sim\tau}\mathrm{Var}_{\boldsymbol{x}\sim\mathcal{D}_c}\left[\|\phi(\boldsymbol{x})\|^2\right]+\frac{4}{K}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}(\Sigma_c)\right]\\
&\quad+\frac{8}{K}\mathrm{Tr}(\Sigma_\tau)\left(\mathrm{Tr}(\Sigma_\tau)+\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+4\left(\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)^2\\
&\quad+4\mathbb{E}_{c_1,c_2\sim\tau}\left[\mathrm{Tr}\left(\Sigma_{c_1}\right)\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)^\top\left(\boldsymbol{\mu}_{c_2}-\boldsymbol{\mu}_{c_1}\right)\right].
\end{aligned}\tag{38}$$

□

To complete the proof of Theorem 1, we calculate $\mathbb{E}_{c_1,c_2}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S\sim\mathcal{D}^{\otimes 2K}}[\alpha]^2$.

$$\begin{aligned}
&\mathbb{E}_{c_1,c_2}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{c_1},S\sim\mathcal{D}^{\otimes 2K}}[\alpha]^2\\
&=\mathbb{E}_{c_1,c_2}\left[\left(\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2})-\mathrm{Tr}(\Sigma_{c_1}))+(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\right)^2\right]\\
&=\mathbb{E}_{c_1,c_2}\left[\left(\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2})-\mathrm{Tr}(\Sigma_{c_1}))\right)^2\right]+\mathbb{E}_{c_1,c_2}\left[\left((\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\right)^2\right]\\
&\quad+2\mathbb{E}_{c_1,c_2}\left[\left(\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2})-\mathrm{Tr}(\Sigma_{c_1}))\right)\left((\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\right)\right]\\
&=\mathrm{Var}_{c_1,c_2\sim\tau}\left[\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2})-\mathrm{Tr}(\Sigma_{c_1}))\right]+\left(\mathbb{E}_{c_1,c_2\sim\tau}\left[\frac{1}{K}(\mathrm{Tr}(\Sigma_{c_2})-\mathrm{Tr}(\Sigma_{c_1}))\right]\right)^2\\
&\quad+\mathbb{E}_{c_1,c_2}\left[\left((\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\right)^2\right]\\
&=\frac{2}{K^2}\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}(\Sigma_c)\right]+\mathbb{E}_{c_1,c_2}\left[\left((\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})^\top(\boldsymbol{\mu}_{c_1}-\boldsymbol{\mu}_{c_2})\right)^2\right].
\end{aligned}\tag{39}$$

From 2nd line to 3rd line, we use the symmetry of the last term with respect to c_1 and c_2 and erase the term.

Combining equation 15, Lemma 5, Lemma 6, and equation 39, we obtain the bound.

A.5 Theorem 1 with *N*-way Classification

The upper bound on the risk of N-way prototype classifier is as follows.

Theorem 7. *Let operation of binary class prototype classifier $\mathcal{M}$ as defined in equation 1. Then for $\overline{\phi(S_c)} = \frac{1}{K}\Sigma_{\boldsymbol{x}\in S_c}\phi(\boldsymbol{x})$, $\mu_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[\phi(\boldsymbol{x})]$, $\Sigma_c = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_c}[(\phi(\boldsymbol{x})-\mu_c)(\phi(\boldsymbol{x})-\mu_c)^\top]$, $\mu = \mathbb{E}_{c\sim\tau}[\mu_c]$,$\Sigma = \mathbb{E}_{c\sim\tau}[(\mu_c-\mu)(\mu_c-\mu)^\top]$, $\Sigma_\tau = \mathbb{E}_{c\sim\tau}[\Sigma_c]$, if $\phi(\boldsymbol{x})$ has its fourth moment, miss classification risk of binary class prototype classifier $\mathrm{R}_{\mathcal{M}}$ satisfy*

$$
\begin{aligned}
&\mathrm{R}(\mathcal{M}(\phi,\boldsymbol{x},\{S_i\}_{i=1}^N),y)\\
&\leq N-1-\sum_{\substack{c=1\\c\neq y}}^{N}\frac{4(\mathrm{Tr}(\Sigma))^2}{\mathbb{E}\mathrm{V}[h_{L2}(\phi(\boldsymbol{x}))]+\mathrm{V}_{\mathrm{Tr}}(\Sigma_y)+\mathrm{V}_{\mathrm{wit}}(\Sigma_\tau,\Sigma,\boldsymbol{\mu})+\mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_y,\boldsymbol{\mu}_c)},
\end{aligned}
\tag{40}
$$

where

$$
\begin{aligned}
\mathbb{E}\mathrm{V}[h_{L2}(\phi(\boldsymbol{x}))] =& \frac{4}{K}\mathbb{E}_{y\sim\tau}\left[\mathrm{Var}_{\boldsymbol{x}_c\sim\mathcal{D}_c}\left[\|\phi(\boldsymbol{x})\|^2\right]\right],\\
\mathrm{V}_{\mathrm{Tr}}(\Sigma_y) =& \left(\frac{4}{K}+\frac{2}{K^2}\right)\mathrm{Var}_{c\sim\tau}\left[\mathrm{Tr}\left(\Sigma_c\right)\right],\\
\mathrm{V}_{\mathrm{wit}}\left(\Sigma_\tau,\Sigma,\boldsymbol{\mu}\right) =& \frac{8}{K}\left(\mathrm{Tr}(\Sigma_\tau)\right)\left(\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)+4\left(\mathrm{Tr}(\Sigma)+\boldsymbol{\mu}^\top\boldsymbol{\mu}\right)^2\\
&+4\mathbb{E}_{c\sim\tau}\left[\mathrm{Tr}\left(\Sigma_y\right)\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_c\right)^\top\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_c\right)\right],\\
\mathbb{E}\,\mathrm{dist}^2_{\mathrm{L2}}(\boldsymbol{\mu}_y,\boldsymbol{\mu}_c) =& \mathbb{E}_{y,c}\left[\left(\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_c\right)^\top\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_c\right)\right)^2\right].
\end{aligned}
\tag{41}
$$

Proof. Let x, y be the input and its class of a query data. We define α_c by $\alpha_c = \left\|\phi(\boldsymbol{x})-\overline{\phi(S_c)}\right\|^2 - \left\|\phi(\boldsymbol{x})-\overline{\phi(S_y)}\right\|^2$. Then a prototype classifier miss-classify a class of input x, $\hat{y}$, if $\exists c\in[1,N], c\neq y, \alpha_c<0$. Hence: $\mathrm{R}_{\mathcal{M}}(\phi) = \Pr(\bigcup_{\substack{c=1\\c\neq y}}^{N}\alpha_c<0)$

By Frechet's inequality, next equation holds:

$$
\mathrm{R}_{\mathcal{M}}(\phi)\leq\sum_{\substack{c=1\\c\neq y}}^{N}\Pr(\alpha_i<0).
$$

Noting that Theorem 1 can be applied to each term in the summation and then we obtain Theorem 7 □

A.6 Detailed performance results

We show in Table 3 the detailed performance results of standard object recognition in this section with 95% confidence margin. The table shows the similar result with Table 1. We can observe that the prototype classifier with L_2-norm, EST+L_2-norm, LDA+L_2-norm performs comparably with ProtoNet and the linear-evaluation-based approach in most of the settings.

Table 3: Classification accuracies with ResNet18 on *mini*ImageNet, *tiered*ImageNet, CIFARFS, and FC100 of ProtoNet, linear-evaluation-based methods [3], centering with L_2-norm [24], and ours. The Baseline without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline are in bold. Regarding to Baseline++, Baseline++ without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline++ are in bold.

	*mini*ImageNet		*tiered*ImageNet		CIFAR-FS		FC100	
	1-shot	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
methods with meta-learning or linear-evaluation								
ProtoNet[1]	56.74 ± 0.84	75.64 ± 0.62	61.75 ± 0.94	81.56 ± 0.68	65.46 ± 0.96	79.52 ± 0.66	35.92 ± 0.71	50.86 ± 0.71
Baseline [3]	55.41 ± 0.82	76.95 ± 0.61	63.38 ± 0.91	83.18 ± 0.64	65.12 ± 0.88	79.68 ± 0.64	40.06 ± 0.68	57.04 ± 0.71
A prototype classifier with feature-transformation methods								
Baseline-w/o-linear , centering+L_2-norm[24]	**57.67 ± 0.83**	75.50 ± 0.67	**65.26 ± 0.88**	81.63 ± 0.64	**65.26 ± 0.86**	78.58 ± 0.66	**41.51 ± 0.72**	**56.44 ± 0.74**
Baseline-w/o-linear	43.86 ± 0.80	72.36 ± 0.92	56.16 ± 0.89	80.33 ± 0.66	54.06 ± 0.85	77.76 ± 0.64	35.90 ± 0.61	55.20 ± 0.77
Baseline-w/o-linear , L_2-norm	**56.57 ± 0.80**	**76.44 ± 0.61**	**65.19 ± 0.87**	**82.93 ± 0.66**	**64.76 ± 0.87**	**79.90 ± 0.66**	**40.96 ± 0.71**	**57.71 ± 0.76**
Baseline-w/o-linear , EST	44.19 ± 0.82	70.85 ± 0.92	57.05 ± 0.93	73.16 ± 0.78	57.16 ± 0.88	78.98 ± 0.69	**48.43 ± 0.91**	**63.75 ± 0.73**
Baseline-w/o-linear , EST+L_2-norm	**56.39 ± 0.79**	**76.24 ± 0.64**	**64.71 ± 0.91**	**83.24 ± 0.67**	**65.54 ± 0.71**	**79.80 ± 0.71**	**50.50 ± 0.97**	**65.31 ± 0.77**
Baseline-w/o-linear , LDA	47.14 ± 0.81	69.87 ± 0.65	57.05 ± 0.90	81.19 ± 0.65	56.49 ± 0.86	78.88 ± 0.66	**50.29 ± 0.85**	**63.56 ± 0.77**
Baseline-w/o-linear , LDA+L_2-norm	**56.37 ± 0.81**	**76.39 ± 0.66**	**65.44 ± 0.94**	**83.16 ± 0.62**	**65.64 ± 0.84**	**80.72 ± 0.67**	**50.40 ± 0.97**	**65.31 ± 0.77**
methods with meta-learning or linear-evaluation								
Baseline++ [3]	55.07 ± 0.81	74.71 ± 0.61	64.02 ± 0.92	83.18 ± 0.64	65.64 ± 0.93	79.80 ± 0.66	36.93 ± 0.70	50.41 ± 0.73
A prototype classifier with feature-transformation methods								
Baseline++-w/o-linear , centering+L_2-norm[24]	**57.00 ± 0.64**	74.06 ± 0.61	**64.92 ± 0.91**	81.41 ± 0.66	**66.14 ± 0.93**	**80.03 ± 0.71**	**38.30 ± 0.74**	**51.06 ± 0.70**
Baseline++-w/o-linear	36.80 ± 0.76	63.76 ± 0.71	48.27 ± 0.91	75.87 ± 0.71	50.18 ± 0.96	74.23 ± 0.70	30.76 ± 0.62	47.62 ± 0.69
Baseline++-w/o-linear , L_2-norm	**57.21 ± 0.83**	**74.89 ± 0.65**	**66.67 ± 0.94**	**82.49 ± 0.68**	**66.84 ± 0.92**	**80.49 ± 0.71**	**38.55 ± 0.72**	**51.15 ± 0.71**
Baseline++-w/o-linear , EST	47.21 ± 0.77	69.11 ± 0.64	53.49 ± 0.90	71.81 ± 0.75	53.36 ± 0.93	73.70 ± 0.74	**39.92 ± 0.88**	**53.61 ± 0.70**
Baseline++-w/o-linear , EST+L_2-norm	**57.00 ± 0.75**	**76.13 ± 0.64**	**65.52 ± 0.97**	**79.96 ± 0.74**	**66.57 ± 0.95**	**79.42 ± 0.70**	**42.89 ± 0.82**	**56.82 ± 0.70**
Baseline++-w/o-linear , LDA	45.95 ± 0.78	68.34 ± 0.68	49.09 ± 0.90	76.17 ± 0.68	52.98 ± 0.91	73.36 ± 0.75	**40.27 ± 0.79**	**55.80 ± 0.74**
Baseline++-w/o-linear , LDA+L_2-norm	**56.78 ± 0.87**	**75.83 ± 0.65**	**66.68 ± 0.90**	**82.53 ± 0.67**	**65.89 ± 0.93**	**79.47 ± 0.70**	**44.21 ± 0.86**	**58.10 ± 0.70**

A.7 Performance results of cross dataset

We show in Table 4 the performance results of cross-dataset scenario in this section with 95% confidence margin. From the table, we can observe that the prototype classifier with L_2-norm, EST+L_2-norm and LDA+L_2-norm performs comparably with linear-evaluation-based methods. Therefore, our analysis results still hold on the cross-domain scenario.

We also found that ProtoNet performs worse than linear-evaluation-based methods and a prototype classifier with feature-transformation methods. In this scenario, we should focus on improving the performance of a prototype classifier with features extracted from a model trained without meta-learning, e.g, Baseline and Baseline++. Since there is no guarantee that the features of a model trained with cross-entropy loss will be distributed in Gaussian distribution, Cao et al. [2]'s bound cannot explain the performance improvement of the prototype classifier while our analysis can.

Table 4: Classification accuracies with ResNet12 and ResNet18 on cross-domain scenario (*mini*ImageNet $\rightarrow$ CUB) of ProtoNet, linear-evaluation-based methods [3], centering with L_2-norm [24], and ours. The Baseline without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline are in bold. Regarding to Baseline++, Baseline++ without linear-evaluation methods with accuracy greater than the lower 95% confidence margin of the accuracy of ProtoNet and Baseline++ are in bold.

	*mini*ImageNet $\rightarrow$ CUB ResNet12		*mini*ImageNet $\rightarrow$ CUB ResNet18	
	1-shot	**5-shot**	**1-shot**	**5-shot**
methods with meta-learning or linear-evaluation				
ProtoNet[1]	38.47 ± 0.69	61.47 ± 0.68	44.01 ± 0.79	61.79 ± 0.75
Baseline [3]	46.91 ± 0.81	66.55 ± 0.72	46.36 ± 0.78	66.97 ± 0.73
A prototype classifier with feature-transformation methods				
Baseline-w/o-linear , centering+L_2-norm[24]	**46.70 $\pm$ 0.79**	65.36 ± 0.70	**46.56 $\pm$ 0.77**	65.96 ± 0.72
Baseline-w/o-linear	43.05 ± 0.72	63.47 ± 0.67	42.11 ± 0.74	64.31 ± 0.73
Baseline-w/o-linear , L_2-norm	**47.64 $\pm$ 0.79**	**66.31 $\pm$ 0.73**	**46.55 $\pm$ 0.83**	**67.21 $\pm$ 0.73**
Baseline-w/o-linear , EST	43.27 ± 0.77	61.59 ± 0.70	43.90 ± 0.77	63.50 ± 0.72
Baseline-w/o-linear , EST+L_2-norm	**46.33 $\pm$ 0.78**	**66.28 $\pm$ 0.71**	**47.10 $\pm$ 0.83**	**66.33 $\pm$ 0.74**
Baseline-w/o-linear , LDA	43.22 ± 0.74	62.47 ± 0.71	44.02 ± 0.74	65.34 ± 0.75
Baseline-w/o-linear , LDA+L_2-norm	**47.65 $\pm$ 0.80**	**66.94 $\pm$ 0.72**	**48.47 $\pm$ 0.82**	**67.21 $\pm$ 0.69**
methods with meta-learning or linear-evaluation				
Baseline++ [3]	45.56 ± 0.82	66.00 ± 0.74	46.26 ± 0.87	63.53 ± 0.71
A prototype classifier with feature-transformation methods				
Baseline++-w/o-linear , centering+L_2-norm[24]	**47.36 $\pm$ 0.82**	64.44 ± 0.75	**46.55 $\pm$ 0.79**	**64.52 $\pm$ 0.71**
Baseline++-w/o-linear	40.68 ± 0.71	61.02 ± 0.71	39.72 ± 0.70	60.00 ± 0.74
Baseline++-w/o-linear , L_2-norm	**47.03 $\pm$ 0.83**	**65.86 $\pm$ 0.75**	**46.87 $\pm$ 0.82**	**65.39 $\pm$ 0.71**
Baseline++-w/o-linear , EST	40.28 ± 0.73	56.79 ± 0.72	40.11 ± 0.74	54.95 ± 0.73
Baseline++-w/o-linear , EST+L_2-norm	**46.64 $\pm$ 0.84**	**65.39 $\pm$ 0.74**	**45.96 $\pm$ 0.78**	**64.23 $\pm$ 0.77**
Baseline++-w/o-linear , LDA	40.44 ± 0.75	57.55 ± 0.69	39.54 ± 0.72	56.51 ± 0.78
Baseline++-w/o-linear , LDA+L_2-norm	**46.32 $\pm$ 0.81**	**65.35 $\pm$ 0.76**	**45.60 $\pm$ 0.81**	**64.17 $\pm$ 0.73**