# Kernel Relative-prototype Spectral Filtering for Few-shot Learning

Tao Zhang[1][0000−0002−0281−9234] and Wu Huang[2][0000−0002−2525−6454]

[1] Chengdu Techman Software Co., Ltd., Chengdu, Sichuan, China
ztuestc@outlook.com
[2] Sichuan University, Chengdu, Sichuan, China
huangwu@scu.edu.cn

**Abstract.** Few-shot learning performs classification tasks and regression tasks on scarce samples. As one of the most representative few-shot learning models, Prototypical Network represents each class as sample average, or a prototype, and measures the similarity of samples and prototypes by Euclidean distance. In this paper, we propose a framework of spectral filtering (shrinkage) for measuring the difference between query samples and prototypes, or namely the relative prototypes, in a reproducing kernel Hilbert space (RKHS). In this framework, we further propose a method utilizing Tikhonov regularization as the filter function for few-shot classification. We conduct several experiments to verify our method utilizing different kernels based on the *mini*ImageNet dataset, *tiered*-ImageNet dataset and CIFAR-FS dataset. The experimental results show that the proposed model can perform the state-of-the-art. In addition, the experimental results show that the proposed shrinkage method can boost the performance. Source code is available at https://github.com/zhangtao2022/DSFN.

**Keywords:** Few-shot learning, Relative-Prototype, Spectral Filtering, Shrinkage, Kernel

## 1 Introduction

Humans have an innate ability to quickly learn from one or several labeled pictures and infer the category of new pictures. In contrast, deep learning, despite its breakthrough success in computer vision, still needs huge data to drive it. This shortcoming seriously hinders its applications in some practical situations where data is scanty. Therefore, it is an important and challenging problem for machines to acquire the human-like ability to make inferences about unknown samples based on too few samples. Inspired by this ability of humans, few-shot learning is proposed and has become a hot spot [9,43,19]. Vinyals et al. [44] proposed a training paradigm that few-shot learning models should learn new categories of unseen examples from query set using very few examples from support set. Meta-learning methods can be well applied to the few-shot settings that need to complete the task that contains both support set and query set.

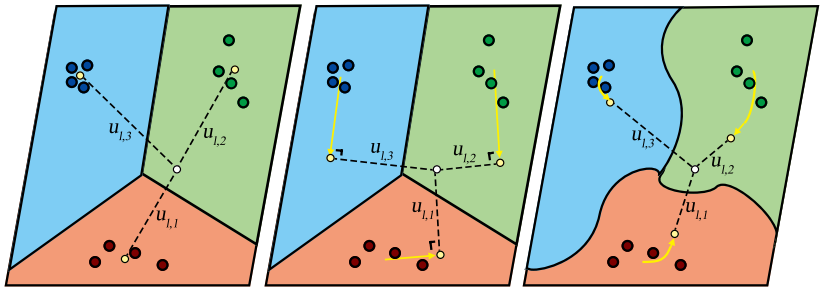


Fig. 1: Comparison of the relative-prototypes (dotted line) in the Prototypical Networks (ProtoNet)[41], Deep Subspace Networks (DSN)[40] and the proposed DSFN. **Left**: ProtoNet in Euclidean space; **Middle**: DSN in Euclidean space; **Right**: DSFN in a reproducing kernel Hilbert space. For them, $u_{l,1}$,$u_{l,2}$ and $u_{l,3}$ are the relative-prototypes with class 1(brown), class 2(green) and class 3(blue), respectively.

Recently, a series of meta-learning models for few-shot setting have been proposed, which can be divided into the metric-based models and the optimization-based models [41,10,42,29,20,24,2,35,17,47,13,23,37,48]. Prototypical Networks (ProtoNet), as one of the metric-based representatives, proposes to use prototypes to represent each category, and to measure the similarity between a sample and a prototype by using Euclidean distance [41]. Based on this idea, some prototype-related models have been proposed in recent years [5,32,11,33].

The prototype in ProtoNet is estimated by support sample mean in each class, which may deviate from the true prototype [22]. In [22], Two bias elimination mechanisms are proposed to eliminate the difference between the prototype estimated value and the true value. In addition, a method of using a combination of semantic information and visual information to better estimate the prototype was proposed, and a significant improvement was obtained [45].

In measurement of sample similarity, Mahalanobis distance would be better than Euclidean distance to capture the information from the distribution within the class[3]. In addition, similarity measurement in hyperbolic space that learns the features of a hierarchical structure could be better than those in Euclidean space for few-shot image classification [8,16].

In this paper, we propose a method called Deep Spectral Filtering Networks (DSFN) aiming to better estimate relative prototypes, or the difference between prototypes and query samples (Fig. 1). Euclidean distance can not fully capture the information of intra-class difference as it is applied to measure the difference between sample and prototype. Thus, the main components of the inner class distance are taken into account in the similarity assessment between the sample and the prototype, which to some extent interferes with this assessment. In our approach, the influence of these components is weakened via spectral filtering. It is similar to the kernel mean shrinkage estimation that aims to reduce the expected risk of mean estimation [26,25,28,27], but the estimated relative prototypes don't have to be close to the true mean.

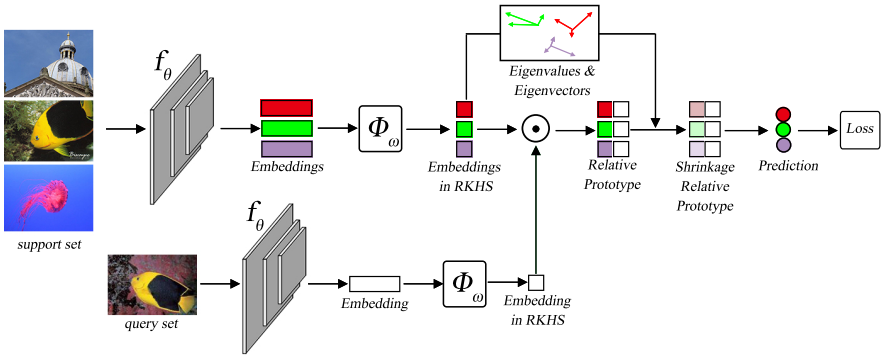


Fig. 2: The overview of our proposed approach. The features of support set and query set extracted from $f_\theta$ are mapped into RKHS by the function $\phi_\omega$. The relative prototypes are shrunk based on the eigenvalues and eigenvectors from the support set.

Recent work has shown that kernel embedding can significantly improve the performance of few-shot learning [8]. Based on kernel mean embedding theory, our approach can measure the relative prototypes in a reproducing kernel Hilbert space (RKHS). The advantage of this is that we can deal with the projection problem of feature space by means of the kernel embedding. The overview of our proposed approach is shown in Fig. 2.

The contributions of our work are summarized as four folds:

1) To our knowledge, this work is the first to estimate relative-prototypes using a kernel shrinkage method for few-shot learning.
2) We propose to estimate relative prototypes instead of prototypes, aiming to better capture the probability information that the query samples belongs to a class.
3) We propose a framework of spectral filtering to estimate the relative prototypes. This approach can filter the interference within cluster variation while measuring the relative prototypes and boost the performance.
4) We introduce kernel embedding for measuring the relative prototypes via spectral filtering in RKHS, which allows us to apply the kernels that achieve the state-of-the-art performance.

## 2 Related Work

### 2.1 Metric-based Few-shot Learning

Recently, some metric-based meta-learning algorithms for few-shot setting, one-shot setting and zero-shot setting were proposed, and the representatives of metric-based meta learning are Matching Networks (MatchingNet) [44] and ProtoNet [41]. MatchingNet proposed an attention mechanism for rapid learning. In addition, it pointed out that in the training procedure, the test condition should

match with the train condition. ProtoNet proposed a simple strategy to improve the performance of MatchingNet. The strategy is using prototype, the mean of support samples in each class, to represent its support set[41]. From the perspective of classifier design, some works proposed a different kind of classifier with MatchingNet and ProtoNet. Simon et al. proposed the dynamic classifiers by using subspaces for few-shot learning called Deep Subspace Networks (DSN)[40]. These classifiers are defined as the projection of the difference between a prototype and a query sample onto a subspace. Lee et al. [20] proposed a linear support vector machine instead of nearest neighbor classifiers for few-shot learning.

Our work is related to the rectification of the prototype. For ProtoNet, the prototype is simply calculated by averaging the support sample values. Some kinds of work show that, for improving the classification performance, the prototype need to be better estimated. For example, compared with the simple visual information, the combination of cross-modal information and visual information can better represent the prototype [45,33]. In addition, a method of transductive setting has been proposed to rectify prototype, which diminishes the intra-class bias via label propagation and diminishes the cross-class bias via feature shifting [22]. Gao et al. used a combination of the instance-level attention and the feature-level attention for the noisy few-shot relation classification [12]. In this method, prototype is a weighted mean of support samples after conducting by the instance-level attention.

Our work is also related to the similarity measurement of two samples. Similarity measurement between samples in ProtoNet is carried out by using Euclidean distance. Euclidean distance implies two assumptions, that is, the characteristic dimensions are not correlated and have consistent covariance. However, in the real world these two assumptions are not necessarily true. Boris N. Oreshkin et al. found that, by using the metric scaling, the performance of few-shot algorithms can be improved by optimizing the metric scaling, and showed that the performance using scaled cosine distance nears that using Euclidean distance[32]. The metric scaling can also be learned from a Bayesian method perspective.[5]. Sung et al. proposed the Relation Networks (Relation Nets) that learns to learn a transferrable deep metric[42]. In addition, Mahalanobis distance is proposed to overcome the defect of Euclidean distance on measuring the similarity between two samples [3,11]. Recently, Khrulkov et al. studied how to measure sample similarity in a hyperbolic space instead of Euclidean space. They found that, compared with Euclidean space, hyperbolic embeddings can benefit the embedding of images and provide a better performance [16]. However, it is more difficult to operate in hyperbolic space, such as sample averaging, which further hinders the use of hyperbolic geometry. Fang et al. [8] provides several positive definite kernel functions of hyperbolic spaces, which enable one to operate in hyperbolic spaces.

### 2.2 Kernel Mean Shrinkage Estimation

The method of kernel mean shrinkage estimation is relevant to our work, which is employed to estimate the relative prototype. In recent years, some work has

been done on the kernel mean shrinkage estimation. Muandet et al. [25] pointed out that the estimation of an empirical average in RKHS can be improved by employing Stein effect. Like James-Stein estimator, they propose several kernel mean shrinkage estimators, e. g., empirical-bound kernel mean shrinkage estimator (B-KMSE) and spectral kernel mean shrinkage estimator (S-KMSE) [27]. B-KMSE is easily optimized but the degree of shrinkage is the same for all coordinates. S-KMSE can shrink differently on each coordinate on the basis of eigenspectrum of covariance matrix, but it is difficult to optimize. Furthermore, "Shrinkage" can be achieved by a generalized filter function that can be embodied by various forms, such as Truncated SVD [28]. For these estimators, the shrinkage parameters are commonly obtained by a cross-validation score.

## 3 Methodology

### 3.1 Preliminary

In metric-based few-shot learning, before measuring the difference between query and classes, one often firstly represent each class by using support samples belonging to them, respectively. Here we consider two different geometric types of the class representation.

**Point type.** In this type, a class is often represented as a point, e.g., a prototype, which is calculated by averaging of support samples in each class [41]. In addition, the point can also be calculated by summing over the support samples[42]. In these cases, The probability of measuring a sample belonging to a class is usually based on the distance between two points. For example, for ProtoNets, the distance square is expressed as

$$d_{l,c}^2 = \|f_\theta\left(\boldsymbol{q}_l\right) - \boldsymbol{\mu}_c\|^2, \text{ with } \boldsymbol{\mu}_c = \frac{1}{n}\sum_{i=1}^{n} f_\theta\left(\mathbf{s}_{i,c}\right), \tag{1}$$

where $\boldsymbol{q}_l$ is the $l$th sample in query set, and $\boldsymbol{s}_{i,c}$ is the $i$th sample in the $c$th class of the support set. In addition, $f_\theta(\cdot)$ with the parameter $\theta$ is a network.

**Subspace type.** In addition to the point type, a class can also be represented as a subspace, e. g., the subspace spanned by the the feature vectors created from the support samples in a class [40]. In this case, a sample should be close to its own class subspace and stay away from the subspaces of other classes. For example, for DSN, the distance square is expressed as

$$d_{l,c}^2 = \|(\boldsymbol{I} - \boldsymbol{P}_c\boldsymbol{P}_c^T)(f_\theta\left(\boldsymbol{q}_l\right) - \boldsymbol{\mu}_c)\|^2, \text{ with } \boldsymbol{\mu}_c = \frac{1}{n}\sum_{i=1}^{n} f_\theta\left(\mathbf{s}_{i,c}\right), \tag{2}$$

where $P_c$ the truncated matrix of $W_c$ the eigenvector matrix of empirical covariance matrix of support set. For a SVM classifier in few-shot learning, the representation of each class is one or more subspaces whose boundaries are so-called hyperplanes, which are determined by the support vectors [20]. In this

case, a sample should belong to its own class subspace and stay away from the hyperplanes.

In the following, we show that the class representations of the two types are not entirely different. For example, we demonstrate that our framework can be embodied as either of ProtoNet and DSN with different filter functions (see the Section 3.4).

### 3.2 Kernel Shrinkage Relative-prototype

Here we propose the definition of kernel shrinkage relative-prototype. Given the $C$-way $n$-shot support set $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ..., \boldsymbol{S}_C\}$ with $\boldsymbol{S}_c = \{\boldsymbol{s}_{1,c}, \boldsymbol{s}_{2,c}, ..., \boldsymbol{s}_{n,c}\}$, and the query set $\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_m\}$. We firstly use function $\phi_\omega$ to map the observations of support samples of class $c$ in feature space, and calculate the prototype as:

$$\boldsymbol{\mu}_c = \frac{1}{n} \sum_{i=1}^{n} \phi_\omega \left( f_\theta \left( \mathbf{s}_{i,c} \right) \right), \tag{3}$$

where $\phi_\omega(\cdot)$ with the parameter $\omega$ is a mapping function, and $f_\theta(\cdot)$ with the parameter $\theta$ is a network. Similar to the concepts in [46,15], we propose the relative-prototype, the difference between a sample in the query set and a prototype of class as a mean form:

$$\boldsymbol{\mu}_{l,c} = \phi_\omega \left( f_\theta \left( \boldsymbol{q}_l \right) \right) - \boldsymbol{\mu}_c = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{m}_{l,i,c}, \tag{4}$$

where

$$\boldsymbol{m}_{l,i,c} = \phi_\omega \left( f_\theta \left( \boldsymbol{q}_l \right) \right) - \phi_\omega \left( f_\theta \left( \boldsymbol{s}_{i,c} \right) \right). \tag{5}$$

Simon et al. proposed a method that using adaptive subspaces instead of prototypes for few-shot learning [40]. In their work, the sample similarity is measured via a distance between a query sample and a subspace created from a support set, which can be seen as a "shrinkage" of the distance between a query sample and a prototype. Enlightened by it, we apply the shrinkage estimation theory to extend their idea and measure the sample similarity in a RKHS, or express the kernel shrinkage estimation of $\boldsymbol{\mu}_{l,c}$ as

$$\boldsymbol{\mu}_{l,c}(\lambda_c) = \boldsymbol{\mu}_{l,c} - \sum_{i=1}^{n} h(\gamma_{i,c}, \lambda_{l,c}) \gamma_{i,c} \langle \boldsymbol{\mu}_{l,c}, \boldsymbol{w}_{i,c} \rangle \boldsymbol{w}_{i,c}, \tag{6}$$

where $\lambda_{l,c}$ is the shrinkage coefficient with the class $c$, $(\gamma_{i,c}, \boldsymbol{w}_{i,c})$ are respectively the eigenvalues and eigenvectors of the empirical covariance matrix $C = \sum_{i=1}^{n} \boldsymbol{r}_{i,c} \otimes \boldsymbol{r}_{i,c}$ where

$$\boldsymbol{r}_{i,c} = \phi_{\omega}\left(f_{\theta}\left(\boldsymbol{s}_{i,c}\right)\right) - \boldsymbol{\mu}_{c}. \tag{7}$$

In Eq. 6, $h(\gamma_{i,c}, \lambda_{l,c})$ is a shrinkage function that approaches $1/\gamma_{i,c}$ as $\lambda_{l,c}$ decreases to 0, implying that the relative prototype is fully shrunk; As $\lambda_{l,c}$ increases, the shrinkage of relative prototype decreases or remain the same. There exist different ways of kernel mean shrinkage estimation that can be realized by constructing $h(\gamma_{i,c}, \lambda_{l,c})$ differently, such as Tikhonov regularization and Truncated SVD [28]. In this work we apply the Tikhonov regularization as the filter function that

$$h(\gamma_{i,c}, \lambda_{l,c}) = \frac{1}{\gamma_{i,c} + \lambda_{l,c}}. \tag{8}$$

In other ways, however, the kernel relative-prototype shrinkage estimation $\boldsymbol{\mu}_{l,c}(\lambda_{l,c})$ in Eq. 6 is not a computable form as the mapping function $\phi_{\omega}$ is not or in some cases can not be known, e. g., $\phi_{\omega} : \mathbb{R}^{d} \rightarrow \mathbb{R}^{\infty}$. Thus, we propose a computable form of $\boldsymbol{\mu}_{l,c}(\lambda_{l,c})$ based on the work by Muandet et al. [28].

**Theorem 1.** *Denote the $n \times n$ metrix $\boldsymbol{K}^{c}_{ss}$ whose entry at the row $i$ and the column $j$ $(\forall i, j)$ can be expressed the kernel form $k(f_{\theta}(\boldsymbol{s}_{i,c}), f_{\theta}(\boldsymbol{s}_{j,c})) = \phi_{\omega}(f_{\theta}(\boldsymbol{s_{i,c}}))^{T}\phi_{\omega}(f_{\theta}(\boldsymbol{s_{i,c}}))$, and the $n \times n$ metrix $\boldsymbol{K}^{l,c}_{qs}$ whose entry at the row $i$ and the column $j$ $(\forall i, j)$ can be expressed the kernel form $k(f_{\theta}(\boldsymbol{s}_{i,c}), f_{\theta}(\boldsymbol{q}_{l,c})) = \phi_{\omega}(f_{\theta}(\boldsymbol{s}_{i,c}))^{T}\phi_{\omega}(f_{\theta}(\boldsymbol{q}_{l,c}))$. The kernel kean shrinkage estimation of $\boldsymbol{\mu}_{l,c}$ in Eq. 6 can be expressed as:*

$$\boldsymbol{\mu}_{l,c}(\lambda_{l,c}) = \boldsymbol{\mu}_{l,c} - \sum_{i=1}^{n} \alpha_{l,i,c}(\lambda_{l,c})\boldsymbol{r}_{i,c}, \tag{9}$$

*with*

$$\boldsymbol{\alpha}_{l,c}(\lambda_{l,c}) = g^{h}(\tilde{\boldsymbol{K}}^{c}_{ss}, \lambda_{l,c})\tilde{\boldsymbol{K}}^{l,c}_{qs}\boldsymbol{I}_{n}, \tag{10}$$

$$\tilde{\boldsymbol{K}}^{c}_{ss} = \boldsymbol{K}^{c}_{ss} - \hat{\boldsymbol{I}}_{n}\boldsymbol{K}^{c}_{ss} - \boldsymbol{K}^{c}_{ss}\hat{\boldsymbol{I}}_{n} + \hat{\boldsymbol{I}}_{n}\boldsymbol{K}^{c}_{ss}\hat{\boldsymbol{I}}_{n}, \tag{11}$$

$$\tilde{\boldsymbol{K}}^{l,c}_{qs} = \boldsymbol{K}^{l,c}_{qs} - \hat{\boldsymbol{I}}_{n}\boldsymbol{K}^{l,c}_{qs} - \boldsymbol{K}^{c}_{ss} + \hat{\boldsymbol{I}}_{n}\boldsymbol{K}^{c}_{ss}, \tag{12}$$

*where $\boldsymbol{\alpha}_{l,c}(\lambda_{l,c}) = [\alpha_{l,1,c}(\lambda_{l,c}), ..., \alpha_{l,n,c}(\lambda_{l,c})]^{T}$ and $\boldsymbol{I}_{n} = [1/n, 1/n, ..., 1/n]^{T}$, and $\{\hat{\boldsymbol{I}}_{n}\}_{i,j} = 1/n$. Suppose the eigen-decomposition that $\tilde{\boldsymbol{K}}^{c}_{ss} = \boldsymbol{V}\boldsymbol{\Psi}\boldsymbol{V}^{\boldsymbol{T}}$, then*

$$g(\tilde{\boldsymbol{K}}^{c}_{ss}, \lambda_{l,c}) = \boldsymbol{V}g^{h}\left(\boldsymbol{\Psi}, \lambda_{l,c}\right)\boldsymbol{V}^{\boldsymbol{T}}, \tag{13}$$

*where $g(\boldsymbol{\Psi}, \lambda_{l,c}) = \mathrm{diag}(h(\gamma_{1,c}, \lambda_{l,c}), ..., h(\gamma_{n,c}, \lambda_{l,c}))$ with the $\tilde{\boldsymbol{K}}^{c}_{ss}$'s eigenvalues $\gamma_{1}, \gamma_{2}, ..., \gamma_{n}$.*

*Proof.* Suppose that $v_{i,j}$ is the entry at the row $i$ and column $j$ of $\boldsymbol{V}$. According to [39], we have $\mathbf{w}_{i,c} = (1/\sqrt{\gamma_i}) \sum_j^n v_{i,j} \boldsymbol{r}_{j,c}$,

$$\begin{aligned}
\boldsymbol{\mu}_{l,c}(\lambda_{l,c}) &= \boldsymbol{\mu}_{l,c} - \sum_{i=1}^{n} h(\gamma_{i,c}, \lambda_{l,c}) \gamma_{i,c} \langle \boldsymbol{\mu}_{l,c}, \mathbf{w}_{i,c} \rangle \mathbf{w}_{i,c} \\
&= \boldsymbol{\mu}_{l,c} - \sum_{j=1}^{n} \sum_{i=1}^{n} v_{i,j} h(\gamma_{i,c}, \lambda_{l,c}) \langle \boldsymbol{\mu}_{l,c}, \sum_{k}^{n} v_{i,k} \boldsymbol{r}_{k,c} \rangle \boldsymbol{r}_{j,c} \\
&= \boldsymbol{\mu}_{l,c} - \sum_{j=1}^{n} \alpha_{l,j,c}(\lambda_{l,c}) \boldsymbol{r}_{j,c}.
\end{aligned} \tag{14}$$

where

$$\begin{aligned}
\alpha_{l,j,c}(\lambda_{l,c}) &= \sum_{i=1}^{n} v_{i,j} h(\gamma_{i,c}, \lambda_{l,c}) \langle \boldsymbol{\mu}_{l,c}, \sum_{k=1}^{n} v_{i,k} \boldsymbol{r}_{k,c} \rangle \\
&= \sum_{i=1}^{n} v_{i,j} h(\gamma_{i,c}, \lambda_{l,c}) \sum_{k}^{n} v_{i,k} \langle \boldsymbol{\mu}_{l,c}, \boldsymbol{r}_{k,c.} \rangle
\end{aligned} \tag{15}$$

Suppose that $\boldsymbol{v_i} = [v_{i,1}, v_{i,2}, ..., v_{i,n}]^T$,

$$\begin{aligned}
\boldsymbol{\alpha}_{l,c}(\lambda_{l,c}) &= \sum_{i=1}^{n} \boldsymbol{v_i} h(\gamma_{i,c}, \lambda_{l,c}) \sum_{k}^{n} v_{i,k} \langle \boldsymbol{\mu}_{l,c}, \boldsymbol{r}_{k,c} \rangle \\
&= \sum_{i=1}^{n} \boldsymbol{v_i} h(\gamma_{i,c}, \lambda_{l,c}) \boldsymbol{v_i}^T \tilde{\boldsymbol{K}}_{qs}^{l,c} \boldsymbol{I}_n \\
&= g(\tilde{\boldsymbol{K}}_{ss}^{c}) \tilde{\boldsymbol{K}}_{qs}^{l,c} \boldsymbol{I}_n.
\end{aligned} \tag{16}$$

Based on Theorem 1, we can calculate the similarity between each sample in query set and prototype of each class using $\boldsymbol{\mu}_{l,c}(\lambda_{l,c})$.

### 3.3 Shrinkage Base Classifiers

For the convenience of calculation, we suppose that all the shrinkage parameters are the same, or $\lambda_{l,c} = \lambda$. The similarity of sample-pairs can be measured using the square of distance between the relative-prototype and original point in RKHS

$$\begin{aligned}
d_{l,c}^2(\lambda, \boldsymbol{S}_c, \boldsymbol{q}_l) &= \left\| \boldsymbol{\mu}_{l,c}(\lambda) \right\|^2 \\
&= \left( \boldsymbol{\alpha}_{l,c}(\lambda) \right)^T \tilde{\boldsymbol{K}}_{ss}^{c} \boldsymbol{\alpha}_{l,c}(\lambda) + \boldsymbol{I}_n^T \tilde{\boldsymbol{K}}_{qq}^{l,c} \boldsymbol{I}_n - 2 \left( \boldsymbol{\alpha}_{l,c}(\lambda) \right)^T \tilde{\boldsymbol{K}}_{qs}^{l,c} \boldsymbol{I}_n
\end{aligned} \tag{17}$$

where $\tilde{\boldsymbol{K}}_{qq}^{l,c}$ can be written as

$$\tilde{\boldsymbol{K}}_{qq}^{l,c} = \boldsymbol{K}_{qq}^{l,c} + \boldsymbol{K}_{ss}^{c} - \boldsymbol{K}_{qs}^{l,c} - \left(\boldsymbol{K}_{qs}^{l,c}\right)^{T}, \tag{18}$$

with the $n \times n$ metrix $\boldsymbol{K}_{qq}^{l,c}$ whose entry at the row $i$ and the column $j$ $(\forall i, j)$ can be expressed the kernel form $k(f_\theta(\boldsymbol{q}_l), f_\theta(\boldsymbol{q}_l)) = \phi_\omega(f_\theta(\boldsymbol{q}_l))^T \phi_\omega(f_\theta(\boldsymbol{q}_l))$.The probability of the sample $\boldsymbol{q}_l$ in query set belonging to class $c$ is

$$P_{\omega,\theta}\left(Y = c|\boldsymbol{q}_l\right) = \frac{\exp\left(-\zeta d_{l,c}^2\left(\lambda, \boldsymbol{S}_c, \boldsymbol{q}_l\right)\right)}{\sum_{c=1}^{C} \exp\left(-\zeta d_{l,c}^2\left(\lambda, \boldsymbol{S}_c, \boldsymbol{q}_l\right)\right)}, \tag{19}$$

where $\zeta$ is the metric scaling parameter. The loss function of DSFN is

$$\mathcal{L}(\omega, \theta) = -\frac{1}{m}\sum_{l=1}^{m} \log P_{\omega,\theta}\left(y_l|\boldsymbol{q}_l\right), \tag{20}$$

where $y_l$ is the label of $\boldsymbol{q}_l$. The few-shot learning process with the proposed DSFN is shown as Algorithm 1.

**Algorithm 1** Few-shot learning with the proposed DSFN

**Input**: Support set $\mathbf{S}$ and query set $\mathbf{Q}$, learning rate $\alpha$.
**Output**: $\theta$

Initialize $\theta$ randomly;
**for** $t = 1$ to $T$ **do**
  Generate episode by randomly sampling $\mathbf{S}^{(t)}$ from $\mathbf{S}$ and $\mathbf{Q}^{(t)}$ from $\mathbf{Q}$;
  **for** $c = 1$ to $C$ **do**
    **for** $l = 1$ to $m$ **do**
      Compute $\tilde{\boldsymbol{K}}_{ss}^{c}$, $\tilde{\boldsymbol{K}}_{qs}^{l,c}$ and $\tilde{\boldsymbol{K}}_{qq}^{l,c}$ using Eq. 11, Eq. 12 and Eq. 18, respectively, where $\boldsymbol{K}_{ss}^{c}$, $\boldsymbol{K}_{qs}^{l,c}$ and $\boldsymbol{K}_{qq}^{l,c}$ are calculated using the samples in $\mathbf{S}^{(t)}$ and $\mathbf{Q}^{(t)}$;
      Compute $\boldsymbol{\alpha}_{l,c}(\lambda_{l,c})$ with Eq. 10,using $\tilde{\boldsymbol{K}}_{qs}^{l,c}$ and eigenvalue decomposition of $\tilde{\boldsymbol{K}}_{ss}^{c}$;
      Compute $d_{l,c}\left(\lambda^{(t)}, \boldsymbol{S}_c^{(t)}, \boldsymbol{q}_l^{(t)}\right)$ using Eq. 17;
    **end for**
  **end for**
  Compute the loss function using Eq. 20;
  Update $\theta$ with $\theta - \alpha\nabla_\theta\mathcal{L}(\omega, \theta)$.
**end for**

### 3.4 Relationship to Other Methods

Here we discuss the connection of our class representation to the point type (e. g., ProtoNet) and subspace type (e. g., DSN) representations. In fact, they are different mainly because they use different filter functions.

**Relationship to ProtoNet.** As the filter function $h(\gamma_{i,c}, \lambda_{l,c}) = 0$ instead of Tikhonov regularization that causes the disappearance of shrinkage effect, and the map function $\phi_\omega$ is identical, the proposed framework (Eq. 6) is embodied as ProtoNet.

**Relationship to DSN.** While using the Truncated SVD as the filter function that $h(\gamma_{i,c}, \lambda_{l,c}) = \mathbb{I}_{(\gamma_{i,c} \geq \lambda_{l,c})} \gamma_{i,c}^{-1}$ instead of Tikhonov regularization, where $\mathbb{I}_{(\gamma_{i,c} \geq \lambda_{l,c})} \gamma_{i,c}^{-1}$ is the indicative function that is 1 if $\gamma_{i,c} \geq \lambda_{l,c}$ else 0, and the map function $\phi_\omega$ is identical, the proposed framework (Eq. 6) is embodied as DSN. In this case, by setting different values of $\lambda_{l,c}$, different dimension of subspace can be selected. Formally, the relationship of DSFN and DSN is shown in Theorem 2 (see detailed proof in supplementary material).

**Theorem 2.** *Suppose that: 1)* $h(\gamma_{i,c}, \lambda_{l,c}) = \mathbb{I}_{(\gamma_{i,c} \geq \lambda_{l,c})} \gamma_{i,c}^{-1}$*; 2)* $\zeta = 1$*; 3)* $\lambda_{l,c} =$ *constant for all* $l$ *and* $c$*; 4) the map function* $\phi_\omega$ *is identical. Eq. 17 is reduced to* $d^2_{l,c}(\lambda, \boldsymbol{S}_c, \boldsymbol{q}_l) = \|(I - P_c P_c^T)(f_\theta(\boldsymbol{q}_l) - \boldsymbol{\mu}_c)\|^2$ *with* $P_c$ *the truncated matrix of* $W_c$*, where* $W_c$ *is the eigenvector matrix of empirical covariance matrix* $C$*.*

Theorem 2 implies that, while using Truncated SVD as the filter function, the loss function of our proposed framework (Eq. 19) can be reduced to the loss of DSN with no regularization (See Eq. 5 in the work by Simon et al.[40]).

## 4 Experiments Setup

### 4.1 Datasets

***mini*ImageNet**. *mini*ImageNet dataset [44] was often used for few-shot learning, which contains a total of 60,000 color images in 100 classes randomly selected from ILSVRC-2012, with 600 samples in each class. The size of each image is 84 × 84. In the data set, the training set, validation set and test set contains the number of classes with 64 : 16 : 20.

***tiered*-ImageNet**. The *tiered*-ImageNet dataset [36] is a benchmark image dataset that is also selected from ILSVRC-2012 but contains 608 classes that is more than that in *mini*ImageNet dataset. These classes are divided into 34 high-level categories, can each category contains 10 to 30 classes. The size of each image is 84 × 84. Further, the categories are divided into the training set, validation set and test set with 20 : 6 : 8.

**CIFAR-FS**. The CIFAR-FS dataset [4] is a few-shot learning benchmark containing all 100 classes from CIFAR-100 [18], and each class contains 600 samples. The size of each image is 32 × 32. The classes are divided into the training set, validation set and test set with 64 : 16 : 20.

### 4.2 Implementation

In training stage, 15-shot 10-query samples are chosen on *mini*ImageNet dataset; 10-shot 15-query samples are chosen on *tiered*ImageNet dataset; 2-shot 20-query samples are chosen for 1-shot task and 15-shot 10-query samples are chosen for

5-shot task on CIFAR-FS dataset. $\lambda$ is set to the best by choosing 0.01,0.1,1,10 or 100. For these datasets, the setting of 8 episodes per batch is utilized in the experiments. The total number of training epochs is 80, and in each epoch 1000 batches are sampled. In testing stage, 1000 episodes are used to assess our model. For the 1-shot K-way learning, another support sample was created by flipping the original support sample, and two support samples are used for spectral filtering in the validation and testing stages. Our model is trained and tested in the PyTorch machine learning package [34].

Two backbones, the Conv-4 and Resnet-12, are utilized as the backbones in our model. For Conv-4, the Adam optimizer with default momentum values ($[\beta_1, \beta_2] = [0.9, 0.999]$) is applied for the training. The learning rate is initially set as 0.005 then decayed to 0.0025, 0.00125, 0.0005 and 0.00025 at 8, 30, 45 and 50 epochs, respectively. For ResNet-12, the SGD optimizer is applied for the training, and the learning rate is initially set as 0.1 then decayed to 0.0025, 0.00032, 0.00014 and 0.000052 at 12, 30, 45 and 57 epochs, respectively.

The identity kernel $k^{ide}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle$ and the RBF kernel $k^{rbf}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \exp\left(-\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2/(2\sigma^2)\right)$ are chosen as the kernel functions, where $\sigma^2$ is assigned as the dimension of embeddings. In addition, the scaling parameter $\zeta$ is learned as a variable. For the filter function, we set the shrinkage coefficient $\lambda$ as the fixed multiple of the maximum eigenvalue with each class.

## 5 Experiments and Discussions

### 5.1 Comparison with State-of-the-art Methods

**Results on *mini*ImageNet dataset.** Firstly, the proposed DSFN and the state-of-the art methods for 5-way classification tasks on *mini*ImageNet dataset are compared in Table 1. Table 1 shows that the proposed DSFN with identity kernel can achieve the bests on 5-way 5-shot classification tasks using both Conv-4 and ResNet-12 backbones, which are higher than DSN with 3.3% and 1.3%, respectively. The RBF kernel can achieve the second best on 5-way 5-shot classification tasks. These results illustrate that the proposed DSFN can achieve the state-of-the-art performance for 5-way 5-shot classification tasks on the dataset.

**Results on *tiered*-ImageNet dataset.** We compare the proposed DSFN with the state-of-the-art methods for 5-way classification tasks on *tiered*ImageNet dataset, as shown in Table 2. It can be seen that, the performance of DSFN with identity kernel and RBF kernel is slightly lower than DSN on 5-way 5-shot classification task. However, they are better than the others on 5-way 5-shot classification task.

**Results on CIFAR-FS dataset.** A further comparison is made on CIFAR-FS dataset, as shown in Table 3. Table 3 shows that the proposed DSFN with RBF kernel performs the best on 5-way 5-shot classification task, whose test accuracy is about 1.2% and 2.8% higher than those of DSN and ProtoNet, respectively. Thus, the proposed DSFN performs the state-of-the-art for 5-way 5-shot classification task on the dataset.

Table 1: Test accuracies (%) from the proposed DSFN and the state-of-the art methods for 5-way tasks on *mini*ImageNet dataset with 95% confidence intervals. ‡ means that training set and validation set are used for training the corresponding model.

| **Model** | **Backbone** | **5-way** | |
|---|---|---|---|
| | | **1-shot** | **5-shot** |
| MatchingNet[44] | Conv-4 | $43.56 \pm 0.84$ | $55.31 \pm 0.73$ |
| MAML [10] | Conv-4 | $48.70 \pm 1.84$ | $63.11 \pm 0.92$ |
| Reptile [31] | Conv-4 | $49.97 \pm 0.32$ | $65.99 \pm 0.58$ |
| ProtoNet[41] | Conv-4 | $44.53 \pm 0.76$ | $65.77 \pm 0.66$ |
| Relation Nets [42] | Conv-4 | $50.44 \pm 0.82$ | $65.32 \pm 0.70$ |
| DSN[40] | Conv-4 | $51.78 \pm 0.96$ | $68.99 \pm 0.69$ |
| **DSFN(identity kernel)** | Conv-4 | $\mathbf{50.21 \pm 0.64}$ | $\mathbf{72.20 \pm 0.51}$ |
| **DSFN(RBF kernel)** | Conv-4 | $\mathbf{49.97 \pm 0.63}$ | $\mathbf{72.04 \pm 0.51}$ |
| SNAIL[24] | ResNet-12 | $55.71 \pm 0.99$ | $68.88 \pm 0.92$ |
| TADAM[32] | ResNet-12 | $58.50 \pm 0.30$ | $76.70 \pm 0.30$ |
| AdaResNet[30] | ResNet-12 | $56.88 \pm 0.62$ | $71.94 \pm 0.57$ |
| LEO‡[38] | WRN-28-10 | $61.76 \pm 0.08$ | $77.59 \pm 0.12$ |
| LwoF[13] | WRN-28-10 | $60.06 \pm 0.14$ | $76.39 \pm 0.11$ |
| wDAE-GNN‡[14] | WRN-28-10 | $62.96 \pm 0.15$ | $78.85 \pm 0.10$ |
| MetaOptNet-SVM[20] | ResNet-12 | $62.64 \pm 0.61$ | $78.63 \pm 0.46$ |
| DSN[40] | ResNet-12 | $62.64 \pm 0.66$ | $78.83 \pm 0.45$ |
| CTM[21] | ResNet-18 | $62.05 \pm 0.55$ | $78.63 \pm 0.06$ |
| Baseline[6] | ResNet-18 | $51.75 \pm 0.80$ | $74.27 \pm 0.63$ |
| Baseline++[6] | ResNet-18 | $51.87 \pm 0.77$ | $75.68 \pm 0.63$ |
| Hyper ProtoNet[16] | ResNet-18 | $59.47 \pm 0.20$ | $76.84 \pm 0.14$ |
| Hyperbolic RBF kernel[8] | ResNet-18 | $60.91 \pm 0.21$ | $77.12 \pm 0.15$ |
| **DSFN(identity kernel)** | ResNet-12 | $\mathbf{61.27 \pm 0.71}$ | $\mathbf{80.13 \pm 0.17}$ |
| **DSFN(RBF kernel)** | ResNet-12 | $\mathbf{59.43 \pm 0.66}$ | $\mathbf{79.60 \pm 0.46}$ |

Table 2: Test accuracies (%) from the proposed DSFN and the state-of-the art methods for 5-way tasks on *tiered*-ImageNet dataset with 95% confidence intervals. ‡ means that training set and validation set are used for training the corresponding model.

| **Model** | **Backbone** | **5-way** | |
|---|---|---|---|
| | | **1-shot** | **5-shot** |
| ProtoNet[41] | ResNet-12 | $61.74 \pm 0.77$ | $80.00 \pm 0.55$ |
| CTM[21] | ResNet-18 | $64.78 \pm 0.11$ | $81.05 \pm 0.52$ |
| LEO‡[38] | WRN-28-10 | $66.33 \pm 0.05$ | $81.44 \pm 0.09$ |
| MetaOptNet-RR[20] | ResNet-12 | $65.36 \pm 0.71$ | $81.34 \pm 0.52$ |
| MetaOptNet-SVM[20] | ResNet-12 | $65.99 \pm 0.72$ | $81.56 \pm 0.53$ |
| DSN [40] | ResNet-12 | $66.22 \pm 0.75$ | $82.79 \pm 0.48$ |
| **DSFN(identity kernel)** | ResNet-12 | $\mathbf{65.46 \pm 0.70}$ | $\mathbf{82.41 \pm 0.53}$ |
| **DSFN(RBF kernel)** | ResNet-12 | $\mathbf{64.27 \pm 0.70}$ | $\mathbf{82.26 \pm 0.52}$ |

Table 3: Test accuracies (%) from the proposed DSFN and some state-of-the art methods for 5-way classification tasks on CIFAR-FS dataset with 95% confidence intervals.

| Model | Backbone | 5-way | |
|---|---|---|---|
| | | 1-shot | 5-shot |
| ProtoNet[41] | ResNet-12 | $72.2 \pm 0.7$ | $83.5 \pm 0.5$ |
| MetaOptNet-RR[20] | ResNet-12 | $72.6 \pm 0.7$ | $84.3 \pm 0.5$ |
| MetaOptNet-SVM[20] | ResNet-12 | $72.0 \pm 0.7$ | $84.2 \pm 0.5$ |
| DSN[40] | ResNet-12 | $72.3 \pm 0.7$ | $85.1 \pm 0.5$ |
| **DSFN(identity kernel)** | ResNet-12 | $\mathbf{70.62 \pm 0.79}$ | $\mathbf{86.11 \pm 0.58}$ |
| **DSFN(RBF kernel)** | ResNet-12 | $\mathbf{71.28 \pm 0.70}$ | $\mathbf{86.30 \pm 0.58}$ |

### 5.2 Ablation Study

**The impact of shrinkage parameter.** The influence of different values of shrinkage parameter $\lambda$ on the performances of the proposed DSFN is shown in Fig. 3. Fig. 3 shows that the general trends drop for these datasets as the shrinkage parameter increases from 0.01 to 100, and the descending trends with 5 shot is more obvious that those with 1 shot. These results indicate that smaller shrinkage parameters (e.g., 1, 0.1, 0.01) or stronger shrinkage effect can better improve the performance of the proposed model.

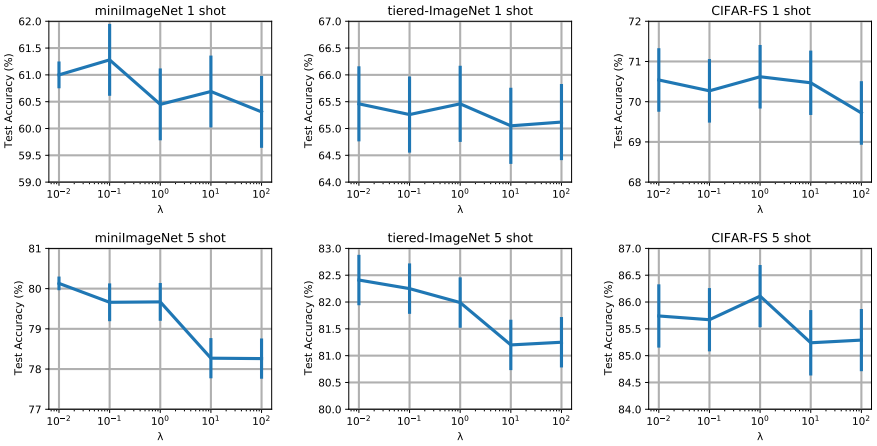


Fig. 3: Test accuracies on few-shot classification tasks from the proposed DSFN against different values of shrinkage parameter, where ResNet-12 is used.

**The effectiveness of shrinkage.** We show an ablation study to illustrate the effectiveness of shrinkage in our work, as shown in Table 4. In this experiment, the performances of identity kernel and RBF kernel with and without shrinkage in few-shot classification tasks are compared. Table 4 shows that, the proposed

Table 4: Accuracies (%) from models with and without shrinkge for 5-way 1-shot and 5-way 5-shot classification tasks, w S: with shrinkge, w/o S: without shrinkge.

| Dataset | Kernel | w S | w/o S | 1-shot | 5-shot |
|---|---|---|---|---|---|
| ***mini*ImageNet** | identity | | ✓ | $60.14 \pm 0.67$ | $77.65 \pm 0.52$ |
| | identity | ✓ | | $61.00 \pm 0.25$ | $80.13 \pm 0.17$ |
| | RBF | | ✓ | $58.74 \pm 0.65$ | $79.18 \pm 0.46$ |
| | RBF | ✓ | | $59.43 \pm 0.66$ | $79.60 \pm 0.46$ |
| ***tiered*-ImageNet** | identity | | ✓ | $65.05 \pm 0.72$ | $81.14 \pm 0.55$ |
| | identity | ✓ | | $65.46 \pm 0.70$ | $82.41 \pm 0.53$ |
| | RBF | | ✓ | $64.23 \pm 0.70$ | $82.07 \pm 0.53$ |
| | RBF | ✓ | | $64.27 \pm 0.70$ | $82.26 \pm 0.52$ |
| **CIFAR-FS** | identity | | ✓ | $70.54 \pm 0.82$ | $85.30 \pm 0.59$ |
| | identity | ✓ | | $70.62 \pm 0.79$ | $86.11 \pm 0.58$ |
| | RBF | | ✓ | $71.18 \pm 0.73$ | $86.09 \pm 0.47$ |
| | RBF | ✓ | | $71.28 \pm 0.70$ | $86.30 \pm 0.46$ |

model with shrinkage performs better than those without shrinkage for both kernels. In addition, the improvement of shrinkage on 5-way 5-shot classification tasks is more obvious than those on 5-way 1-shot classification tasks, probably because the eigenvalues and eigenvectors with one shot is hard to learn.

### 5.3 Time Complexity

Our proposed DSFN approach has the time complexity of $\mathcal{O}(\max(CN^3, CN^2D))$, where $C$, $N$, $D$ are the number of way, shot and feature dimensionality, respectively. The proposed DSFN approach is slower than DSN ($\mathcal{O}(\min(CND^2, CN^2D))$) and ProtoNet ($\mathcal{O}(CND)$) due to the kernel matrix calculation, eigen-decomposition and multiple matrix multiplication. A way to reduce the time complexity is using some efficient algorithms, such as the fast adaptive eigenvalue decomposition [7] and faster matrix multiplication [1].

## 6 Conclusion

In this work, we propose a framework called DSFN for few-shot learning. In this framework, one can represent the similarity between a query and a prototype as the distance after spectral filtering of support set in each class in RKHS. DSFN is an extension of some mainstream methods, e. g., ProtoNet and DSN, and with appropriate filter function, the framework of DSFN can be embodies as those methods. In addition, we also showed that in this framework, one can explore new methods by applying diverse forms, e. g., Tikhonov regularization, as the filter function, and diverse forms of kernels. Several experiments verified the effectiveness of various specific forms of the proposed DSFN. Future works should take a closer look at the selection of filter function and the role of shrinkage parameter in the proposed framework.

## References


1. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 522–539. SIAM (2021)
2. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: Advances in neural information processing systems. pp. 3981–3989 (2016)
3. Bateni, P., Goyal, R., Masrani, V., Wood, F., Sigal, L.: Improved few-shot visual classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14493–14502 (2020)
4. Bertinetto, L., Henriques, J.F., Torr, P.H., Vedaldi, A.: Meta-learning with differentiable closed-form solvers. arXiv preprint arXiv:1805.08136 (2018)
5. Chen, J., Zhan, L., Wu, X., Chung, F.: Variational metric scaling for metric-based meta-learning. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence. pp. 3478–3485 (2020)
6. Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C.F., Huang, J.B.: A closer look at few-shot classification. arXiv preprint arXiv:1904.04232 (2019)
7. Chonavel, T., Champagne, B., Riou, C.: Fast adaptive eigenvalue decomposition: a maximum likelihood approach. Signal processing **83**(2), 307–324 (2003)
8. Fang, P., Harandi, M., Petersson, L.: Kernel methods in hyperbolic spaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10665–10674 (2021)
9. Feifei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(4), 594–611 (2006)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning. p. 1126–1135. JMLR. org. (2017)
11. Fort, S.: Gaussian prototypical networks for few-shot learning on omniglot. arXiv preprint arXiv:1708.02735 (2017)
12. Gao, T., Han, X., Liu, Z., Sun, M.: Hybrid attention-based prototypical networks for noisy few-shot relation classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6407–6414 (2019)
13. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4367–4375 (2018)
14. Gidaris, S., Komodakis, N.: Generating classification weights with gnn denoising autoencoders for few-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 21–30 (2019)
15. Kang, D., Kwon, H., Min, J., Cho, M.: Relational embedding for few-shot classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8822–8833 (2021)
16. Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., Lempitsky, V.: Hyperbolic image embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6418–6428 (2020)
17. Koch, G., Zemel, R., Salakhutdinov, R., et al.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. vol. 2 (2015)
18. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

19. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction **350**(6266), 1332–1338 (2015)
20. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. arXiv:1904.03758 pp. 10657–10665 (2019)
21. Li, H., Eigen, D., Dodge, S., Zeiler, M., Wang, X.: Finding task-relevant features for few-shot learning by category traversal. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1–10 (2019)
22. Liu, J., Song, L., Qin, Y.: Prototype rectification for few-shot learning. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. pp. 741–756 (2020)
23. Mangla, P., Kumari, N., Sinha, A., Singh, M., Krishnamurthy, B., Balasubramanian, V.N.: Charting the right manifold: Manifold mixup for few-shot learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2218–2227 (2020)
24. Mishra, N., Rohaninejad, M., Chen, X., Abbeel, P.: A simple neural attentive meta-learner. In: 6th International Conference on Learning Representations (2018)
25. Muandet, K., Fukumizu, K., Sriperumbudur, B., Gretton, A., Scholkopf, B.: Kernel mean estimation and stein effect. In: International Conference on Machine Learning. pp. 10–18 (2014)
26. Muandet, K., Fukumizu, K., Sriperumbudur, B., Scholkopf, B.: Kernel mean embedding of distributions: A review and beyond. arXiv preprint arXiv:1605.09522 (2016)
27. Muandet, K., Sriperumbudur, B., Fukumizu, K., Gretton, A., Scholkopf, B.: Kernel mean shrinkage estimators. Journal of Machine Learning Research **17** (2016)
28. Muandet, K., Sriperumbudur, B., Scholkopf, B.: Kernel mean estimation via spectral filtering. arXiv preprint arXiv:1411.0900 (2014)
29. Munkhdalai, T., Yu, H.: Meta networks. In: Proceedings of the 34th International Conference on Machine Learning. pp. 2554–2563 (2017)
30. Munkhdalai, T., Yuan, X., Mehri, S., Trischler, A.: Rapid adaptation with conditionally shifted neurons. In: International Conference on Machine Learning. pp. 3664–3673 (2018)
31. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
32. Oreshkin, B.N., Lopez, P.R., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. In: Advances in Neural Information Processing Systems. pp. 721–731 (2018)
33. Pahde, F., Puscas, M., Klein, T., Nabi, M.: Multimodal prototypical networks for few-shot learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2644–2653 (2021)
34. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
35. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning (2016)
36. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676 (2018)
37. Rodríguez, P., Laradji, I., Drouin, A., Lacoste, A.: Embedding propagation: Smoother manifold for few-shot classification. In: European Conference on Computer Vision. pp. 121–138 (2020)
38. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960 (2018)

39. Scholkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural computation **10**(5), 1299–1319 (1998)
40. Simon, C., Koniusz, P., Nock, R., Harandi, M.: Adaptive subspaces for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4136–4145 (2020)
41. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. pp. 4077–4087 (2017)
42. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H.S., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1199–1208 (2018)
43. Vanschoren, J.: Meta-learning: A survey. arXiv:1810.03548 (2018)
44. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Advances in neural information processing systems. pp. 3637–3645 (2016)
45. Xing, C., Rostamzadeh, N., Oreshkin, B., O Pinheiro, P.O.: Adaptive cross-modal few-shot learning. Advances in Neural Information Processing Systems **32**, 4847–4857 (2019)
46. Ye, H.J., Hu, H., Zhan, D.C., Sha, F.: Few-shot learning via embedding adaptation with set-to-set functions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8808–8817 (2020)
47. Zhang, H., Koniusz, P.: Zero-shot kernel learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7670–7679 (2018)
48. Ziko, I., Dolz, J., Granger, E., Ayed, I.B.: Laplacian regularized few-shot learning. In: International Conference on Machine Learning. pp. 11660–11670 (2020)