One-Shot Image Classification by Learning to Restore Prototypes

Wanqi Xue
Department of Computer Science
National University of Singapore
wanqixue0@gmail.com

Wei Wang
Department of Computer Science
National University of Singapore
wangwei@comp.nus.edu.sg

Abstract

One-shot image classification aims to train image classifiers over the dataset with only one image per category. It is challenging for modern deep neural networks that typically require hundreds or thousands of images per class. In this paper, we adopt metric learning for this problem, which has been applied for few- and many-shot image classification by comparing the distance between the test image and the center of each class in the feature space. However, for one-shot learning, the existing metric learning approaches would suffer poor performance because the single training image may not be representative of the class. For example, if the image is far away from the class center in the feature space, the metric-learning based algorithms are unlikely to make correct predictions for the test images because the decision boundary is shifted by this noisy image. To address this issue, we propose a simple yet effective regression model, denoted by RestoreNet, which learns a class agnostic transformation on the image feature to move the image closer to the class center in the feature space. Experiments demonstrate that RestoreNet obtains superior performance over the state-of-the-art methods on a broad range of datasets. Moreover, RestoreNet can be easily combined with other methods to achieve further improvement.

1 Introduction

Over the past decade, we have witnessed the great success of deep learning in computer vision. With large amounts of annotated data, deep learning models achieved impressive breakthroughs again and again (He et al. 2016; Krizhevsky, Sutskever, and Hinton 2012; Huang et al. 2017). However, in practical applications, large quantities of labeled data is expensive or sometimes impossible to acquire. In such a situation where only a few samples per category are available, both training from scratch and fine-tuning on the small dataset are likely to cause severe overfitting, leading to poor recognition performance. Humans, in contrast, have the ability to quickly learn a new concept from one or a few examples. The significant gap between human and machine intelligence encourages the interest of researchers. Many endeavours have been done to narrow the gap (Finn, Abbeel, and Levine 2017; Snell, Swersky, and Zemel 2017; Vinyals et al. 2016; Ravi and Larochelle 2016; Munkhdalai and Yu 2017).

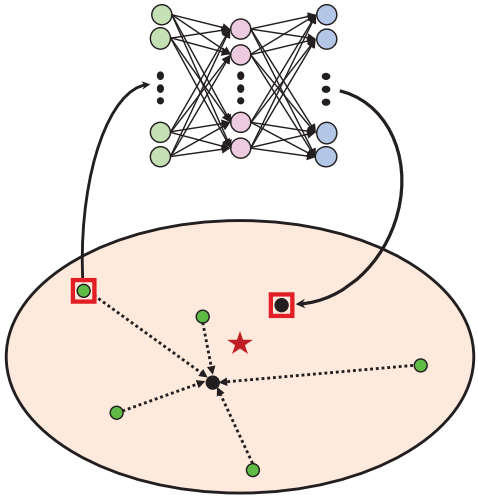


Figure 1: The challenge for metric learning in one-shot image classification and our solution to it. Each green point represents a training image. Dark points denote the prototypes and the red star marks the center of the class.

A popular category of solutions is based on meta-learning, where a meta-learner is trained to generate classifiers. The training is conducted in episodic manner, where the each episode is constituted by two sets of data, namely the support set and the query set. The generated classifier is trained over the support set and evaluated on the query set. The meta learner is then updated based on the evaluation performance. The idea is to transfer some class agnostic knowledge from the training data to the test data via the meta learner. Different meta-learning approaches have been proposed. Metric-learning-based methods, e.g. Prototypical network (Snell, Swersky, and Zemel 2017), train the meta-leaner to transform the images into a metric space where nearest neighbour classifiers can be applied. MAML trains the meta-leaner to learn a good initialization state (Finn, Abbeel, and Levine 2017; Li et al. 2017) for the convolutional neural network

Figure 2: 2D visualization of image features for one-shot classification of two classes, namely Alaskan Malamute (framed in pink) and Golden Retriever (framed in blue). After restoration, the original prototype of Malamute (framed in red, left) is moved to the right, which is closer to the class center (marked by the red star). We visualize the shifted prototype (right) using its nearest real image. The figure is plotted by applying t-SNE of ResNet18 features of samples from *mini*ImageNet. Best viewed in color.

(ConvNet) classifiers.

Metric-learning-based approaches are simple and effective. However, they would suffer from poor performance for one-shot learning, i.e., learning from only one single example per category. Take Figure 1 as an example, when there are more training images, the average of these images' features, denoted as the prototype, is more likely to be around the real class center, even though some images are far away from the center. In contrast, if there is only one single training image and it is far away from the center, then the nearest neighbour classifier is unlikely to make correct predictions for the test images as the decision boundary is shifted by this noisy image (i.e., the prototype).

In this paper, we focus on one-shot learning and propose a simple solution towards the issue mentioned above. The intuition of our solution is to train a transformation network in the feature space to move the noisy training image close to the center of the cluster. During training, *RestoreNet* learns from training pairs constituted by the features of noisy images and their corresponding class prototypes. Each class prototype is constructed using many images from the class and thus is reliable. During test, the feature of the image from the support set is fed into *RestoreNet*. We average the original feature and the transformed feature to get the final image representation, which is used as the prototype of its class for the classification of the query images (via nearest neighbour classification). Figure 2 presents the visualization of the image features for one-shot classification of two classes, namely Alaskan Malamute and Golden Retriever. The example training image of Alaskan Malamute (framed in red, left) merely includes the head and its representation (i.e., the prototype) is located far from the center of its cluster (marked by red star). After restoration, the original prototype is moved to the right, which is closer to the class center.

Our contribution is three-fold: firstly, we identify the challenge of metric-learning-based approaches for one-shot learning. Secondly, we propose a simple method, i.e., *RestoreNet*, to address the challenge by moving the generated class prototype closer to the class center in the feature space. The proposed model can be combined with other methods easily and realize further enhancement. Finally, experiments on four benchmark datasets demonstrate that our model improves significantly over the state-of-the-art methods for one-shot learning tasks.

2 Related Work

One-shot (resp. few-shot) learning requires the classifiers to quickly adapt to new classes using only one (resp. few) example from each target class. Fine-tuning classifier on such sparse data is likely to get severe overfitting. To address this problem, different approaches have been proposed.

Data augmentation (Schwartz et al. 2018; Chen et al. 2018; Wang et al. 2018b; Hariharan and Girshick 2017) resolve the data issue directly by data augmentation. Delta-encoder (Schwartz et al. 2018) applies the extracted intra-class deformations or "deltas" to the one-shot (resp. few-shot) example of a novel class to generate new samples. (Wang et al. 2018b; Chen et al. 2018) trains a network which can effectively mix noise with image representations. They generate data by applying different noise. Another way to increase the training dataset is via self-training (Rosenberg, Hebert, and Schneiderman 2005). In self-training framework, a predictor is first learned on the initial training set. Then the predictor is applied to predict the labels of a set of unlabeled images. These images are added to the training set to re-train the original predictor. In (Ren et al. 2018),

the unlabeled images are assigned with weights before they are added to the training set, which are used to sample the images to create the training batches. In (Chen et al. 2019), images from the original training set and the unlabeled images are edited to synthesize new images. Our proposed solution is orthogonal to these data augmentation methods. To confirm it, we adopt the first approach in one of our experiments.

Meta-learning Many recent works (Santoro et al. 2016; Munkhdalai and Yu 2017; Sung et al. 2018; Vinyals et al. 2016; Snell, Swersky, and Zemel 2017; Wang and Hebert 2016; Wang, Ramanan, and Hebert 2017) follow the meta-learning paradigm. They train a meta-learner over a series of training episodes. The meta-learner then generates the classifier for the target task by exploiting the accumulated class agnostic knowledge. For example, (Finn, Abbeel, and Levine 2017; Li et al. 2017; Ravi and Larochelle 2016) train the meta-learner to find a good initialization state and/or learn an effective optimizer, which are applied to optimize the classifier for the target task. Then the classifier can generalize better to new data within a few gradient-descent update steps. Metric learning (Sung et al. 2018; Vinyals et al. 2016; Snell, Swersky, and Zemel 2017) based approaches can also be formalized under the meta-learning paradigm. They aim to learn a feature space where the prototype of each class is the center of the training images in the class. Nearest neighbour classification is applied to classify the query images using Euclidean distance, cosine distance, etc. Recently, (Wang and Hebert 2016; Wang, Ramanan, and Hebert 2017) propose to train a meta-learner to transform the classifier trained over few examples to the classifier trained over many examples by adapting the classifier parameters. This kind of approaches are somewhat similar to *RestoreNet*. However, there are mainly two differences: 1) the motivation. We try to improve the performance by adjusting the FEATURE of NOISY examples. Therefore, we train *RestoreNet* using the farthest example, which are considered as noisy examples. (Wang and Hebert 2016) tries to improve the performance by adjusting the few-shot CLASSIFIER (parameters) to be similar to the many-shot classifier where the transform model is trained using RANDOMLY selected few-shot classifiers. 2) consequently, different techniques are applied. We get the FEATURE of each example by averaging (like ensembling) the original feature and transformed feature, whereas (Wang and Hebert 2016) uses the transformed model as biased regularization.

3 Methodology

3.1 Background

Problem Definition For N-way K-shot learning, we are given a support set of labeled images $\mathcal{S}_{novel} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N*K}$, where $\boldsymbol{x_i}$ is the image, $y_i \in C_{novel}$ is the label, C_{novel} is the class set, N is the number of classes in $\mathcal{S}_{novel}$ ($C_{novel} \geq N$) and K is the number of images per class in $\mathcal{S}_{novel}$. For one-shot learning, $K = 1$. The task is to train a image classifier over $\mathcal{S}_{novel}$. Typically, we also have an additional dataset $\mathcal{D}_{base} = \{(\boldsymbol{x_i}, y_i)\}$, where $y_i \in C_{base}$, and $C_{base} \cap C_{novel} = \emptyset$. $\mathcal{D}_{base}$ has many images for each class in C_{base}.

Episodic Training O. Vinyals et al. (Vinyals et al. 2016) propose an episodic paradigm for meta-learning based few-shot learning. For N-way K-shot learning, a training episode is constructed by sampling N classes from C_{base} ($|C_{base}| >> N$), K images for each of these classes from $\mathcal{D}_{base}$ as the support set, and multiple query images for each of these classes from $\mathcal{D}_{base}$ as the query set. The classifier is trained over the support set and evaluated on the query set. The evaluation result is used to update the meta-learner. The idea behind this paradigm is to mimic the test setting during training, taking advantage of large amounts of labeled data in $\mathcal{D}_{base}$.

Prototypical Network J. Snell et al. (Snell, Swersky, and Zemel 2017) propose this simple yet effective model for few-shot learning. Following the episodic paradigm, it learns an embedding function $f(\boldsymbol{x})$ via ConvNet and generates a prototype of each class via Equation 1. Then the probability for a query image from class c is calculated via Equation 2. The embedding network is trained by feeding the probability and the ground truth label of the query image into the cross-entropy loss. During testing, Prototypical Network applies Nearest Neighbor (*NN*) classifier for each query image, assigning it with the label of their nearest prototype.

$$\boldsymbol{p}_c = \frac{1}{|S_c|} \sum_{(\boldsymbol{x_i}, y_i) \in S_c} f(\boldsymbol{x_i}) \quad (1)$$

$$P(y = c|\boldsymbol{x}) = \frac{e^{d(f(\boldsymbol{x}), \boldsymbol{p}_c)}}{\sum_{c'} e^{d(f(\boldsymbol{x}), \boldsymbol{p}_{c'})}} \quad (2)$$

3.2 Learning to Restore Prototypes

Even though Prototypical Network has shown to be effective for few-shot learning, its performance for one-short learning is barely satisfactory. We conjecture that the inaccurate classification is due to the learned prototypes. When there is only one image per class provided for training, the prototype is just the feature of this image, which would be far away from the class center if the image is not discriminative for this class. Consequently, the classification boundary would be shifted into a inappropriate position. In this section, we introduce our proposed model, dubbed as *RestoreNet*, to "restore" the prototype, i.e., moving it closer to the class center where the true prototype is more likely to situate.

Firstly, we adapt Prototypical Network to train the feature embedding function $f(\boldsymbol{x})$. Different to the original Prototypical Network, following (Zhou, Wu, and Li 2018), we add an additional label classification branch as a regularization, which consists of a two-layer MLP network, a Softmax layer and a cross-entropy loss. The parameters of the embedding network are trained w.r.t the summation of the new loss and the original loss from Prototypical Network. Once this step is done, we freeze the parameters in $f(\cdot)$ and use it just for feature extraction. We use the same naming style in (Zhou, Wu, and Li 2018) and denote this network as DEML+Prototypical Nets. This adaption is able to realize more than one percent enhancement over the original Pro-

Table 1: Summary of the datasets

Datasets	Number of images	Seen classes	Unseen classes	Resolution	Fine-grained	Strictly balanced
*mini*ImageNet	60,000	64 + 16	20	Medium	No	Yes
CIFAR-100	60,000	64 + 16	20	Low	No	Yes
Caltech-256	30,607	100 + 56	50	High	No	No
CUB-200	11,788	100 + 50	50	High	Yes	No

totypical Network. We use DEML+Prototypical Nets as the baseline in our experiments.

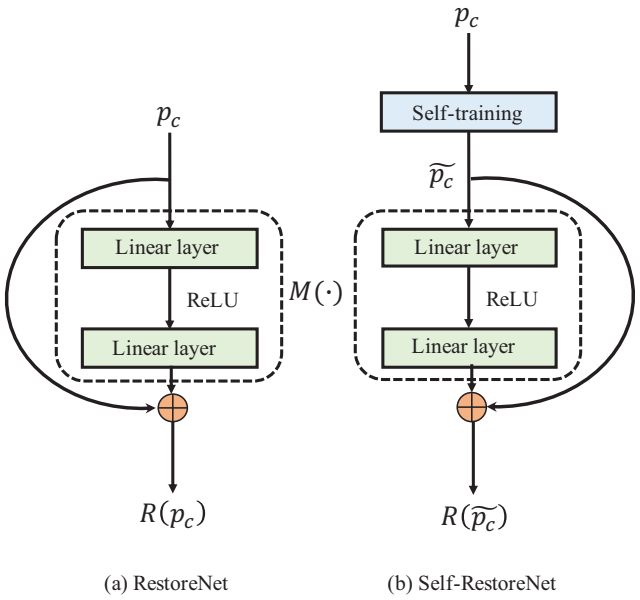
Secondly, we train a regression model to restore the prototypes. The regression model is a MLP network, whose output is denoted as $M(\bar{\boldsymbol{x}})$ where $\bar{\boldsymbol{x}} = f(\boldsymbol{x})$. The loss function is the squared Euclidean distance between $M(\bar{\boldsymbol{x}})$ and the true prototype of each class. The training data is collected as follows. For each class $c \in C_{base}$, we generate its prototype according to Equation 1 where S_c includes all images from class c in D_{base}. In this way, the prototype, denoted by $\boldsymbol{t}_c$, is considered to be discriminative for the class and is thus used as the target, i.e., the truth prototype, to train $M(\cdot)$. Next, for each target prototype $\boldsymbol{t}_c$, we select its λ farthest images from class c, which are considered as noisy and non-discriminative examples of class c. Each of the selected noisy images and its target prototype constitutes a training pair.

Lastly, during inference, we average $M(\boldsymbol{p_c})$ and $\boldsymbol{p_c}$ as the final prototype $R(\boldsymbol{p_c})$ following Equation 3. The structure looks like that in ResNet. We adopt this skip-connection structure for 1) ensemble modelling; 2) data augmentation by considering $M(\boldsymbol{p_c})$ as a new image; 3) avoid mistransformation by the regression network. We compute the distance between $R(f(\boldsymbol{x}))$ and $f(\boldsymbol{x'})$ directly for nearest neighbor classification, where $\boldsymbol{x'}$ is the query image and $\boldsymbol{x}$ is the image from the support set S_{novel}. The workflow for *RestoreNet* is depicted in Figure 3(a).

$$R(\boldsymbol{p_c}) = \frac{1}{2}M(\boldsymbol{p_c}) + \frac{1}{2}\boldsymbol{p_c} \tag{3}$$

3.3 Self-Training

In real-life, the query images are usually processed in batch to improve the system throughput(Wang et al. 2018a). To further improve the prototype, we try to exploit the query images in the query set via self-training (Ren et al. 2018; Chen et al. 2019). When classifying an image, we use all other images in the query set to constitute an unlabeled set U. For each initial prototype, we retrieve its γ nearest images from U and add them into the support set to refine the prototype (Equation 1). We denote the refined prototype as $\widetilde{\boldsymbol{p_c}}$. All procedures described above just happen in inference without retraining. We depict the workflow in Figure 3(b). Note that this self-training step is optional in our model, denoted as *Self+RestoreNet*. We adopt this scheme to show that *RestoreNet* can be easily combined with other methods to realize further performance improvement.



(a) RestoreNet (b) Self-RestoreNet

Figure 3: Workflow for RestoreNet and Self-RestoreNet

4 Experiments

4.1 Datasets

We evaluated our model on multiple benchmark datasets for one-shot image classification. They are *mini*ImageNet, CIFAR-100, Caltech-256 and Caltech-UCSD Birds-200-2011 (CUB-200). These datasets span a large variety of properties and can simulate various application scenarios. We adopt the same data splits with previous works (Schwartz et al. 2018; Chen et al. 2018; Zhou, Wu, and Li 2018; Ravi and Larochelle 2016). More details about those datasets are summarized in Table 1.

4.2 Implementation details

In order to make a fair comparison with the state-of-the-art algorithms, we follow (Chen et al. 2018; 2019; Mishra et al. 2017; Zhou, Wu, and Li 2018; Schwartz et al. 2018) and adopt ResNet18 (He et al. 2016) as our feature extractor, which outputs a 512-dimensional vector as the feature for each image. As described in Section 3.2, during training we add a label classifier branch to the Prototypical Network. This additional branch is implemented via a MLP whose hidden layer has 256 units and output layer has $|C_{base}|$ units. ReLU is used as the activation layer. The two loss terms are weighted summed as the total loss for the whole network.

Note that the additional image classifier will be discarded after training. We use 30-way 1-shot episodes with 10 query images per category to train the network. In each epoch, 600 such episodes are randomly sampled. The learned feature extractor is applied in all subsequent experiments. We optimize the networks via Adam with a initial learning rate 10^{-3}, annealed by half for every 20 epochs.

RestoreNet is trained after we obtain the feature extractor. We use a two-layer MLP for $M(\cdot)$, where the hidden layer has 256 units. ReLU is chosen as the activation function for the hidden layer. The output layer has the same number of units as the input. We tune λ, which is the number of selected noisy images per class, on the validation dataset. It is 100, 30, 5 ,1 for *mini*ImageNet, CIFAR-100, Caltech-256 and CUB-200 respectively. Adam with a fixed learning rate 10^{-3} is used to train the regression netowrk, i.e., $M(\cdot)$. Training *RestoreNet* is fast, which takes only tens of seconds. We report the average performance on 10,000 randomly generated episodes from the test split. Each episode contains 30 query images per category.

For Self-training scheme, as described in Section 3.3, we take the whole query set (exclude current image to be classified) as the unlabeled set U in experiments on CIFAR-100, Caltech-256 and CUB-200. While in experiments on *mini*ImageNet, we adopt an alternative implementation method following (Chen et al. 2019) for fair comparison with it. Instead of applying self-training over the query set, we supply another unlabeled images set U_{novel} to each episode as the unlabeled set U. For an episode, its supplied unlabeled images set U_{novel} has the same samples distribution, i.e., number of examples per class, with the query set. We stipulate that $\gamma = 4$, i.e., each initial prototype retrieves its 4 nearest images from unlabeled set U to do self-training.

4.3 Results and discussion

We report the performance of our model on miniImageNet in Table 2. It can be found that our proposed method outperforms the state-of-the-art methods in one-shot learning. For fair comparison, we reimplement Prototypical Network(Snell, Swersky, and Zemel 2017) by replacing its feature extractor with ResNet18(He et al. 2016). Existing papers have done the experiments of Matching Network(Vinyals et al. 2016), Relation Network(Sung et al. 2018) and MAML(Finn, Abbeel, and Levine 2017) using ResNet18 as the feature extractor. Therefore we directly take the corresponding results from the paper (Chen et al. 2018). Note that some approaches in Table 2 require additional data, e.g. DEML+Meta-SGD(Li et al. 2017) needs an external large-scale dataset *ImageNet-200*(Li et al. 2017); Dual TriNet(Chen et al. 2018) uses word embeddings or human-annotated class attributes for data augmentation; Delta-encoder(Schwartz et al. 2018) needs to be exposed to large number of samples to extract intra-class deformations; Self-training based methods like Self-Jig(Chen et al. 2019) and Self-RestoreNet also require access to external samples. Our method, RestoreNet (not Self-RestoreNet), can work without any restriction. What is more, compared with other models, ours is significantly simpler in structure.

RestoreNet is also able to achieve state-of-the-art or even superior performance on the other three datasets, CIFAR-100, Caltech-256 and CUB-200, as presented in Table 3. We infer that our approach can take effects under different tasks and scenarios.

Table 2: The 5-way, 1-shot classification results(%) on *mini*ImageNet. The "±" indicates 95% confidence intervals over tasks. The "±" is not reported in Delta-encoder.

Models	1-shot Acc.
Meta-LSTM (Ravi and Larochelle 2016)	43.44±0.77
Meta-Net (Munkhdalai and Yu 2017)	49.21±0.96
Matching Nets (Vinyals et al. 2016)	43.56±0.84
(Deep)	47.89±0.86
ProtoNets (Snell, Swersky, and Zemel 2017)	49.42±0.78
(Deep)	56.35±0.77
Relation Nets (Sung et al. 2018)	50.44±0.82
(Deep)	57.02±0.92
MAML (Finn, Abbeel, and Levine 2017)	48.70±1.84
(Deep)	52.23±1.24
Meta-SGD (Li et al. 2017)	50.47±1.87
(Deep)	52.31±1.14
SNAIL (Mishra et al. 2017)	55.71±0.99
DEML+Meta-SGD (Zhou, Wu, and Li 2018)	58.49±0.91
Dual TriNet (Chen et al. 2018)	58.12±1.37
Delta-encoder (Schwartz et al. 2018)	**59.90**
Self-Jig (Chen et al. 2019)	58.80±1.36
RestroreNet (Ours)	**59.28±0.20**
Self-RestroreNet (Ours)	**61.14±0.22**

4.4 Ablation study

Does *RestoreNet* work consistently? To find out if *RestoreNet* takes effect across different tasks and how much enhancement it is able to achieve, we strictly control variables and conduct a series of N-way one-shot experiments on *mini*ImageNet. The results averaged over 600 randomly generated test episodes are presented in Table 4. We can find that *RestoreNet* achieves obvious and consistent improvement on all these tasks. Baseline and RestoreNet use $\boldsymbol{p_c}$ and $R(\boldsymbol{p_c})$ as the prototypes respectively (Figure 3(a)).

In order to explore whether *RestoreNet* is able to achieve further performance improvement when combined with other state-of-the-art algorithms, we adopt the Self-training scheme and test *RestoreNet* on it. As shown in Table 5, Self-training is an effective algorithm which can significantly boost the performance of the baseline by more than two percent, reaching a challenging results at 59.78%. Despite this, *RestoreNet* still improves the performance of the model by a obvious margin, around 1.36 percent.

How to configure λ During the training of *RestoreNet*, only images with serious noise in each class are fed to the network. We select the training images in this way because RestoreNet is proposed to correct the prototype generated from the noisy image. If we use all available images in D_{base}, then both noisy images and normal images are included in the training data, which would confuse the model on how to restore the prototype. In other words, it would pose difficulty on the training (optimization) process. Instead, if we

Table 3: The 5-way, 1-shot classification results(%) on CIFAR-100, Caltech-256 and CUB-200. The "±" indicates 95% confidence intervals over tasks. Note that "±" is not reported in some previous works. Average performances on 600 randomly generated episodes are reported.

Models	CIFAR-100	Caltech-256	CUB-200
Matching Nets (Vinyals et al. 2016)	50.53±0.87	48.09±0.83	49.34
MAML (Finn, Abbeel, and Levine 2017)	49.28±0.90	45.59±0.77	38.43
DEML+Meta-SGD (Zhou, Wu, and Li 2018)	61.62±1.01	62.25±1.00	66.95±1.06
Dual TriNet (Chen et al. 2018)	63.41±0.64	63.77±0.62	69.61±0.46
Delta-encoder (Schwartz et al. 2018)	66.7	**73.2**	69.8
RestroreNet (Ours)	**66.87±0.94**	64.10±0.89	**74.32±0.91**
Self+RestroreNet (Ours)	**69.09±0.97**	**68.28±0.96**	**76.85±0.95**

Table 4: N-way one-shot tasks results(%) with the enhancements of RestoreNet on *mini*ImageNet. The "±" indicates 95% confidence intervals over tasks. Average performances on 600 randomly generated episodes are reported.

Models	5-way	7-way	9-way	11-way	13-way	15-way	20-way
Baseline	57.67±0.83	49.11±0.68	43.38±0.54	38.92±0.44	35.29±0.40	32.48±0.35	27.69±0.27
RestroreNet	59.56±0.84	50.55±0.68	44.54±0.55	39.98±0.43	36.34±0.39	33.52±0.35	28.48±0.27
Enhancement	1.89	1.44	1.16	1.06	1.05	1.04	0.79

Table 5: Five-way one-shot classification accuracy(%) on miniImageNet. Results presented are for $\boldsymbol{p_c}$, $\widetilde{\boldsymbol{p_c}}$ and $R(\widetilde{\boldsymbol{p_c}})$ respectively (shown in Figure 3(b)). The "±" indicates 95% confidence intervals over tasks.

Models	5-way 1-shot Acc.
DEML+Prototypical Nets (Baseline)	57.63±0.20
Self training + Baseline	59.78±0.22
Self+RestroreNet	61.14±0.22
Enhancement	1.36

Table 6: *mini*ImageNet five-way one-shot classification results(%) for the **simplest** *RestoreNet*(512-1-512). The "±" indicates 95% confidence intervals over tasks.

	5-way 1-shot Acc.
Baseline	57.67±0.83
RestoreNet	58.98±0.83
Enhancement	1.31
Self+Baseline	60.37±0.90
Self+RestoreNet	61.64±0.90
Enhancement	1.27

only select the λ farthest images, we are likely to exclude those normal images.

To verify our assumption above, we conduct experiments on *mini*ImageNet. We gradually increase λ, which is the number of noisy images collected from each category, from 100 to 600 (all images belonging to the class) and train the corresponding $M(\cdot)$. We report the 5-way 1-shot results together with the enhancements in Table 7. It can be found that *RestoreNet* realize the best performance when λ equals to 100. As λ increases, in general, the enhancement effects of *RestoreNet* drops. And when λ equal to 600, which is the case trying to learn an universal model, it gives us the worst performance.

How simple can *RestoreNet* be? Our training strategy that only feed the network with noisy images significantly reduce the difficulty of learning for *RestoreNet*. It is for this reason that *RestoreNet* is able to achieve good performance with a simple structure. How simple can *RestoreNet* be while keeping the enhancement? To answer this question, we reimplement the regression network for $M(\cdot)$ as the simplest network which is able to handle a 512-dimension to 512-dimension regression task. This model is just a two-layer MLP with the single hidden layer of only one units. λ is set as 100 here. The performance of this simplest *RestoreNet* on *mini*ImageNet is presented in Table 6. From the results, we can still observe enhancements, although they are slightly weaker than that of a more complex structure.

4.5 Visualization

To understand how *RestoreNet* works, we select a class, Golden Retriever, from the test split of *mini*ImageNet and visualize samples in this category by applying t-SNE(Maaten and Hinton 2008) to their feature vectors. As shown in Figure 4, the red star marks the "true" prototype (calculated by averaging feature vectors of all 600 images) of this class while the violet triangle represents the initial prototype. The restored prototype is marked by the green triangle. It can be found that the initial prototype (violet triangle) is moved much closer to the "true" prototype by the *RestoreNet*.

We further visualize a two-way one-shot classification task which is sampled from *mini*ImageNet. The two classes to be classified are Golden Retriever (blue) and Alaskan Malamute (pink). As shown in Figure 5, violet triangles mark samples in the support set (one image per class) while points (blue and pink) represent images in the query set. After applying *RestoreNet*, violet triangles which represent the initial prototypes are pushed to their corresponding green

Table 7: *mini*ImageNet five-way one-shot classification results(%) for *RestoreNets* trained under different λ, which is the number of noisy samples collected from each category. Average performances on the same 10,000 randomly generated episodes are reported. The "±" indicates 95% confidence intervals over tasks.

λ	100	200	300	400	500	600
Baseline	57.63±0.20	57.63±0.20	57.63±0.20	57.63±0.20	57.63±0.20	57.63±0.20
RestroreNet	59.28±0.20	58.93±0.20	59.12±0.20	58.84±0.20	58.55±0.20	58.21±0.20
Enhancement	**1.65**	1.30	1.49	1.21	0.92	0.58
Self+Baseline	59.78±0.22	59.78±0.22	59.78±0.22	59.78±0.22	59.78±0.22	59.78±0.22
Self+RestroreNet	61.14±0.22	61.08±0.22	60.68±0.22	60.90±0.22	60.41±0.22	60.18±0.22
Enhancement	**1.36**	1.30	0.90	1.12	0.63	0.40

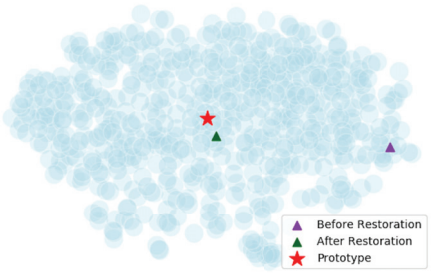


Figure 4: Visualization for images of a class, Golden Retriever, in *mini*ImageNet. The red star shows the target prototype while triangles mark a proposed prototype(learned from a one-shot example) before(violet) and after(green) restoration. *RestoreNet* manages to move the initial proposed prototype much more close to its destination. The figure is plotted by applying t-SNE to ResNet18 features of images. Best viewed in color

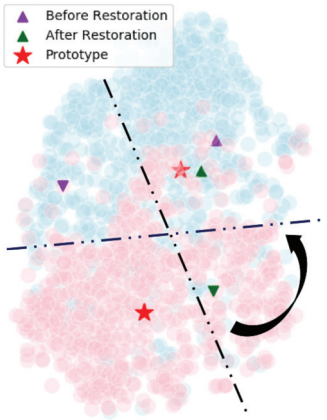


Figure 5: Visualization of a two-way one-shot classification task sampled from *mini*ImageNet. After restoration, violet triangles which represent initial proposed prototypes are pushed to their corresponding green triangles. The green triangles are significantly closer to the target prototypes(marked by the red stars) compared to the violet. The figure is plotted by applying t-SNE to ResNet18 features of images. Best viewed in color

Table 8: Distances between samples and their corresponding class centers, averaged among all samples in the test set.

Average dist. (test set)	p	$M(p)$	$R(p)$
*mini*ImageNet	1.7981	0.9386	1.0822
CIFAR-100	1.7791	1.3790	1.1305
Caltech-256	1.5930	1.2339	1.0101
CUB-200	1.5142	1.3592	1.0403

triangles. It can be found that the green triangles are closer to the target true prototypes (marked by the red stars) compared to the violet. And if we plot the perpendicular bisector of two triangles in the same color, it will give us a decision boundary of the two class. It is obvious that decision boundary produced by the green triangle pair is much better than that produced by the violet ones.

4.6 Statistical analysis

To find out if *RestoreNet* works as expected, i.e., moving feature vectors of samples more close to their clusters' centers, we calculate the average distances between the learned prototype (from each test image) and the corresponding class center. We compare three types of prototypes, namely the output from the feature extraction network (p), the direct output from *RestoreNet* $M(p)$ and the output after applying skip-connection $R(p)$. Equation 3 illustrates the relation among p, $M(p)$ and $R(p)$. As shown in Table 8, the statistics demonstrate that *RestoreNet* takes effect and the average distances using $M(p)$ and $R(p)$ are reduced compared with the distance calculated using p.

5 Conclusion

For one-shot learning, if the training image is far away from the class center, then metric-learning-based approaches would fail to find a good decision boundary in the feature space. In this work, we propose a simple yet effective model (*RestoreNet*) to move the the class prototype constructed from the (noisy) image closer to the class center. Experiments demonstrate that our model obtains superior performance over the state-of-art approaches on a broad range of tasks. In addition, we conduct ablation study and visualize the prototypes to verify the effects of *RestoreNet*.

Acknowledgements

This work is supported by the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity RD Programme (No. NRF2016NCR-NCR002-020), and FY2017 SUG Grant. We would like to thank all anonymous reviewers for their valuable comments.

References

Chen, Z.; Fu, Y.; Zhang, Y.; Jiang, Y.-G.; Xue, X.; and Sigal, L. 2018. Semantic feature augmentation in few-shot learning. *arXiv preprint arXiv:1804.05298*.

Chen, Z.; Fu, Y.; Chen, K.; and Jiang, Y.-G. 2019. Image block augmentation for one-shot learning. In *AAAI*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Hariharan, B., and Girshick, R. 2017. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*.

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

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9:2579–2605.

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2017. A simple neural attentive meta-learner. *ICLR*.

Munkhdalai, T., and Yu, H. 2017. Meta networks. In *ICML*.

Ravi, S., and Larochelle, H. 2016. Optimization as a model for few-shot learning. In *ICLR*.

Ren, M.; Triantafillou, E.; Ravi, S.; Snell, J.; Swersky, K.; Tenenbaum, J. B.; Larochelle, H.; and Zemel, R. S. 2018. Meta-learning for semi-supervised few-shot classification. In *ICLR*.

Rosenberg, C.; Hebert, M.; and Schneiderman, H. 2005. Semi-supervised self-training of object detection models.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-learning with memory-augmented neural networks. In *ICML*.

Schwartz, E.; Karlinsky, L.; Shtok, J.; Harary, S.; Marder, M.; Kumar, A.; Feris, R.; Giryes, R.; and Bronstein, A. 2018. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NIPS*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NIPS*.

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

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NIPS*.

Wang, Y.-X., and Hebert, M. 2016. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*.

Wang, W.; Gao, J.; Zhang, M.; Wang, S.; Chen, G.; Ng, T. K.; Ooi, B. C.; Shao, J.; and Reyad, M. 2018a. Rafiki: machine learning as an analytics service system. *Proceedings of the VLDB Endowment* 12(2):128–140.

Wang, Y.-X.; Girshick, R.; Hebert, M.; and Hariharan, B. 2018b. Low-shot learning from imaginary data. In *CVPR*.

Wang, Y.-X.; Ramanan, D.; and Hebert, M. 2017. Learning to model the tail. In *NIPS*.

Zhou, F.; Wu, B.; and Li, Z. 2018. Deep meta-learning: Learning to learn in the concept space. *arXiv preprint arXiv:1802.03596*.