One-Shot Image Learning Using Test-Time Augmentation

Keiichi Yamada and Susumu Matsumi

December 7, 2021

One-Shot Image Learning Using Test-Time Augmentation

Keiichi Yamada and Susumu Matsumi

Department of Information Engineering
Meijo University, Aichi, Japan
yamadak@meijo-u.ac.jp

Abstract. Modern image recognition systems require a large amount of training data. In contrast, humans can learn the concept of new classes from only one or a few image examples. A machine learning problem with only a few training samples is called few-shot learning and is a key challenge in the image recognition field. In this paper, we address one-shot learning, which is a type of few-shot learning in which there is one training sample per class. We propose a one-shot learning method based on metric learning that is characterized by data augmentation of a test target along with the training samples. Experimental results demonstrate that expanding both training samples and test target is effective in terms of improving accuracy. On a benchmark dataset, the accuracy improvement by the proposed method is 2.55 percent points, while the improvement by usual data augmentation which expands the training samples is 1.31 percent points. Although the proposed method is very simple, it achieves accuracy that is comparable or superior to some of existing methods.

Keywords: Image classification · One-shot learning · Test-time augmentation.

1 Introduction

Image recognition performance has significantly improved with the development of deep learning technology. Achieving high performance with modern image recognition systems requires a large amount of labeled data for training. However, collecting and annotating data generally incurs enormous effort and cost. In addition, it is sometimes practically difficult to provide large amounts of training data. In contrast, humans can learn the concept of new classes from one or a few image examples and recognize the images of those classes. The type of machine learning problem in which there are only a few training samples is called few-shot learning (FSL).

FSL exhibits low performance when using common supervised learning methods because the models overfit to a small number of training samples. Therefore, FSL generally uses prior knowledge from meta-training data.

This paper addresses image classification problem, particularly one-shot learning (OSL), which is a type of FSL in which the training data (referred to as the support data in FSL) include only one sample per class. This paper proposes an approach of OSL which is characterized by data augmentation of a test target along with one training sample per class. We reveal the properties of this method and demonstrate its performance through experiments on the miniImageNet and tieredImageNet datasets.

2 Related Work

Basic FSL approaches include metric learning [28, 25, 15, 10], meta-learning [18, 5, 12, 21], and data augmentation [6, 16, 22, 19, 2]. These approaches are occasionally combined.

In the metric-learning-based approach, an embedding space in which samples of the same class are close together is learned from meta-training data. There are various methods depending on how the test target is classified using the support data in the embedding space. Examples include Matching networks [28] and Prototypical networks [25]. Note that the proposed method is based on metric-learning.

Data augmentation is widely used in image recognition including deep learning [23]. The standard image data augmentation techniques include flipping, shifting, rotating, rescaling, and random cropping of the training images. Data augmentation is also used in FSL to address the shortage of training samples [6, 16, 22, 19, 2]. However, data augmentation in previous FSL studies expands the support data, whereas the proposed method expands both the support data and the test target simultaneously.

Test-time augmentation is a method that expands a test target and uses the averages of the output predictions as the final prediction results. This method is often used in general image recognition [8, 26, 7]. We apply a type of test-time augmentation to OSL. To the best of our knowledge, research focusing on the use of test-time augmentation for OSL has not been reported to date.

3 Problem Definition

Three datasets are used for training and testing, including meta-training, in FSL. They are the base set (D_b), which contains the meta-training data for obtaining prior knowledge; the support set (D_s), which contains the support data consisting of a small number of samples for learning novel classes; and the testing set (D_q), which contains the test targets for testing the FSL results. The classes in datasets D_s and D_q are mutual and are both C_{novel}. The class that D_b has is C_{base}, and there are no mutual classes between D_b and D_s. In other words,

$$D_b = \{(x_i, y_i)\}_{i=1}^{N_b},\ y_i \in C_{\text{base}} \tag{1}$$

$$D_s = \{(x_i, y_i)\}_{i=1}^{N_s},\ y_i \in C_{\text{novel}} \tag{2}$$

$$D_q = \{(x_i, y_i)\}_{i=1}^{N_q},\ y_i \in C_{\text{novel}} \tag{3}$$

$$C_{\text{base}} \cap C_{\text{novel}} = \phi, \tag{4}$$

where the label of data x_i is expressed as y_i. Here, x_i is an image.

The FSL problem where the number of novel classes $|C_{\text{novel}}|$ is N and D_s has K labeled samples for each class is referred to as N-way K-shot learning. K is generally 1 to 5. This paper addresses OSL, a type of FSL in which $K = 1$. In the case of OSL, $Ns = N$. The objective of FSL is to obtain a model f that predicts the label y_q of a test target x_q in D_q when D_b and D_s are given, which is expressed as follows.

$$\hat{y_q} = f(x_q; D_b, D_s),\ (x_q, y_q) \in D_q. \tag{5}$$

4 Proposed Method

Fig. 1 shows an overview of the proposed method for a 5-way OSL case, where x_1–x_5 are the support data, and x_q is the test target whose correct class is the class of x_1. An example in an actual image is shown in Fig. 3.

The learning and prediction of the proposed method are based on metric learning. Here, we employ a convolutional neural network (CNN) as an embedded function $g(x; \theta)$ with θ as parameter. Metric learning is performed by training the CNN with D_b as the supervised training data, and parameter θ of the embedded function is obtained. Then, the class y_q of x_q is predicted with classifier h using D_s as the supervised training data in this embedded space.

$$\hat{y_q} = h\Big(g(x_q; \theta(D_b)); \big\{(g(x_i; \theta(D_b)), y_i)\big\}_{i=1}^{N_s}\Big),$$
$$(x_q, y_q) \in D_q,\ (x_i, y_i) \in D_s. \tag{6}$$

We employ the nearest neighbor classifier with the cosine distance as h.

The proposed method expands both the support data $x_i, i = 1, ..., N_s$ in D_s and the test target x_q. The transformation function to expand an image x J times is given as $a_j(x), j = 1, ..., J$. Then, the average vector in the embedded space g of the data where x_i in D_s is expanded J times is expressed as $\text{mean}\{g(a_j(x_i))\}_{j=1}^{J}$. In addition, the average vector in the embedded space g of the data where the test target x_q is expanded J times is expressed as $\text{mean}\{g(a_j(x_q))\}_{j=1}^{J}$. In the proposed method, when a set of $\text{mean}\{g(a_j(x_i))\}_{j=1}^{J}$ with its teacher label y_i, $i = 1, ..., N_s$, is given, the class of x_q is predicted by determining the class of $\text{mean}\{g(a_j(x_q))\}_{j=1}^{J}$ using classifier h. In other words, we obtain the following:

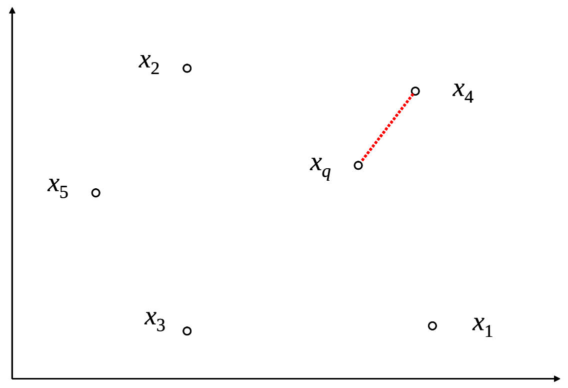
$$\hat{y_q} = h\Big(\text{mean}\{g(a_j(x_q))\}_{j=1}^{J}; \big\{\big(\text{mean}\{g(a_j(x_i))\}_{j=1}^{J}, y_i\big)\big\}_{i=1}^{N_s}\Big),$$
$$(x_q, y_q) \in D_q,\ (x_i, y_i) \in D_s, \tag{7}$$

where $\theta(D_b)$ in g is omitted to keep the formula simple.

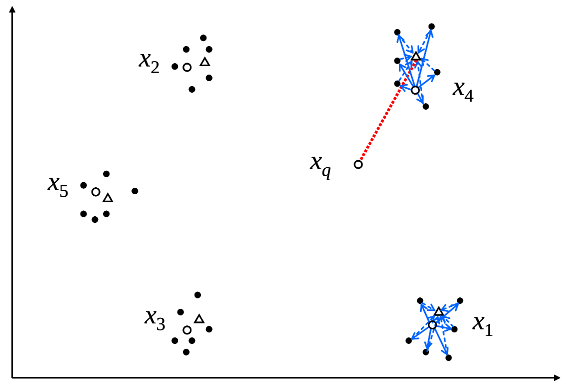
5 Experimental Methods

5.1 Datasets

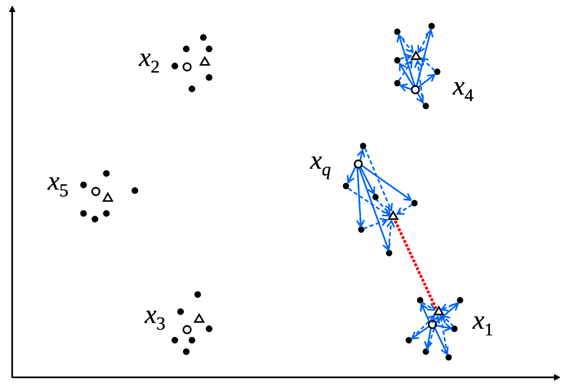
The miniImageNet [28] and tieredImageNet [19] datasets, which are commonly used as benchmarks for FSL, were used in our experiment.



(a) without augmentation



(b) with support data augmentation



(c) proposed method

Fig. 1. Illustration of the proposed method, showing an example of 5-way OSL where x_1–x_5 are the support data and x_q is the test target whose correct class is the x_1 class. The circle represents the embedded vector of each original image. The black dots represent the embedded vectors of the expanded images, and the triangle mark represents the average vector. An example in an actual image is shown in Fig. 3.

The minilmageNet dataset, which is a subset of ImageNet [3], is a 100-class dataset of 600 labeled images per class. Here, the image resolution was 84×84 pixels. According to [18], 64 classes were used as the base class, 16 classes were used as the validation class, and 20 classes were used as the novel class.

The tieredImageNet dataset is also a subset of ImageNet; however, it is a larger dataset than miniImageNet. The classes in the tieredImageNet are grouped according to the hierarchical structure of WordNet. Each class includes an average of 1,282 images. According to [19], 20 superclasses (351 classes) were used as the base class, 6 superclasses (97 classes) were used as the validation class, and 8 superclasses (160 classes) were used as the novel class. The images were resized to 84×84 pixels.

5.2 Backbone

Conv-4 and ResNet-18 were used as the backbone of the embedded function.

Conv-4 is a CNN with four Conv blocks formed from a convolution layer with 3×3 filters, batch normalization, the ReLU activation function, and a 2×2 max pooling layer. In our experiment, the number of filters for the four Conv blocks was 64-64-64-64. Three fully connected layers were connected after the fourth Conv block. The first two fully connected layers had 512 units with the ReLU activation function. Dropout with a rate of 0.5 was applied between these fully connected layers. The export from the first fully connected layer (fc1) was used as a 512-dimension embedded vector. The results of our preliminary experiments indicated that the accuracy of OSL was higher when using the export from fc1 as the embedded vector than when using the export from the fourth Conv block (maxpool4), as shown in Table 1.

The structure of our ResNet-18 is essentially the same as the 18-layer residual network described in [7]. However, the first 7×7 convolution was changed to 3×3 convolution. In addition, by removing the 3×3 max pooling and changing the stride of the first residual block convolution to 1, the first two down-sampling processes were eliminated. In addition, after the global average pooling (GAP), we added one fully connected layer of 512 units with the ReLU activation function. The export from this fully connected layer was used as a 512-dimension embedded vector. The results of our preliminary experiments demonstrated that the accuracy of OSL was higher when using the export from the added fully connected layer (added fc1) as the embedded vector than when using the GAP export, as in Table 1.

5.3 Training Methods

Conv-4 and ResNet-18 were both trained from scratch on D_b using mini-batch stochastic gradient descent with weight decay and momentum using cross-entropy loss. Here, the batch size, weight decay, and momentum were 25, 5×10^{-4}, and 0.9, respectively. The learning rate of Conv-4 was 10^{-3} for the first 60 epochs, followed by 10^{-4} for 10 epochs and 10^{-5} for 10 epochs. The learning rate of ResNet-18 was 3×10^{-2} for 60 epochs, followed by 10^{-3} for 10 epochs and 10^{-4}

Table 1. 5-way and 10-way OSL accuracy and 95% confidence interval on miniImageNet for different embeddings. For Conv-4, using the export from the first fully connected layer (fc1) for the embedded vector has higher accuracy than export from the fourth Conv block (maxpool4). For ResNet-18, using the export from the first fully connected layer added afterward (added fc1) for the embedded vector has higher accuracy than export from the global average pooling (GAP).

Backbone	Embedding	Accuracy (%)	
		5-way	10-way
Conv-4	maxpool4	44.93 ± 0.51	30.69 ± 0.38
	fc1	**50.18** ± 0.57	**34.77** ± 0.44
ResNet-18	GAP	55.90 ± 0.59	39.96 ± 0.48
	added fc1	**57.52** ± 0.60	**41.14** ± 0.50

for 10 epochs for miniImageNet. For tieredImageNet, it was 10^{-2} for 80 epochs, followed by 10^{-3} for 10 epochs and 10^{-4} for 10 epochs. During training of these CNNs, we added general jitter of random crop, rotation, and horizontal flip.

5.4 Experimental Conditions

The transformation function $a_j(x)$ shown in Table 2 was used to expand image data x. Here, the shift(x, d_h, d_v) function shifts x by d_h in the horizontal direction and d_v in the vertical direction, the flip(x) function flips x horizontally, and the rotate(x, d_r) function rotates x by d_r. In our experiment, a_1–a_5 were used when expanding the data 5 times (5×), a_1–a_{10} were used when expanding the data 10×, a_1–a_{18} were used when expanding the data 18×, and a_1–a_{22} were used when expanding the data 22×. Note that Δ and Δ_r were fixed to 5 pixels and 5 degrees, respectively.

We compared the proposed method with the following five scenarios. When expanding the support data D_s but not averaging the embedded vectors, and not expanding the test target x_q,

$$\hat{y_q} = h\Big(g(x_q); \big\{\{(g(a_j(x_i)), y_i)\}_{j=1}^{J}\big\}_{i=1}^{N_s}\Big). \tag{8}$$

When expanding D_s and averaging the embedded vectors, and not expanding x_q,

$$\hat{y_q} = h\Big(g(x_q); \big\{(\text{mean}\{g(a_j(x_i))\}_{j=1}^{J}, y_i)\big\}_{i=1}^{N_s}\Big). \tag{9}$$

When not expanding D_s, and expanding x_q but not averaging the embedded vectors,

$$\hat{y_q} = h\Big(\{g(a_j(x_q))\}_{j=1}^{J}; \{(g(x_i), y_i)\}_{i=1}^{N_s}\Big). \tag{10}$$

When not expanding D_s, and expanding x_q and averaging the embedded vectors,

$$\hat{y_q} = h\Big(\text{mean}\{g(a_j(x_q))\}_{j=1}^{J}; \{(g(x_i), y_i)\}_{i=1}^{N_s}\Big). \tag{11}$$

Table 2. Transformation function $a_j(x)$ to expand image data x; a_1–a_5 are used when expanding the data 5×, a_1–a_{10} are used when expanding the data 10×, a_1–a_{18} are used when expanding the data 18×, and a_1–a_{22} are used when expanding the data 22×. $\varDelta$ is 5 pixels and $\varDelta_r$ is 5 degrees.

$a_1(x) = \text{shift}(x, 0, 0)$	$a_{12}(x) = \text{shift}(x, \varDelta, -\varDelta)$
$a_2(x) = \text{shift}(x, \varDelta, 0)$	$a_{13}(x) = \text{shift}(x, -\varDelta, \varDelta)$
$a_3(x) = \text{shift}(x, -\varDelta, 0)$	$a_{14}(x) = \text{shift}(x, -\varDelta, -\varDelta)$
$a_4(x) = \text{shift}(x, 0, \varDelta)$	$a_{15}(x) = \text{flip}(\text{shift}(x, \varDelta, \varDelta))$
$a_5(x) = \text{shift}(x, 0, -\varDelta)$	$a_{16}(x) = \text{flip}(\text{shift}(x, \varDelta, -\varDelta))$
$a_6(x) = \text{flip}(\text{shift}(x, 0, 0))$	$a_{17}(x) = \text{flip}(\text{shift}(x, -\varDelta, \varDelta))$
$a_7(x) = \text{flip}(\text{shift}(x, \varDelta, 0))$	$a_{18}(x) = \text{flip}(\text{shift}(x, -\varDelta, -\varDelta))$
$a_8(x) = \text{flip}(\text{shift}(x, -\varDelta, 0))$	$a_{19}(x) = \text{rotate}(x, \varDelta_r)$
$a_9(x) = \text{flip}(\text{shift}(x, 0, \varDelta))$	$a_{20}(x) = \text{rotate}(x, -\varDelta_r)$
$a_{10}(x) = \text{flip}(\text{shift}(x, 0, -\varDelta))$	$a_{21}(x) = \text{flip}(\text{rotate}(x, \varDelta_r))$
$a_{11}(x) = \text{shift}(x, \varDelta, \varDelta)$	$a_{22}(x) = \text{flip}(\text{rotate}(x, -\varDelta_r))$

When expanding D_s but not averaging the embedded vectors, and expanding x_q but not averaging the embedded vectors,

$$\hat{y_q} = h\Big(\{g(a_j(x_q))\}_{j=1}^{J}; \big\{\{(g(a_j(x_i)), y_i)\}_{j=1}^{J}\big\}_{i=1}^{N_s}\Big). \tag{12}$$

When there were multiple test elements, as in (10) and (12), the classes were determined based on the total closest distance. The proposed method and these comparison methods were implemented using MatConvNet [27].

6 Experimental Results

Fig. 2 shows main results. This figure presents the 5-way OSL accuracy on miniImageNet for the cases of not expanding the data (1×) and expanding the data by 22× when using Conv-4 as the backbone. For the latter case, the accuracy of the comparison methods (8)–(12) and the proposed method (7) are shown. As the figure shows, the accuracy when expanding both the support data and test target (12)(7) was higher than when expanding only the support data (8)(9) or only the test target (10)(11). The accuracy of the proposed method (7) was 2.55 percent points higher than when not expanding the data (1×), while the accuracy improvement by usual data augmentation which expands only the support data (9) was 1.31 percent points. When comparing using the averaged embedded vector and not using it ((9) vs (8), (11) vs (10), (7) vs (12)), the accuracy of the former was slightly higher. Furthermore, the use of the average vector has the advantage of reducing the search cost of the nearest neighbor classifier h.

Table 3 presents the 5-way and 10-way OSL accuracy on minilmageNet for Conv-4 when the data were expanded 5×, 10×, 18×, and 22×. In this table,

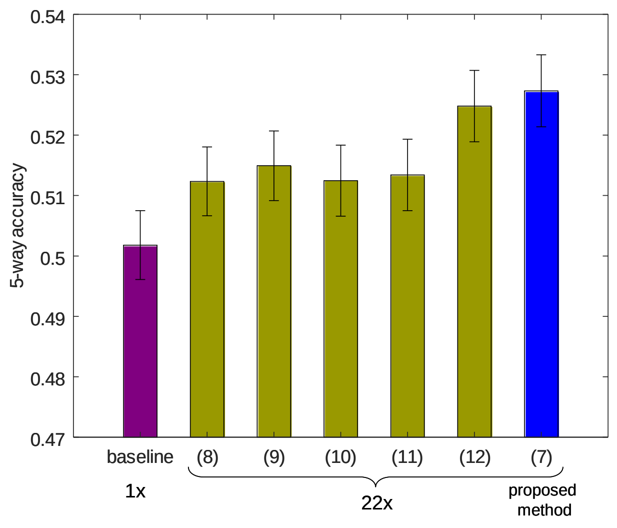


Fig. 2. 5-way OSL accuracy and 95% confidence interval for the cases of not expanding the data (1×) and expanding the data by 22× on miniImageNet for Conv-4. The accuracy of the proposed method (7) which expands both the support data and test target is significantly higher than the cases (8)–(11) which expand either the support data or the test target.

the accuracy of the comparison methods (8)–(12) and the proposed method (7) are shown in order from top to bottom. Here, *exp* and *ave* in the table signify using data expansion and average of the embedded vectors, respectively. As can be seen, for any case, including the comparison methods, the accuracy when expanding the data was higher than when not expanding the data (50.18% for 5-way and 34.77% for 10-way, as displayed in Table 1). Out of these methods, the proposed method (the bottom row of the table) had the highest accuracy. When examining the relationship between expansion rate and accuracy, the larger the expansion rate was, the higher the accuracy was. The rightmost column in the table displays the costs of the nearest neighbor search for each respective case. Here, N and J are the number of the way (i.e., the number of novel classes) and expansion rates, respectively. Since the search cost when no data expansion was applied is N, the proposed method can improve the accuracy without increasing the search cost.

Table 4 presents the accuracy of 5-way OSL on miniImageNet and tieredImageNet for Conv-4 and ResNet-18 when not expanding the data (1×) and when using the proposed method of expanding the data by 22×. For any combination of dataset and backbone, the accuracy improved by 1.3–3.3 percent points with the proposed method of expanding the data by 22× compared to the case of not expanding the data.

Table 3. 5-way and 10-way OSL accuracy (%) on miniImageNet for each expansion rate when using Conv-4 as the backbone. The bottom row is the proposed method. The terms exp and ave signify using data expansion and the embedded vector averaging for the support data D_s and the test target x_q, respectively. The rightmost column presents the calculation costs of the nearest neighbor search.

D_s		x_q		5×		10×		18×		22×		cost
exp	ave	exp	ave	5-way	10-way	5-way	10-way	5-way	10-way	5-way	10-way	
✓	-	-	-	50.83	35.40	51.17	35.65	51.25	35.75	51.23	35.75	NJ
✓	✓	-	-	50.87	35.40	51.24	35.73	51.40	35.87	51.49	35.97	N
-	-	✓	-	50.63	35.18	50.96	35.46	51.10	35.59	51.25	35.69	NJ
-	-	✓	✓	50.78	35.32	51.04	35.54	51.18	35.67	51.34	35.79	N
✓	-	✓	-	51.32	35.85	52.01	36.41	52.29	36.70	52.48	36.89	NJ^2
✓	✓	✓	✓	**51.47**	**35.96**	**52.09**	**36.49**	**52.42**	**36.79**	**52.73**	**37.08**	N

Table 4. 5-way OSL accuracy and 95% confidence interval for no data expansion (1×), and the proposed method when expanding the data 22×.

Dataset	Backbone	5-way Accuracy (%)	
		1×	22×
miniImageNet	Conv-4	50.18 ± 0.57	**52.73** ± 0.60
	ResNet-18	57.52 ± 0.60	**58.84** ± 0.61
tieredImageNet	Conv-4	56.15 ± 0.46	**59.45** ± 0.47
	ResNet-18	65.13 ± 0.50	**67.42** ± 0.50

In Table 5, the proposed method is compared with several existing methods in terms of accuracy. Here, regular FSL which does not use unlabeled data were the comparative target, and semi-supervised FSL [13, 19, 11, 29] that uses external unlabeled data and transductive FSL [13, 17, 4, 20, 29] that uses information from test data other than the test target were excluded. As can be seen, the simple proposed method (Ours) achieved accuracy that is comparable or superior to the methods listed in the table. Although the accuracy of the proposed method was lower than DSN in the miniImageNet+ResNet case, it was higher than that in the miniImageNet+Conv-4, tieredImageNet+Conv-4, and tieredImageNet+ResNet cases.

Fig. 3 shows the results of visualizing the distribution of the embedded vector using t-SNE [14]. The figure shows an example of 5-way OSL when expanding the image data by 22×. Here, x_1–x_5 are the support data, and x_q is the test target. The plot points indicate the embedded vectors of the original image and the expanded images of each class. $\bigcirc$ represents the original image vector, while $\triangle$ represents the average of the expanded image vectors. The figure also shows the original image. In this example, x_q was incorrectly identified as the x_4 class when no data expansion was applied; however, x_q was correctly identified as the x_1 class when using the proposed method. When examining the original image vector $\bigcirc$ and expanded average vector $\triangle$ of x_1, x_4, and x_q, the figure

Table 5. Proposed method (Ours) compared with several existing methods in terms of accuracy.

Dataset	Backbone	Model	5-way Accuracy (%)
miniImageNet	Conv-4	MatchingNet [28]	43.56 ± 0.84
		MAML [5]	48.70 ± 1.84
		ProtoNet [25]	49.42 ± 0.78
		Baseline++ [1]	48.24 ± 0.75
		TapNet [30]	50.68 ± 0.11
		DSN [24]	51.78 ± 0.96
		Support-based init [4]	50.69 ± 0.63
		Ours (22×)	**52.73** ± 0.60
miniImageNet	ResNet-18	MatchingNet [28, 1]	52.91 ± 0.88
	ResNet-18	MAML [5, 1]	49.61 ± 0.92
	ResNet-18	ProtoNet [25, 1]	54.16 ± 0.82
	ResNet-18	Baseline++ [1]	51.87 ± 0.77
	ResNet-12	TapNet [30]	61.65 ± 0.15
	ResNet-12	DSN [24]	**62.64** ± 0.66
	WRN-28-10	Support-based init [4]	56.17 ± 0.64
	ResNet-18	Ours (22×)	58.84 ± 0.61
tieredImageNet	Conv-4	MAML [5, 13]	51.67 ± 1.81
		ProtoNet [25, 13]	53.31 ± 0.89
		MetaOptNet-SVM [9]	54.71 ± 0.67
		TapNet [30]	57.11 ± 0.12
		Support-based init [4]	58.42 ± 0.69
		Ours (22×)	**59.45** ± 0.47
tieredImageNet	ResNet-12	ProtoNet [25, 24]	61.74 ± 0.77
	ResNet-12	MetaOptNet-SVM [9]	65.99 ± 0.72
	ResNet-12	DSN [24]	66.22 ± 0.75
	WRN-28-10	Support-based init [4]	**67.45** ± 0.70
	ResNet-18	**Ours** (22×)	**67.42** ± 0.50

demonstrates that while x_q was close to x_4 in the original image vector, x_q was closer to x_1 in the expanded average vector.

Table 6 shows the results of expanding the data by 18× when transforming image x with a random shift amount rather than expanding the image using transformation functions a_1–a_{18} with a fixed shift amount Δ of 5. This table displays the accuracy of 5-way and 10-way OSL on miniImageNet for Conv-4. For transformation with a random shift amount, two random values Δ_1 and Δ_2 that were sampled per transformation from a uniform distribution of [-10, 10] were used and transformed with shift(x, Δ_1, Δ_2). Here, the probability of horizontal flip was 0.5 (9 out of 18). Table 6 also shows the results of expanding with a_1–a_{18} with the shift amount Δ fixed at a value of 5, which was also shown in Table 3, for the purpose of comparison. Table 6 shows that using a fixed shift amount, as the proposed method, resulted in higher accuracy than using a

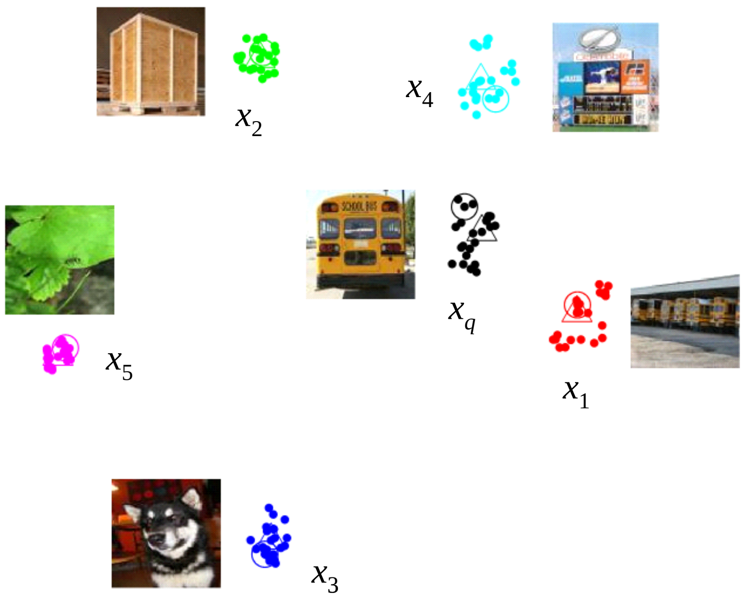


Fig. 3. Example of visualizing the embedded vector distribution using t-SNE when expanding image data by 22×. x_1–x_5 are the support data; x_q is the test target. The plot points indicate the embedded vectors of the original image and the expanded images of each class. ◯ represents the original image vector while △ represents the average of the expanded image vectors.

Table 6. OSL accuracy (%) when transforming with a random shift amount instead of fixing the shift amount when expanding the data 18× on miniImageNet for Conv-4.

D_s		x_q		random Δ		fixed Δ	
exp	ave	exp	ave	5-way	10-way	5-way	10-way
✓	-	-	-	49.42	33.92	51.25	35.75
✓	✓	-	-	49.79	34.22	51.40	35.87
-	-	✓	-	48.92	33.55	51.10	35.59
-	-	✓	✓	49.07	33.71	51.18	35.67
✓	-	✓	-	51.61	36.07	52.29	36.70
✓	✓	✓	✓	52.10	36.45	**52.42**	**36.79**

random shift amount. We believe the reason for this result is that when the shift amount is determined randomly per image, there are differences in the expansion method between images.

7 Conclusion

In this paper, we have proposed an OSL method for image classification that is characterized by the data expansion of a test target along with support data. The experimental results demonstrate that expanding both the support data and test target is effective in terms of improving accuracy. Accuracy can be improved without increasing the cost of nearest neighbor search using the average of the embedded vectors of the expanded images. The proposed method achieved performance that is comparable or superior to some existing methods on the miniImageNet and tieredImageNet datasets despite being a rather simple method. Tasks for future research include learning the transformation function and the parameters from meta-training data, and combining this method with other methods.

References

1. Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C.F., Huang, J.B.: A closer look at few-shot classification. In: International Conference on Learning Representations (2019)
2. Chen, Z., Fu, Y., Wang, Y.X., Ma, L., Liu, W., Hebert, M.: Image deformation meta-networks for one-shot learning. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8672–8681 (2019)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
4. Dhillon, G.S., Chaudhari, P., Ravichandran, A., Soatto, S.: A baseline for few-shot image classification. In: International Conference on Learning Representations (2020)
5. 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. Proceedings of Machine Learning Research, vol. 70, pp. 1126–1135 (2017)
6. Hariharan, B., Girshick, R.: Low-shot visual recognition by shrinking and hallucinating features. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 3037–3046 (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. pp. 1097–1105 (2012)
9. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10649–10657 (2019)
10. Li, W., Wang, L., Xu, J., Huo, J., Gao, Y., Luo, J.: Revisiting local descriptor based image-to-class measure for few-shot learning. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7253–7260 (2019)
11. Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T.S., Schiele, B.: Learning to self-train for semi-supervised few-shot classification. In: Advances in Neural Information Processing Systems. vol. 32 (2019)

12. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning, arXiv preprint, arXiv:1707.09835 (2017)
13. Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S., Yang, Y.: Learning to propagate labels: Transductive propagation network for few-shot learning. In: International Conference on Learning Representations (2019)
14. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
15. Oreshkin, B., Rodríguez López, P., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. In: Advances in Neural Information Processing Systems. vol. 31, pp. 721–731 (2018)
16. Qi, H., Brown, M., Lowe, D.G.: Low-shot learning with imprinted weights. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5822–5830 (2018)
17. Qiao, L., Shi, Y., Li, J., Tian, Y., Huang, T., Wang, Y.: Transductive episodic-wise adaptive metric for few-shot learning. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3602–3611 (2019)
18. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: International Conference on Learning Representations (2017)
19. Ren, M., Ravi, S., Triantafillou, E., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. In: International Conference on Learning Representations (2018)
20. Rodríguez, P., Laradji, I., Drouin, A., Lacoste, A.: Embedding propagation: Smoother manifold for few-shot classification. In: Computer Vision – ECCV 2020. pp. 121–138 (2020)
21. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. In: International Conference on Learning Representations (2019)
22. Schwartz, E., Karlinsky, L., Shtok, J., Harary, S., Marder, M., Kumar, A., Feris, R., Giryes, R., Bronstein, A.: Delta-encoder: an effective sample synthesis method for few-shot object recognition. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
23. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(60), 2196–1115 (2019)
24. Simon, C., Koniusz, P., Nock, R., Harandi, M.: Adaptive subspaces for few-shot learning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4135–4144 (2020)
25. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. vol. 30, pp. 4077–4087 (2017)
26. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015)
27. Vedaldi, A., Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In: Proceedings of the 23rd ACM International Conference on Multimedia. pp. 689–692 (2015)
28. Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., Wierstra, D.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems. vol. 29, pp. 3630–3638 (2016)
29. Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M.: Generalizing from a few examples: A survey on few-shot learning. ACM Comput. Surv. **53**(3) (2020)

30. Yoon, S.W., Seo, J., Moon, J.: TapNet: Neural network augmented with task-adaptive projection for few-shot learning. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7115–7123 (2019)