Task-Adaptive Feature Reweighting for Few Shot Classification

Nan Lai[1,2], Meina Kan[1,3], Shiguang Shan[1,2,3](✉), and Xilin Chen[1,2]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, China
{lainan,kanmeina,sgshan,xlchen}@ict.ac.cn

[2] University of Chinese Academy of Sciences, Beijing, China

[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China

Abstract. Few shot classification remains a quite challenging problem due to lacking data to train an effective classifier. Lately a few works employ the meta learning schema to learn a generalized feature encoder or distance metric, which is directly used for those unseen classes. In these approaches, the feature representation of a class remains the same even in different tasks (In meta learning, a task of few shot classification involves a set of labeled examples (support set) and a set of unlabeled examples (query set) to be classified. The goal is to get a classifier for the classes in the support set.), i.e. the feature encoder cannot adapt to different tasks. As well known, when distinguishing a class from different classes, the most discriminative feature may be different. Following this intuition, this work proposes a task-adaptive feature reweighting strategy within the framework of recently proposed prototypical network [6]. By considering the relationship between classes in a task, our method generates a feature weight for each class to highlight those features that can better distinguish it from the rest ones. As a result, each class has its own specific feature weight, and this weight is adaptively different in different tasks. The proposed method is evaluated on two few shot classification benchmarks, *mini*ImageNet and *tiered*ImageNet. The experiment results show that our method outperforms the state-of-the-art works demonstrating its effectiveness.

Keywords: Few shot classification · Feature reweighting · Meta-learning

1 Introduction

In recent years, deep learning has achieved impressive performance on image classification task [1–4]. Despite the success, it requires a large amount of data to update massive parameters. In contrast, the human visual system can learn a new visual concept quickly from few data. The ability of learning quickly from few data is very important for an artificial visual system, as in practice

C. V. Jawahar et al. (Eds.): ACCV 2018, LNCS 11364, pp. 649–662, 2019.
https://doi.org/10.1007/978-3-030-20870-7_40

labeling data manually is very expensive and training on a large scale dataset is time-consuming. Few shot classification is such a kind of technique that aims to recognize a set of new classes from few examples. The challenge of this problem is how to get an effective classifier from few data and limited class variation.

A straightforward approach is fine-tuning a pre-trained model by using those new classes in a task. However, this would cause overfitting problem, especially when the classes for pre-training have large discrepancy with the new classes [19]. Although regularization methods can be used to alleviate the overfitting problem to some extent, they cannot fully solve it. Lately, a new trend of few shot classification methods arises, i.e. quite a few works apply meta learning to few shot classification. For example, MAML [10] attempts to get a model to be easy to fine-tune by learning a good network initialization such that a small number of update steps with few data could produce good generalization performance on a new task. Matching Network [12] learns a network to get the embeddings of few labeled examples and unlabeled queries, over which a cosine similarity metric is used to predict the label of each query. Prototypical Network [6] learns a generalized distance metric space in which classification can be performed by computing distances with prototype representations of each class. On *mini*ImageNet, these approaches significantly outperform the straightforward fine-tuning methods, e.g. more than 10% improvement [12], making meta learning quite promising for few shot classification. The effectiveness of meta learning may benefit from its ability of learning to learn generalized knowledge across different tasks, which can be transferred to a new task.

Among the meta-learning approaches, prototypical network [6] is a fairly simple and elegant solution, and thus attracts much attention. In this work, we attempt to extend prototypical network [6] to further improve its performance. In this method as well as other existing approaches, the feature representation of a class remains the same even in different tasks. This means that the feature encoder cannot adapt to a specific task using the peculiar discriminant information of the task, e.g. the relationship of classes in this task. As well known, when distinguishing a class from different classes, the most discriminative feature is different, e.g. when distinguishing a dog from a cat the head plays a crucial role, while the neck matters most when distinguishing a dog from a giraffe.

Following above intuition, this work proposes a task-adaptive feature reweighting strategy within the framework of prototypical network [6]. Specifically, our method consists of three components, a feature encoder, a weight generator and a classifier, among which the generator is newly introduced module in this work. By considering the discrepancy between a class and the rest ones, the generator produces a feature weight for each class to highlight those most discriminative features to distinguish it from the rest classes. As a result, when compared with different classes, the same class is equipped with different feature weights to focus on different features for better classification. Hence, this method is more discriminative than prototypical network [6]. From the point of view of meta learning, the weight generator is a meta-learner, which learns to pick out those most discriminative features between a class and the other classes to be compared with. By training the network across different tasks as most meta learning methods do, the weight generator generalizes well to a new task. Experimental results on two commonly used few shot classification benchmarks, *mini*ImageNet and *tiered*ImageNet, show that our proposed method outperforms the state-of-the-art works demonstrating its effectiveness.

2 Related Work

Generative Approaches. In the early stage of few shot classification, most works employ generative models to solve this problem. One of the earliest work [20] develops a Bayesian probabilistic model on the features of previous classes with the premise that a prior from previous classes could be generalized to novel classes. Hierarchical Bayesian Program learning method [16] proposes to use a hierarchical Bayesian model to construct handwritten characters from strokes for digital classification. [11] proposes to compose characters from patches and construct an AND-OR graph using patches to represent each character object. Such powerful generative models perform well on classes with limited intra-class variation such as handwritten characters. However, these models cannot capture the vast variation for unconstrained classes.

Regularization and Data Augmentation Approaches. Regularization technique is a natural way to alleviate the overfitting problem, making fine-tuning a pre-trained model feasible for few shot classification. [8] proposes to group the parameters of a network into clusters and the parameters in the same cluster share the same weight and gradient, which effectively regularizes the dimensionality of parameter search space. Another straightforward method for few shot classification is data augmentation. In addition to regular augmentation techniques, e.g. random crop and rotation, color jittering and PCA jittering, modern generative networks [24,26–29] are proposed to generate realistic images. Despite significant progress, these models suffer from the problem of mode collapse and training instability.

Meta-Learning Approaches. Lately, meta learning becomes a popular way to solve few shot recognition problem. Meta learning learns at two level: (1) learns a meta-learner which extracts knowledge through a slow learning phase across all tasks and (2) then learns rapidly for a target task using the meta-learner. The strategy of training across different tasks makes the learned model (i.e. the meta-learner) well generalize to novel tasks. MAML [10] learns a good initialization of networks across all tasks. On the basis of this initialization, only few updates of parameters are enough to achieve a good performance on a new target task. Meta-SGD [9] learns a meta-learner that can not only give an initialization to a network but also give update direction and learning rate to any differentiable learner such that it can learn effectively from a few examples even in one step. Some other works [6,12,14] learn a distance metric across tasks that can be transferred to the target task. Matching Network [12] learns a network that can extract full context representation for few labeled examples and queries and then uses cosine similarity to predict the label of queries. Prototypical Network [6] builds a prototype for each class, which is computed as the average of samples in that class, and then computes the similarity between a query and each prototype via Euclidean distance. Different from [6,12], Relation Network [14] directly learns a nonlinear similarity metric to compute the similarity of two images.

Memory-Based Approaches. There are some works that use memory-based networks for few shot classification task. Memory networks are networks augmented with an external memory that can be read and written based on location and content. The external memory is used to store knowledge or past experience that is useful for few shot recognition task. In [7], the few shot classification task is formalized as a sequence classification problem. At each time step, an LSTM controller read past experience from external memory and writes current feature to external memory. And the retrieved memory is used to classify current image. Here the external memory is used to hold data samples presented at previous steps. In [5], two memory components are introduced, one called external memory used to store representations of a large set of auxiliary classes, and the other called abstraction memory used to store concise information related to target categories. Read operation is defined on the external memory to retrieve related information from auxiliary classes. Two operations are defined on the abstraction memory, one read operation to retrieve useful memory from the abstraction memory to classify current query and one write operation which updates the abstraction memory using the information of current query and the retrieved memory from external memory. To use memory networks, the key is to design how to handle the memory. And until now how to train a memory network is still a challenging problem.

3 Method

3.1 Problem

Few shot classification aims at getting an effective classifier by only using few examples for each class. A few shot classification task generally involves a support set S containing few labeled examples and a query set Q containing examples to be classified. If the support set contains N unique classes and K examples for each of these classes, the task is called N-way K-shot classification.

Recently meta learning is widely used to few shot classification task. Meta learning, also referred to as learning to learn, endeavors to learn from sparse data rapidly. It learns at two level, a gradual learning process for a meta learner which can guide a rapid learning process for a learner targeting a specific task. Meta learning approaches usually consists of two phases, meta training and meta testing. In meta-training phase, a meta learner is trained across different tasks sampled from a meta dataset. The meta dataset is usually a large labeled dataset containing C classes with many samples for each class, and has no overlap with the support set and query set of the testing task. In each training iteration, the meta learner randomly samples a N-way K-shot task from meta dataset to

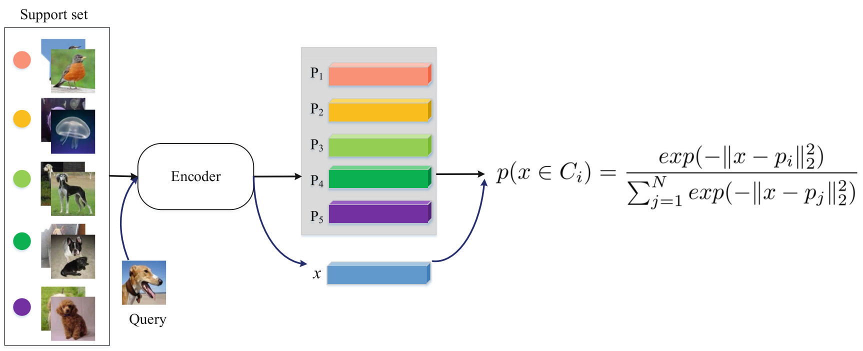
Support set
Encoder
Query
P_1
P_2
P_3
P_4
P_5
x
$p(x \in C_i) = \frac{exp(-\|x - p_i\|_2^2)}{\sum_{j=1}^{N} exp(-\|x - p_j\|_2^2)}$

Fig. 1. Prototypical network [6]. It consists of an encoder used to extract feature and a non-parameterized classifier. The classifier computes a probability distribution over classes of query x based on its distance with the prototypes.

mimic the testing task. The selected support set together with the query set can also be called an episode. The meta learner is optimized by minimizing the classification loss on the query set in this episode. As the training process goes on, different tasks, i.e. different episodes, are sampled and used to train the meta learner. Therefore, the meta learner can learn useful and well generalized knowledge across different tasks. In the phase of meta-testing, the trained meta learner is directly used for a novel classification task.

3.2 Baseline Method: Prototypical Network [6]

Prototypical network is one of the state-of-the-art meta-learning methods for few shot classification. It is used as the baseline of our method considering its simplicity and effectiveness. Prototypical network consists of two modules, an encoder and a non-parameterized classifier, as shown in Fig. 1. The encoder E maps an input image I into a feature representation z in an embedding feature space $\mathcal{Z}$, i.e.,

$$z = E(I) \tag{1}$$

where the encoder E is usually structured as a four-layer convolutional network. In the embedding feature space, a prototype for each class p_i is computed as the mean of the few examples in the support set, formulated as follows:

$$p_i = \frac{1}{K}\sum_{k=1}^{K} z_i^k \tag{2}$$

where z_i^k is the k-th sample of i-th class in the support set. The classifier produces a softmax-like distribution over classes of a query x based on its Euclidean distances with the prototypes:

$$p(y = i|x) = \frac{exp(-d(x, p_i))}{\sum_{j=1}^{N} exp(-d(x, p_j))}, \quad i = 1, 2, \cdots, N. \tag{3}$$

$$d(x, p_i) = \|x - p_i\|_2^2, \quad i = 1, 2, \cdots, N. \tag{4}$$

where $p(y = i|x)$ is the probability of predicting query x as the i-th class. Using the standard cross-entropy loss as supervision signal, the whole network is trained across different tasks from scratch in an end-to-end manner.

In testing phase, the encoder E is directly used to a novel task to extract the feature representation of samples in support set and query set. Finally, the classification of a query sample is simply performed by finding the nearest class prototype under the Euclidean distance metric.

3.3 Our Method: Task-Adaptive Prototypical Network

In prototypical network [6], the learned feature representation is generally discriminative across many tasks, but not finely carved for a specific testing task. As well known, when distinguishing a class from different classes, the most discriminative features are different. For example, when distinguishing snow from hail, the most discriminative feature is its shape, while distinguishing snow and soil the most discriminative feature turns to color. In other words, a feature representation across the board is not optimal, and a carved one for the specific task is preferred. Following this intuition, this work proposes a task-adaptive feature reweighting strategy to extend prototypical network [6] for further improvement. Feature weighting highlights those most discriminative features that can distinguish a class from the rest classes in the same task, not from all other classes. When distinguishing one class from different classes in different tasks, those highlighted features are different. Thus our proposed network is called as Task-adaptive Prototypical Network.

Overall Framework. As Fig. 2 shows, our proposed task-adaptive Prototypical Network consists of three main components, an encoder used to extract features of examples, a feature weight generator used to produce a feature weight for each class and a non-parameterized classifier used for final classification. The feature weight generator is the newly introduced in this work and the rest two parts are the same as prototypical network [6].

In detail, for a N-way K-shot classification task, the images of the support set are fed into the encoder and the encoder outputs a feature representation for each image. Then these features are fed into the weight generator module and the module produces a feature weight for each class, which is used to re-weight the features of this class. Based on the feature weights, the distance between a query sample x and the i-th prototype p_i is computed as below:

$$d(x, p_i) = \|w_i \cdot x - w_i \cdot p_i\|_2^2 \tag{5}$$

where w_i is weight vector, p_i and x are feature vectors and $\cdot$ means dot product operation. The probability of query x belonging to the i-th class is computed as:

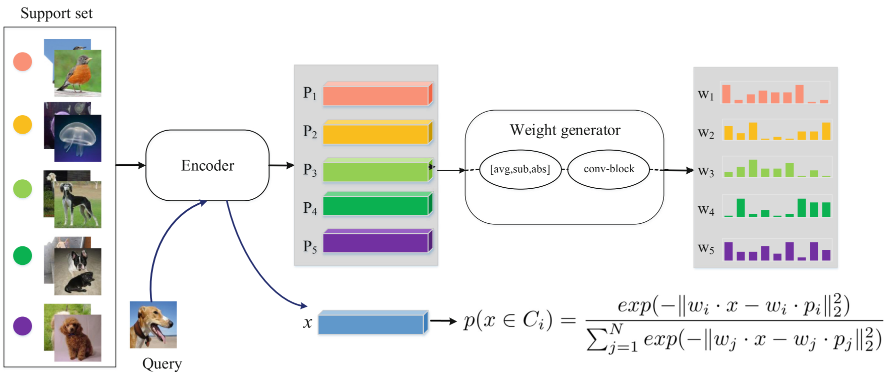
Support set
Encoder
Query
P_1
P_2
P_3
P_4
P_5
Weight generator
[avg,sub,abs]
conv-block
w_1
w_2
w_3
w_4
w_5
x
$p(x \in C_i) = \frac{exp(-\|w_i \cdot x - w_i \cdot p_i\|_2^2)}{\sum_{j=1}^{N} exp(-\|w_j \cdot x - w_j \cdot p_j\|_2^2)}$

Fig. 2. Our task-adaptive prototypical network is an extension of Prototypical Network [6]. It consists of three components, an encoder used to extract feature, a newly introduced weight generator used to produce feature weight for each class and a non-parameterized classifier. The classifier computes a probability distribution over classes of query x based on its distance with the re-weighted prototypes.

$$p(y = i|x) = \frac{exp(-\|w_i \cdot x - w_i \cdot p_i)\|_2^2)}{\sum_{j=1}^{N} exp(-\|w_j \cdot x - w_j \cdot p_j)\|_2^2)} \tag{6}$$

Similar to prototypical network [6], in training phase the parameters of the encoder and weight generator can be easily optimized by using the standard cross-entropy loss as supervision signal in an end-to-end manner. In the testing phase, the encoder and weight generator is directly used to a novel task.

Weight Generator Module. It is obvious that the most discriminative feature for a class would be among those features that are different between this class and the rest ones in the same task. Therefore, the weight generator takes the feature differences between the classes as input and outputs the feature weights.

Specifically, the weight generator is structured as a small sub-network with several convolutional layers, denoted as G. As shown in Fig. 3, given a support set S of a N-class K-shot task, the weight generator computes the feature weight of one class based on the feature differences between its prototype and the rest ones, which can be formalized as:

$$w_i = G\left(\left|p_i - \frac{1}{N-1}\sum_{j \neq i} p_j\right|\right) \tag{7}$$

where w_i is the feature weight vector of the i-th class with each value between 0 and 1 and its dimension is the same as feature dimension.

The generator targets to learn how to compute discriminating power of a feature based on its inter-class variation, i.e. feature differences. Here we adopt convolutional structure to model G. The reasons are two-folded. The first one is that sharing the same kernel in the way of convolution can reduce the parameters. The second one is that the feature weight is only related to feature variations rather than the feature itself, which means that different features can use the

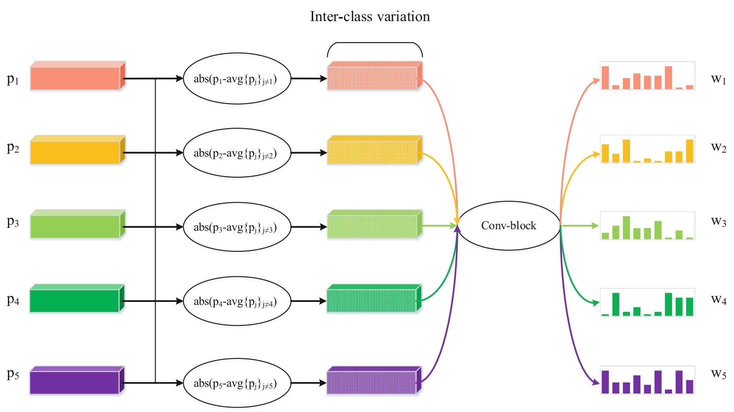

Fig. 3. The weight generator in the 5-way 1-shot classification scenario. For each class, the feature differences between its prototype and the other four prototypes are computed and averaged as inter-class variation. Then the inter-class variation together with its prototype are fed into a convolution block and the feature weight is produced. The weight generator is trained together with the encoder in an end-to-end way.

same parameters to compute their weights. Hence the generator can be naturally implemented as a convolutional structure.

This whole task-adaptive network is trained from scratch using the schema of meta learning. The standard cross-entropy loss function is taken as the supervision signal, and the back propagation and gradient descent algorithm are used to optimize the following objective in an end-to-end manner:

$$\min_{E,G} L = \sum_{I \in Q} -log(p(y = y_I | I)) \tag{8}$$

where I is an image in the query set Q of an episode and y_I is the ground truth label of image I.

4 Experiments

Our proposed method is evaluated by comparing it with the state-of-the-art ones [6,9,10,12,14] on two few shot classification benchmarks, *mini*ImageNet [12] and *tiered*ImageNet [21].

4.1 Experimental Setting

In our experiments, the encoder is composed of four convolutional blocks as prototypical network [6]. The newly introduced weight generator consists of one tiny convolutional block with kernel shared to significantly reduce the number of parameters. A detailed structure is listed in Table 1. The whole network is trained from scratch via Adam with random initialization.

Table 1. The structure of our task-adaptive prototypical network. Here, conv(n × n,c) means a convolutional layer with c channels and n is the kernel size. bn means batch normalization layer and relu is the non-linear ReLU activation layer. max-pool(n × n) is a max-pooling layer with n × n kernel size.

Encoder				Weight generator
block1	block2	block3	block4	

conv(3 × 3,64)
bn
relu
max-pool(2 × 2)
	conv(3 × 3,64)
	bn
	relu
	max-pool(2 × 2)
		conv(3 × 3,64)
		bn
		relu
		max-pool(2 × 2)
			conv(3 × 3,64)
			bn
			relu
			max-pool(2 × 2)
				conv(3 × 3,2)
				relu
				conv(1 × 1,1)

For fair comparison, all methods are evaluated on the same evaluation setting, i.e. 600 episodes are randomly sampled from the test set, with each episode containing 15 query images per class in both 1-shot and 5-shot scenarios. The final classification performance is computed as the average classification accuracy over these 600 episodes.

4.2 Few-Shot Classification on *mini*ImageNet

*mini*ImageNet, firstly proposed by Matching Network [12], is a subset of ILSVRC-12 dataset [22]. This dataset contains 100 classes with 600 images in each class. We use 64 classes as training set, 16 classes as validation set and the remaining 20 classes as test set, same as the compared methods for fair comparison. All the images are resized to 84 × 84. In training process, each episode contains 30 classes and 15 queries for each class on the 1-shot scenario, and 20 classes with 15 queries for each class on the 5-shot scenario.

Table 2 shows the performance of few shot classification on the *mini*ImageNet. From Table 2, we can see that the Prototypical Network [6] performs the best in 5-shot scenario among the existing works, but only ordinarily in the 1-shot scenario. With the proposed feature re-weighting strategy, our method outperforms the Prototypical Network and other existing works in both 1-shot and 5-shot scenarios, especially on the one-shot scenario. Compared to Relation Network [14], our model with half number of parameters gets a nearly 2% higher performance on the 1-shot scenario and a nearly 4% improvement on the 5-shot scenario. The superior performance on this dataset demonstrates the effectiveness of our proposed feature re-weighting strategy. Moreover, the reweighting strategy can be easily integrated into any other framework besides prototypical network [6].

4.3 Few-Shot Classification on *tiered*ImageNet

*tiered*ImageNet is another larger dataset for few shot classification. It is proposed in [21]. Like *mini*ImageNet, it is also a subset of ILSVRC-12 [22]. This dataset contains 34 categories and 608 classes in total, with each category containing between 10 and 30 classes. These categories are split into 20 for training, 6 for

Table 2. Few-shot classification accuracies on *mini*Imagenet. All accuracy results are averaged over 600 test episodes and are reported with 95% confidence intervals. The performances of other methods are copied from their report. 'Y' in column Finetune means the method fine-tunes the model learned at meta-training stage for a test episode. The best-performing method is highlighted.

Model	Finetune	5-way	
		1-shot	5-shot
Matching network [12]	N	43.56% ± 0.84%	55.31% ± 0.73%
MAML [10]	Y	48.70% ± 1.84%	63.11% ± 0.92%
Prototypical network [6]	N	49.42% ± 0.78%	68.20% ± 0.66%
Graph network [13]	N	50.33% ± 0.36%	66.41% ± 0.63%
Meta-SGD [9]	Y	50.47% ± 1.87%	64.03% ± 0.94%
Relation network [14]	N	50.44% ± 0.82%	65.32% ± 0.70%
Ours	N	**52.10% ± 0.60%**	**69.07% ± 0.53%**

validation and 8 for testing. The classes in one category belong to the same high-level concept. Hence, the training classes are sufficiently distinct from the testing classes, making tieredImageNet a more challenging dataset.

The results are shown in Table 3. From Table 3 we can see that on this more challenging dataset, our method also achieves state-of-the-art performance on both the 1-shot and 5-shot scenarios demonstrating the effectiveness of the feature re-weighting strategy again.

4.4 Visualization of Generated Feature Weight

To better understand the weight generator, we further visualize the feature weights to see if it learned those most discriminative features between classes. Figure 4 shows the learned feature weights of two tasks, one comparing a horse to a donkey, the other comparing this horse to a zebra. Here, the visualization of one weight value is achieved by computing the gradient with respect to the input image of the feature equipped with this weight value [30]. By this way, what the feature equipped with one specific weight value looks like can be visualized. As can be seen, when comparing a horse with a donkey, the feature with the top three largest weight value focus on its head totally different from the head of the donkey. And when comparing the same horse with a zebra, the features with the top three largest weight values focus on its partial body without black and white strips which is the most different between these two classes. As expected, the most highlighted feature of the same class under different tasks is different. This result proves that the feature weight produced by the weight generator truly highlights those most discriminative features and is adaptive to different tasks.

Table 3. *tiered*Imagenet few shot classification performance. All accuracy results are averaged over 600 test episodes and are reported with 95% confidence intervals. The performances of other methods are achieved by running their released code. 'Y' in column Finetune means the method fine-tunes the model learned at meta-training stage for a test episode. For each task, the best-performing method is highlighted.

Model	Finetune	5-way	
		1-shot	5-shot
Matching network [12]	N	40.75% ± 0.80%	51.13% ± 0.71%
MAML [10]	Y	44.83% ± 1.85%	66.41% ± 0.09%
Meta-SGD [9]	Y	49.90% ± 1.92%	65.97% ± 0.09%
Graph network [13]	N	50.84% ± 0.36%	68.67% ± 0.63%
Prototypical network [6]	N	52.12% ± 0.68%	69.82% ± 0.58%
Relation network [14]	N	53.71% ± 0.94%	70.28% ± 0.77%
Ours	N	**53.85% ± 0.67%**	**72.36% ± 0.56%**

We provide more visualization results of generated weights, including success cases shown in Fig. 5 and failure cases shown in Fig. 6. These success cases further verify the effectiveness of our proposed weight generator. The failures could be roughly categorized into two kinds. The first category is caused by distraction of background with rich texture (see a, d in Fig. 6) and high-contrast global features (see b in Fig. 6). In some other cases, the features with larger weights are missing or occluded in the respective original images (see the black gradient images in c, e, f of Fig. 6).

Moreover, Fig. 7 shows some success and failure classification cases of our method and prototypical network [6]. From the figure, we can see that for those middle-level hard queries prototypical network [6] fails to classify, our method can handle them successfully.

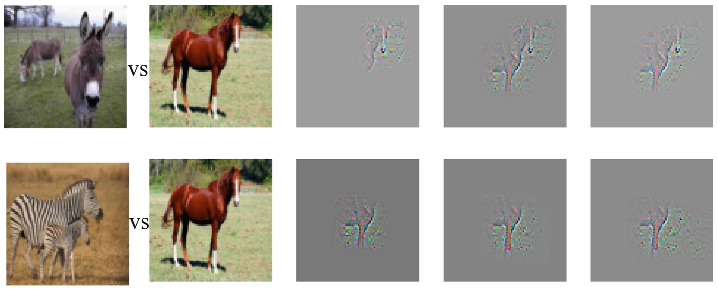

Fig. 4. Feature weight visualization for two different tasks. The first two columns are two classes to be compared in support set. The remaining three columns show the feature equipped with the top three largest weight of the horse.

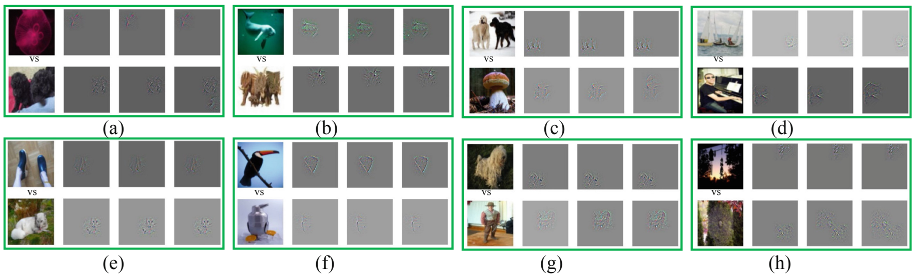

Fig. 5. Visualization of features with top-3 largest weights for each class in 2-way 1-shot setting. Eight success cases given. From these examples we can see that our reweighting mechanism truly highlights those most discriminative features.

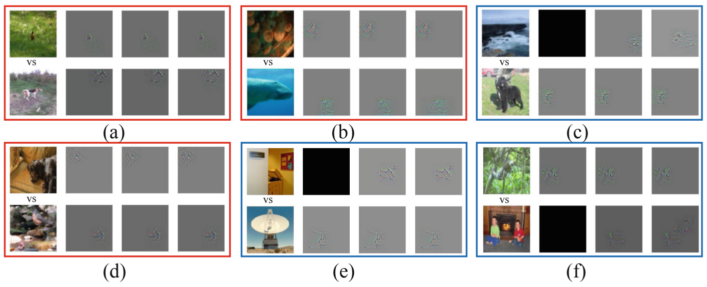

Fig. 6. Visualization of features with top-3 largest weights for each class in 2-way 1-shot setting. Six failure cases given. Failures can be categorized into two kinds: one caused by the distraction of background or high-contrast global feature, e.g. a, b, d, and the other caused by missing (occluded) feature e.g. c, e, f.

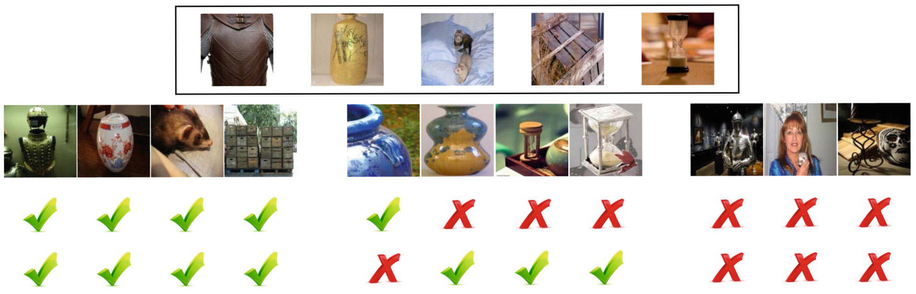

Fig. 7. An example of a 5-way 1-shot classification. The first row and the second row are respectively the support set and the query set. The third row and the fourth row respectively show the classification accuracy of prototypical network [6] and our method.

5 Conclusions

In this work, we propose a novel task-adaptive feature reweighting module to extend the recent prototypical network [6] for better classification performance. The newly introduced feature reweighting module can highlight those most discriminative features of each class to better distinguish it from the rest ones. As a result, each class has its own feature weights and these feature weights are adaptively different in different tasks. We conduct extensively experiments to evaluate the effectiveness of our method. On one hand, we qualitatively demonstrate that the feature weights produced by the weight generator can truly highlight those most discriminative features of each class and are adaptive to different tasks. On the other hand, quantitative experimental results show that our method can achieve the state-of-the-art performance on two commonly used benchmarks for few shot classification task demonstrating the effectiveness of our method.

Acknowledgements. This work was partially supported by National Key R&D Program of China under contracts No.2017YFA0700804 and Natural Science Foundation of China under contracts Nos. 61650202, 61772496, 61402443 and 61532018.

References

1. Krizhevsky A., Sutskever I., Hinton G.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS) (2012)
2. Szegedy C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
4. Huang, G., Liu, Z., Maaten, L., Weinberger, K.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
5. Xu, Z., Zhu, L., Yang, Y.: Few-shot object recognition from machine-labeled web images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
6. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems (NIPS) (2017)
7. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning (ICML) (2016)
8. Yoo, D., Fan, H., Boddeti, V., Kitani, K.: Efficient k-shot learning with regularized deep networks. In: AAAI Conference on Artificial Intelligence (AAAI) (2018)
9. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-SGD: Learning to Learn Quickly for Few Shot Learning. CoRR abs/1707.09835 (2017)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning (ICML) (2017)
11. Wong, A., Yuille, A.: One shot learning via compositions of meaningful patches. In: IEEE International Conference on Computer Vision (ICCV) (2015)

12. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems (NIPS) (2016)
13. Satorras V., Estrach J.: Few-shot learning with graph neural networks. In: International Conference on Learning Representations (ICLR) (2018)
14. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P., Hospedales, T.: Learning to compare: relation network for few-shot learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
15. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Workshop on Deep Learning (2015)
16. Lake, B., Salakhutdinov, R., Tenenbaum, J.: Human-level concept learning through probabilistic program induction. Science **350**, 1332–1338 (2015)
17. Salakhutdinov, R., Tenenbaum, J., Torralba, A.: One-shot learning with a hierarchical nonparametric Bayesian model. In: ICML Workshop on Unsupervised and Transfer Learning (2012)
18. Hariharan, B., Girshick, R.: Low-shot visual recognition by shrinking and hallucinating features. In: IEEE International Conference on Computer Vision (ICCV) (2017)
19. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems (NIPS) (2014)
20. Li, F., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **28**, 594–611 (2006)
21. Ren, M., et al.: Meta-learning for semi-supervised few-shot classification. In: International Conference on Learning Representations (ICLR) (2018)
22. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**, 211–252 (2015)
23. Guo, L., Zhang, L.: One-shot Face Recognition by Promoting Underrepresented Classes. CoRR abs/1707.05574 (2017)
24. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS) (2014)
25. Rezende, D., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: International Conference on Machine Learning (ICML) (2014)
26. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. CoRR abs/1411.1784 (2014)
27. Radford, A., Metz, L., Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR abs/1511.06434 (2015)
28. Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., Smolley, S.: Least squares generative adversarial networks. In: IEEE International Conference on Computer Vision (ICCV) (2017)
29. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein GANs. In: Advances in Neural Information Processing Systems (NIPS) (2017)
30. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. CoRR abs/1312.6034 (2013)