Semi-Supervised Few-shot Pattern Recognition with Meta-Learning

Yibing Liu

Research School of Computer Science,Australian National University
u6068252@anu.edu.au

Abstract. Traditional deep learning models need a large number of training data to get state-of-the-art performance. In this paper, we would like to propose a supervised-prototypical network that can train neural network in small dataset and get good performance. The model will learn a k-means clustering which can classify non-linearly separable patterns. Then we improve it to a semi-supervised model which can utilize coarse-grained data to implement pattern classification in term of few-shot learning. The shared weights network and convolutional neural network used to enhance the models.

Keywords: few-shot learning · deep learning · digits classification

1 Introduction

Based on research, children can recognize a new world after meet it once. [2] And also, a person can split Segway from other vehicles easily at one encounter. [6] So, it is easy to be concluded that humans have the ability to recognize a new object with one instance or few instances. [5] Although the deep learning technique has been successfully used in supervised pattern recognition and classification and gets state-of-the-art results in many problems, like the crowd recognition, yet, to get a good performance, the supervised learning need a large-scale of manually labeled training data by experts, while in many field this is nearly impossible. [11] With a small dataset, the model will be easily overfitted during training. So, we need to develop methods to solve lack of training data.

The supervised prototypical network, proposed by Snell [10], is a few-shot learning method. It attacks the overfitting which is a key problem caused by training with small manually labeled data and the reason that makes the model's ability of generalization poor. The prototypical network assumes that there is a embedding space in which embeddings of digits cluster around a prototype that can represent the class of the digits. [10] The model maps embedding of instances to another matrix space with neural network and then classifies queries by finding the nearest prototypes to them. [10]

In some scenarios, the lack of training data can be relieved by using coarse-grained data. The coarse-grained data is a large dataset without labels but contains instances related to the target patterns. For instance, if we want to train a supervised model to classify pictures of bicycles, cars and planes, we can download a lot of pictures related to them from the web by using search engines. However, in some scenarios, these related instances might be entangled together and are not labeled. In order to utilize them to improve the performance, the model should be able to classify these unlabeled data by itself. M. Ren [9] improves the Prototypical Network to a semi-supervised model which can be trained with combination of a small manually labeled dataset and coarse-grained dataset and get state-of-the-art performance at pattern classification. The semi-supervised model can learn to classify these related but unlabeled data to the correct classes.

Since most pattern classification problems are not linearly separable, the neural network needs to learn a embedding space where the patterns can be classified linearly. The prototypical neural network provides a non-linear layer to learn non-linearity needed by the classification. In order to learn a better embedding space, we use Shared Weights Network, proposed by Gedeon [4], to pre-train the non-linear mapping function to improve the performance of prototypical network. In the bottom of the Prototypical Network, the convolutional neural network (CNN), proposed by LeCun [7], is used to extract features from the images and represent them with better feature embeddings.

In this paper, the cascading classifier, proposed by Alpaydin [1], is the baseline which is compared with the supervised prototypical network to measure the learning capacity of few-shot learning models when training dataset is small. Then a convolutional neural network is applied in the bottom of the prototypical model to improve the representation of training instances and shared weights are used to pre-train the non-linear function to get a better mapping. We then compare the semi-supervised prototypical network with the supervised model in term of pattern classification.

2 Method

The small manually labeled training dataset can be represented as $\mathbf{D} = \{y_i, \boldsymbol{x}_i | 0 \leq i \leq |\mathbf{D}|\}$. The $\boldsymbol{x}_i \in \mathbb{R}^N$ is a N dimension embedding which represents attributes of an instance. The $y_i \in \{0, ..., K\}$ is a label representing the class to which the $\boldsymbol{x}_i$ belongs. The model will input instances and then predict to which class they belong in the output layer. Since the model's aim is to classify patterns, by minimizing difference between the predicted label and the true label, our model is trained.

2.1 Supervised Prototypical Network

The Supervised prototypical network,proposed by Snell [10], will split the training dataset into two parts. The $\mathbf{D}_t$ is collection of query examples and the $\mathbf{D}_p$ is collection of support vectors. We use $\mathbf{D}_p^k$ to denotes the set of instances labeled as class $k \in \{0, 1, ..., K\}$ in $\mathbf{D}_p$ and $\boldsymbol{x}_i^k \in \mathbf{D}_p^k$ represents a example in it. The model maps original N-dimensional features to M-dimensional space through a non-linear function with learnable parameters θ as the following:

$$f_\theta : \mathbb{R}^N \rightarrow \mathbb{R}^M \tag{1}$$

We map $\boldsymbol{x}_i^k \in \mathbf{D}_p^k$ to M-dimensional space with Function 1 to get support points of class k. Then each prototype of class k, $\mathbf{p}_k$, is mean vector of support points and calculated as the following:

$$\boldsymbol{p}_k = \frac{1}{|\mathbf{D}_p^k|} \sum_{i=1}^{|\mathbf{D}_p^k|} f_\theta(\boldsymbol{x}_i^k) \tag{2}$$

When a query $\boldsymbol{x}_i \in \mathbf{D}_t$ is given, the squared euclidean distance function *dist* is used to compute the distance between $\boldsymbol{x}_i$ and each prototype $\boldsymbol{p}_k$, and then we use softmax to produce the distribution over classes for the query as shown in equation 3. The query will be classified to the class with maximum probability.

$$p(k|\boldsymbol{x}_i) = \frac{\exp\left(-dist(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{p}_k))\right)}{\sum_{k'=0}^{K} \exp\left(-dist(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{p}_{k'}))\right)} \tag{3}$$

To train the model, we need to minimize negative log-probability, $-log(p(k|\boldsymbol{x}_i))$, of true class k. Few-shot learning of supervised prototypical network is shown in figure 1.

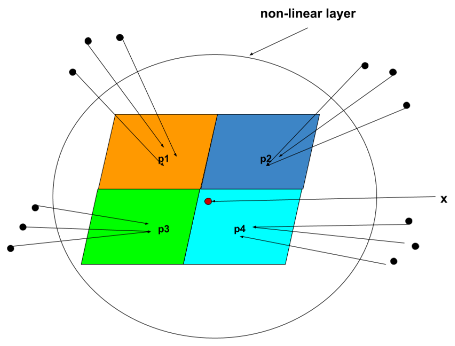

Fig. 1: Supervised Prototypical network for few-shot learning. $\boldsymbol{p}_1$,$\boldsymbol{p}_2$,$\boldsymbol{p}_3$,and $\boldsymbol{p}_4$ are prototypes of each class calculated by mean of support examples of corresponding class. $\boldsymbol{x}$ is query point which is classified to a cluster whose prototype is the closest to it. The dark points represent support instances. The query points and support instances are mapped to another matrix space by a non-linear layer.

2.2 Semi-supervised Prototypical Network

The semi-supervised prototypical network is proposed by Ren [9]. It assumes that extra unlabeled examples, denoted by $\mathbf{D}_u = \{\hat{\boldsymbol{x}}_i | 0 \leq i \leq |\mathbf{D}_u|\}$, will be combined with small manually labeled data set $\mathbf{D}_p$ as support examples. The $\hat{\boldsymbol{x}}_i \in \mathbb{R}^N$ is an unlabeled instance. These unlabeled instances are used to refine prototype $\boldsymbol{p}_k$ to $\hat{\boldsymbol{p}}_k$.

All instances in the unlabeled dataset can be classified to one of the legitimate classes as shown in figure 2. Therefore, the soft k-means can be used to refine prototypes. The unlabeled instances will be mapped to the M dimensional embedding space by function 1 and partially assigned to each legitimate class. The prototypes of each legitimate class will be refined as the following:

$$\hat{\boldsymbol{p}}_k = \frac{\sum_{i=1}^{|\mathbf{D}_p^k|} f_\theta(\boldsymbol{x}_i^k) + \sum_{j=1}^{|\mathbf{D}_u|} w_k f_\theta(\hat{\boldsymbol{x}}_j)}{|\mathbf{D}_p^k| + |\mathbf{D}_u|}, \text{where} \quad w_k = \frac{\exp\left(-||f_\theta(\hat{\boldsymbol{x}}_j) - \boldsymbol{p}_k||^2\right)}{\sum_{k'=1}^{K} \exp\left(-||f_\theta(\hat{\boldsymbol{x}}_j) - \boldsymbol{p}_{k'}||^2\right)} \tag{4}$$

The w_k is a weight that measures the compatibility between unlabeled instance $\hat{\boldsymbol{x}}_j$ and prototype $\hat{\boldsymbol{p}}_k$. Then as in section 2.1, a query $\boldsymbol{x} \in \mathbf{D}_t$ will be classified by equation 3 with refined prototypes and the model will be trained by minimizing negative log-probability, $-log(p(k|\boldsymbol{x}))$, of true class k.

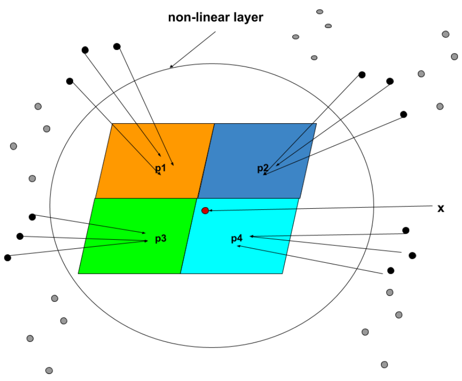

Fig. 2: Soft K-means Semi-supervised Prototypical Network. The $\boldsymbol{p}_1$,$\boldsymbol{p}_2$,$\boldsymbol{p}_3$,and $\boldsymbol{p}_4$ are prototypes of each class calculated by mean of support examples of corresponding class and will be refined to $\hat{\boldsymbol{p}}_i$ where $i \in 1, ..., 4$. The $\boldsymbol{x}$ is query point which is classified to a cluster with the closest refined prototype to it. The dark circles represent support instances and the gray are unlabeled instances. They both can be classified to legitimate classes represented by $\boldsymbol{p}_1$,$\boldsymbol{p}_2$,$\boldsymbol{p}_3$,and $\boldsymbol{p}_4$. The query points, unlabeled points and support instances are mapped to another embedding space by a non-linear layer.

2.3 Shared Weights Network

The shared weights model compresses and decodes the image, which means that the image is the input, compressed by several hidden layers and then symmetrically decoded by other hidden layers.[9] These hidden layers used to compress images share weights with symmetrical layers to decode images. The weight-sharing mechanism is implement by a simulator in which during back-propagation the weight are different and updated by the standard algorithm, and after that we average weights of symmetrical layers in the compressing and decoding process.[9] In figure 1, it is a shared weights network with one hidden layer. $\boldsymbol{W1}$ is weights matrix that maps input to hidden layer and $\boldsymbol{W2}$ maps hidden layer to output layer. Connection between input and hidden layer shares weights with the one between hidden and output layer. So,the value of $\boldsymbol{W1}$ and $\boldsymbol{W2}$ is the same.

We use D to denote the collection of images used to train the shared weights network. $\boldsymbol{x} \in D$ is a image, and it is compressed by a function $f(\boldsymbol{x}) = \boldsymbol{x} * \boldsymbol{W_1^T} + \boldsymbol{b_1}$ and decoded by another function $g(\boldsymbol{h}) = \boldsymbol{h} * \boldsymbol{W_2^T} + \boldsymbol{b_2}$. After the compressed image is decoded, we minimize the distance between the original image $\boldsymbol{x}$ and the predicted one to train the shared weight network. So, the loss function is:

$$l(\boldsymbol{x}) = d(x, g(f(x))) \tag{5}$$

In Function (4), the d represents distance between two images and we choose euclidean distance to compute the difference between original image and the generated one. In prototypical network, when squared euclidean distance is used in Equation (3), it is a linear model.[8] Since the digits are not linearly separable, we add a non-linear layer

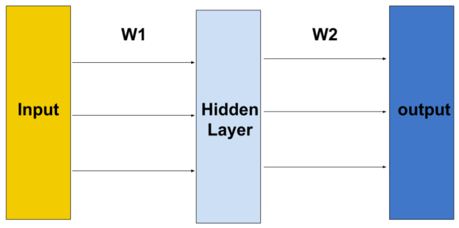

Fig. 3: Shared Weights Network with one hidden layer

to map the feature vector to a high dimensional space where the digits can be separated with linear model. So, to learn a better model, the shared weight network is used to pre-train the non-linear layer, and then we fine-tune the pre-trained non-linear in prototypical network.

2.4 Cascading Classifier

The cascading classifier is a multistage algorithm. It builds a cascade of a parametric linear model which learn rules and a non-parametric k-nearest neighbor (k-NN) model which learns exception rejected by rule. [1] It also divides the training data $\mathbf{D}$ into two parts: $\mathbf{D}_r$ for linear model to learn rules and $\mathbf{D}_{exc}$ for K-NN to collect exceptions. We use cascading classifier as the baseline to compare with supervised prototypical network when the training data is reduced.

3 Experiment Settings

In this section, we introduce dataset used in experiments and an overview of all experiments.

3.1 Dataset

I choose two dataset to work with as the following:

Optical Recognition of Handwritten Digits Dataset (UCI dataset) [3] aims to let machine to classify images of handwritten digits by 44 persons. Digits' images are split into two parts: training data from 30 persons and test data from the other different 14 persons. Digits recognition is a interesting problem in computer vision, and this well-formed dataset makes it easy to investigate the problem. Since each attribute is integer from 1 to 16, we don't need to re-normalize them. The test data is writer-independent which can make the overfitting caused by lack of training data more obvious. If the model has poor ability of generalization, it will perform worse in test dataset written by totally different people. It will be used to compare the performance between few-shot learning network and traditional neural network.

MNIST Handwritten Digits Dataset (MNIST dataset) [8] which contains 60,000 images of digits for training and 10,000 for testing. These images have been normalized to a bounding box in which each image is represented by a 28 by 28 matrix of pixels. It contains ten classes which are corresponding to digits from 0 to 9. The dataset is easy to be preprocessed and we will use it to measure the neural network with CNN as the bottom layers.

The UCI dataset is used to measure the learning capacity of few-shot learning models by comparing supervised prototypical neural network and cascading classifier. Also, impact of the shared weights can be measured between the prototypical models with or without pre-training. The MNIST dataset is used to evaluate the influence of shared weights and convolutional neural network by adding them to the prototypical network. We also compare the performance of semi-supervised and supervised prototypical network in term of few-shot learning.

3.2 Metrics

We use accuracy and macro f1 score to evaluate the performance of our model. Accuracy exactly shows the ratio of queries that are correctly classified, and macro f1 score comprehensively measures the precision and recall.

3.3 Experiments Overview

The experiments are trying to answer the following questions:

1. How is the performance of few-shot learning networks compared with traditional neural networks?
2. What's the influence of shared weights and convolutional neural network?
3. How does the performance of semi-supervised prototypical network compare with the supervised one in term of pattern classification?

In section 4.1, we will compare the cascading classifier with the supervised prototypical network to evaluate the learning ability of few-shot learning models when training dataset is small. Then in section 4.2 a convolutional neural network is applied in the bottom of the prototypical model to improve the representation of training instances and shared weights are used to pre-train the non-linear function to get a better mapping. We compare the influence of them to the prototypical networks. We compare the influence of both to the supervised prototypical networks. Finally, we compare the semi-supervised prototypical network with the supervised model in term of pattern classification in section 4.3.

4 Results and Discussions

In this section, we will show results of a series of experiments and discuss the influence of share weights, convolutional neural network and evaluates the capacity of few-shot learning models. We use **Cascading** to represent the cascading classifier. The **super-Proto** is used to represent the supervised prototypical network and the **share-super-Proto** is used to represent the supervised prototypical network which is pre-trained with shared weights network. The **semi-Proto** denotes the soft k-means semi-supervised prototypical network. We will directly indicate whether the model's bottom layer is CNN in each experiment.

4.1 Comparison of Cascading Network and Supervised Prototypical Network

Metrics / Network	Accuracy	Macro F1
Cascading	0.8894	0.8887
super-Proto	0.9360	0.9354

Table 1: result of experiments

In this section, we compare the performance of **Cascading** and **super-Proto** when the training dataset is small. In this way the learning capacity of few-shot learning models when training dataset is small can be measured. The bottom of them is a linear layer without using convolutional neural network. The UCI dataset is used in the experiments. It has been split into two clusters. The one written by 30 persons is used as training data and another by 14 persons as writer-independent test data. The training data is divided into three part: 1934 examples as query set, 946 examples as support set, and the 943 as writer-dependent test set.The query set is used as $\mathbf{D}_t$ for prototypical network and $\mathbf{D}_r$ for cascading network. The $\mathbf{D}_p$ of prototypical and $\mathbf{D}_{exc}$ of cascading classifier is given support set. We test these two models with writer-independent test data. To test their performance when the training data is small, we reduce the examples in query set. We refine $\mathbf{D}_p$ and $\mathbf{D}_{exc}$ to keep 50 examples for each class and in total there will be 500 examples. To see their performance when data set is small, we would reduce the training dataset, $\mathbf{D}_t$ and $\mathbf{D}_r$, to contain 500 examples.

In Table 1, when the training data is reduced to contain 500 examples, the accuracy and macro f1 score of **cascading** is 0.8894 and 0.8887, and for **super-Proto** they are 0.9360 and 0.9354. We can see that the performance of **super-Proto** is much better than the cascading network. When the training dataset contains 1934 examples, the accuracy of **Cascading** is nearly 96%. [1] When training dataset is small, it performs much worse.

From the result we can see that, the **super-Proto** beats the cascading network when the training dataset is small. With a small training dataset, the **Cascading** performs less better compared with itself trained in a large dataset. The main reason is that the **Cascading** just trains a linear model which learns rules and gives exceptions to its K nearest neighbor classifier. [1] The K nearest neighbor classifier performs as a non-linear classifier, but it is not trainable. When training data is small, the linear model cannot learn a good rule, and its bad performance will influence K nearest neighbor which cannot learn by itself. The K nearest neighbor doesn't perform well when the number of exceptions increase because of poor generalization of linear model. The **super-Proto** can learn a better non-linear mapping than the cascading network even the training dataset is small.

4.2 Impact of Shared Weights and Convolutional Neural Network

Impact of shared weights To measure the influence of shared weights to supervised prototypical network, we use the same setting of UCI dataset in section 4.1 to train and test the **super-Proto** and **share-super-Proto**. The $\mathbf{D}_t$ is used to pre-train the non-linear layer of supervised prototypical network with shared weights network. The bottom of them is still linear layer. The results are shown in table 2. It shows that when we pre-train the prototypical network with shared weights network, the accuracy and f1 score reduce to 0.8647and 0.8631 which is worse than the supervised prototypical network only and even the cascading network.

Metrics / Network	Accuracy	Macro F1
super-Proto	0.9360	0.9354
share-super-Proto	0.8647	0.8631

Table 2: Result of supervised prototypical network with or without pre-training.

So, we can conclude that the pre-training learns a non-linear function which is not suitable to prototypical network. Yet, in another aspect, the result of prototypical network with pre-training shows that the pre-training process will influence the final performance of prototypical network. Although shared weights network is not competent to learn a better non-linear mapping, there are other networks we can explore.

Impact of CNN To measure the influence of the CNN, we compare performance between the supervised prototypical network with linear layer as bottom and the one with CNN. The dataset used for training and test is full MNIST dataset. The training dataset is also split into support dataset and query dataset. The $\mathbf{D}_t$ and $\mathbf{D}_p$ respectively contain 30000 images. The results are shown in table 3. The accuracy of the model with linear bottom is 0.9305 and the f1 score is 0.9314. For the model with CNN in the bottom, the accuracy and f1 score is respectively 0.9898 and 0.9899.

Metrics / Network	Accuracy	Macro F1
super-Proto with linear bottom	0.9305	0.9314
super-Proto with CNN bottom	0.9898	0.9899

Table 3: Result of supervised prototypical network with linear layer or CNN in the bottom.

Form the above results, we can conclude that the supervised prototypical network with CNN performs much better than another one. The CNN gives a better representation of images and the new feature embeddings can make the non-linear layer in the model learn a better mapping.

Impact of CNN and shared weights comprehensively We apply both convolutional neural network and shared weights network to the supervised prototypical network, and the result of training and test in MNIST dataset is shown in the table 4. To evaluate their influence comprehensively, we also train and test another model on the MNIST dataset which only replace the CNN bottom with linear layer. The setting of the MNIST dataset is the same to the setting when we measure impact of CNN.The accuracy and f1 score of comprehensive supervised prototypical network is respectively 0.8470 and 0.8493. When replace CNN with linear bottom, the accuracy and f1 score become 0.7967 and 0.7999.

Network \ Metrics	Accuracy	Macro F1
share-super-Proto with linear bottom	0.7967	0.7999
share-super-Proto with CNN bottom	0.8470	0.8493

Table 4: Result of pre-trained supervised prototypical network with linear or CNN bottom.

From table 4 and 3, we can conclude that the shared-weights will degrade the promotion from CNN. Even with a better representation of images, the shared weights network still cannot learn a better non-linear mapping function.

Summary From the above experiments, we can conclude that the CNN can promote the performance of prototypical network by representing the images with better feature embeddings, but the shared weights pre-training will degrade the non-linear function. Since the semi-supervised model adopt the similar structure, the influence of shared weights and CNN will be the same to it.

4.3 Comparison of Supervised and Semi-supervised Prototypical Network

When we compare the semi-supervised and supervised model in term of few-shot classification, we adopt the CNN and give up shared weights pre-training. We use the MNIST dataset to perform 5-way classification which means that the models will train and test with instances that can be classified to 5 classes. For each class, the $\mathbf{D}_p^k$ contains 5 instances and 10 are for the query set. We also randomly sample instances from the same classes to form $\mathbf{D}_u$ for semi-supervised model. Both of them have the CNN as bottom. The result is shown in table 5. The accuracy and f1 score for supervised model are respectively 0.9026 and 0.9040. For semi-supervised one, the value is 0.8872 and 0.8880.

Network \ Metrics	Accuracy	Macro F1
super-Proto	0.8990	0.9000
semi-Proto	0.8872	0.8880

Table 5: Result of semi-supervised and supervised prototypical network.

From the result we can see that the supervised model have better performance. The semi-supervised model doesn't perform better by utilize these extra training data. With more experiments, we find that, occasionally, the performance of semi-supervised model is better than the supervised model, but this is seldom. However, the results also show that the semi-supervised model can successfully classify the unlabeled dataset to correct classes, and this inspire us to work with another scenario. Sometimes, the coarse-grained dataset from web might include noises which are classes that cannot be classified to legitimate class. For instance, images of bicycle might be entangled with planes. So, the semi-supervised model can be explored further.

5 Conclusion and future work

In this paper we talk about the few-shot learning which aims to train a model having art-of-state performance for pattern classification with small training dataset. We propose a prototypical network which have better performance than cascading network on classification when the training dataset is small. Since the instances are not linearly separable, the prototypical network uses a non-linear function to learn all non-linearity. We want to use shared weights network to pre-train the non-linear function to get better performance. Yet, the experiments shows that the shared weights network cannot learn a better non-linear function to map the original image to a linear separable space. Therefore, we can explore other networks to find one which is competent. To improve their performance, we also use CNN to represent the images in a better way and the experiments show that it promotes the performance sharply. We then introduce semi-supervised model to explore scenarios where extra related training data is available but is not labeled. The semi-supervised model successfully utilize the unlabeled instances by learning but its performance doesn't become better. However, in the future, we can explore methods to combine a noised dataset with a small manually labeled dataset to train the deep learning model. Although the prototypical network can use few data for each class to get good performance on pattern recognition, yet to predict a existing class it still needs some training data belonging to it. If there is not training data for a class, how can we train a model to get good performance? This problem is called zero-shot learning. So, in the future we can improve the prototypical network for zero-shot learning.

References

1. Alpaydin, E., Kaynak, C.: Cascading classifiers. Kybernetika **34**(4), 369–374 (1998)
2. Carey, S., Bartlett, E.: Acquiring a single new word. (1978)
3. Dheeru, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
4. Gedeon, T., Catalan, J., Jin, J.: Image compression using shared weights and bidirectional networks. In: Proceedings 2nd International ICSC Symposium on Soft Computing (SOCO'97). pp. 374–381
5. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop. vol. 2 (2015)
6. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: Proceedings of the Annual Meeting of the Cognitive Science Society. vol. 33 (2011)
7. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10), 1995 (1995)
8. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
9. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676 (2018)
10. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. pp. 4080–4090 (2017)
11. Zhang, L., Shi, M.: Crowd counting via scale-adaptive convolutional neural network. arXiv preprint arXiv:1711.04433 (2017)