Small-Sample Image Classification Method of Combining Prototype and Margin Learning

Xiaoxu Li*, Liyun Yu*, Dongliang Chang†, Zhanyu Ma†, Nian Liu‡ and Jie Cao*
* Lanzhou University of Technology, Lanzhou, China
E-mail: xiaoxulilut@gmail.com
† Beijing University of Posts and Telecommunications, Beijing, China
E-mail: mazhanyu@bupt.edu.cn
‡ South-to-North Water Diversion Middle Route Information Technology Co., Ltd., China
E-mail: liunian@nsbd.cn

***Abstract*—Image classification is a fundamental and important task in the field of computer vision and artificial intelligence. In recent years, image classification has made breakthrough progress based on deep learning on large-scale datasets. However, it still exits big challenges on small-sample image data. The main difficulty is that the deep neural network easily overfit small-sample data and has big variance. Ensemble learning is a good way to overcome overfitting and reduce the variance of the model; however, the existing ensemble methods based on deep neural network still could overfit on small-sample image data due to the big randomness of deep neural network. In this paper, we propose a new ensemble method for small-sample image classification tasks. The proposed method based on VGG16 network, we modified the structure of the VGG16 network to two branches, one branch is a classifier based on prototype learning, and the other is a classifier based on margin learning. The experimental results on two small-sample image datasets, the LabelMe dataset and the Caltech101 dataset, show that the proposed method has better performance and higher stability than other referred methods.**

***Index Terms*—Small-sample, ensemble method, prototype and margin learning**

I. Introduction

In recent years, with the development of deep learning, convolutional neural networks (CNNs) are increasingly used in computer vision field, such as image classification [1], [2], object recognition [3], [4], character recognition [5], video tracking [6], etc. In the past few years, due to the rapid development of the deep learning and the various complex algorithms, the tasks of image recognition and scene classification have reached a high level on large-scale data samples. However, over-fitting problems will inevitably occur under small-sample conditions when the depth of the model is large. In addition, many kinds of data are following the long tail distribution in real life, obtaining a large number of samples are time-consuming and infeasible. Therefore, how to solve the problem under the small-sample datasets is a hot topic.

Among the relation work of small-sample classification, regularization techniques and ensemble learning are commonly used methods. Regularization techniques are effective approaches to relieve the overfit neural network. Commonly used regularization techniques include Parameter Norm Penalties [7], Dropout [8], Noise robustness [9], Early termination [10], etc. The Dropout method prevents the severe joint adaptation of neurons by randomly deleting neurons and their connections in the neural network during training. DropConnect [11] is an extension of Dropout, it is proposed by L.Wan. Different from Dropout, DropConnect randomly selects a subset of weights from the network and sets it to zero. In 2016, Tong Xiao et al. proposed a Domain Guided Dropout [12]. Different from the standard dropout, all neurons are treated equally. Domain Guided Dropout assigns a specific dropout rate for each domain according to the validity of each neuron on that domain. Most of these works achieved good performance. However, these regularization techniques can not well relieve overfitting problems in small-sample image classification tasks.

Ensemble learning is a good way to overcome overfitting and reduce the variance of the model. The ensembling method [13], [14] is based on the neural network can be roughly divided into two categories. One category is the classical ensembling classification method, including Bagging and its variants [15], [16], Boosting and its variants, and Mixture of Experts (MoE) [17]. This category of ensembling has a significant effect on the large-scale dataset. However, each base classifier has a large generalization error when the sample size of the training dataset is relatively small, which will also reduce the generalization performance when ensembling all base classifiers.

Another category is the new ensembling method which takes advantage of the characteristics of neural network. Most of them focus on how to obtain multiple base classifiers without increasing training time. For example, Alan Mosca[18] proposed a new method, Boosted Residual Networks, which takes advantage of the development of deep learning and previous white-box ensembling to achieve improved results for benchmark datasets. Gao Huang et al. [19] proposed Snapshot Ensembling, which ensembling multiple neural networks without increasing training costs by training a single neural network, it converges along its optimized path to multiple local minima, and preserves model parameters. Temporal Ensembling is proposed by Samuli Laine et al [20]. Temporal Ensembling trains on a single network by self-ensembling, combines the Dropout regularization under different conditions, different regularizations and input enhancement conditions to generate

multiple sub-networks, and obtains the final prediction.

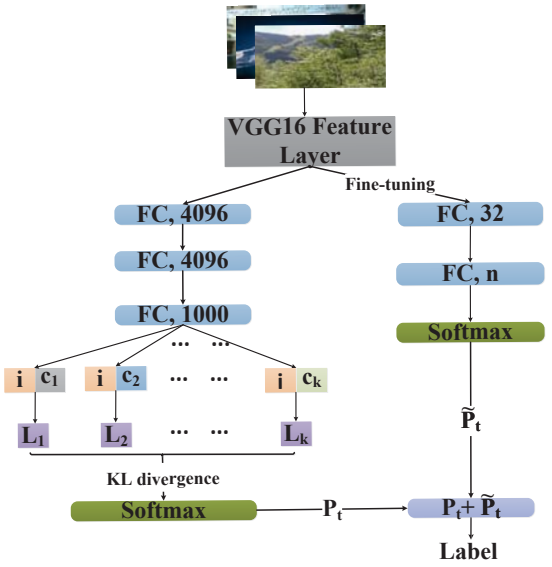

Fig. 1. The ensembling method of combining prototype and margin learning. The left branch is a classifier based on prototype learning, and the right branch is a classifier based on margin learning.

The above ensembling classification methods have achieved good results under large-scale samples. However, due to the big randomness of deep neural network, the existing ensemble methods still overfit on small-sample image data. And these methods are also difficult to ensure base classifiers have better performance and larger diversity. In this paper, we propose a new ensembling classification method. The propose ensembling method conclude two branches, one branch is a classifier based on prototype learning, and the other is a classifier based on margin learning. The two branches learn separately in training, and they are used together in the testing process.

To investigate the effectiveness of this method, we conducted experiments on the LableMe [21] dataset and the Caltech101 [22] dataset. The experimental results show that the propose method has a good generalization performance on these two datasets. In addition, compared with other methods, the propose method obtains more stable results.

II. Theoretical Basis

The proposed ensembling method of combining prototype and margin learning is shown in Figure 1. Our method based on the original VGG16 network, we modified the structure of the VGG16 network to two branches.

As shown in Figure 1, the left branch of the network is the same as the original VGG16 classifier. Considering we use small-sample datasets, the right branch of the network only uses two randomly initialized fully connected layers. Different from the original VGG16, the feature layer and the left branch of the network are pre-trained on the Imagenet dataset. After that, the parameters of the feature layer and the left branch are frozen during the network training. We use the output of the fully connected layers to calculate the class center of each class when the training is over, and each class center is saved. For the right branch of the network, we use cross-entropy loss for model optimization and use a lower learning rate to fine-tuning it. Hence, we can regard the left branch of the network as a prototype learning, and the right branch of the network is seen as a margin learning. Next, we introduce the process of the proposed method in detail.

Suppose x_i is the *i*-th sample under the training set, considering a K class classification task in which the softmax loss is used. For an input sample with $f(x)$ as its extracted deep feature vector of the left branch, its probability can be expressed by Eq. 1, in which $i \in M$, M is the number of training samples.

$$X_i = Softmax(f(x_i)) \ . \tag{1}$$

Because of the parameters of feature layer and the left branch of the network are frozen during the network training, we only need to optimize the right branch of the network. Therefore, we use cross-entropy loss and a lower learning rate to fine-tuning it.

After the model optimization is over, we can use the output of the left branch of the network to calculate the class center of each class, it can be expressed as Eq. 2, in which $j \in [1, ..., K]$, C_j represents a set of samples in the *j*-th class.

$$X_j = \frac{1}{M} \sum_{X_i \in C_j} X_i \ . \tag{2}$$

In the testing process, on the left branch of the network, we calculate the probability value of each test sample belonging to each class, it can be expressed X_t. Then, the distance between the *t*-th test sample and the *j*-th class center is calculated by KL divergence, it can be expressed by Eq. 3, in which D_{KL} represents the KL divergence. After that, the KL divergence is converted into the corresponding probability value P_t by Equation 4, in which $L_t = [L_{t1}, ..., L_{tk}]$.

$$L_{tj} = D_{KL}(X_j || X_t) \ . \tag{3}$$

$$P_t = Softmax(L_t) \tag{4}$$

Similarly, the right branch of the network can also get the probability values $\tilde{P}_t$ of testing samples, it can be reflected in Eq. 5, in which x_t indicates the *t*-th sample of testing dataset; g(x) indicates the extracted deep feature vector of the right branch. Finally, we combine the probabilities of the left branch P_t and the right branch $\tilde{P}_t$ to classify the test samples, as shown in Eq. 6, in which a is hyperparameter.

$$\tilde{P}_t = Softmax(g(x_t)) \ . \tag{5}$$

$$P = aP_t + \tilde{P}_t \ . \tag{6}$$

TABLE I
Comparison of the classification accuracies and standard deviation of four methods on the LabelMe dataset and the Caltech101 dataset.

Dataset		Snapshot	Dropout	VGG16	Ours
LabelMe	Mean	0.9088	0.9077	0.9112	**0.9201**
	Std.	0.0044	0.0036	0.0054	0.0036
Caltech101	Mean	0.9188	0.9206	0.9238	**0.9289**
	Std.	0.0025	0.0026	0.0027	0.0024

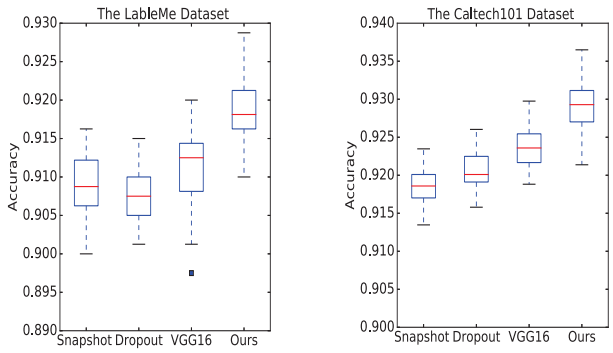

Fig. 2. Box plot comparison of the accuracies obtained by Snapshot, Dropout, VGG16, Ours on the LabelMe dataset and the Caltech101 dataset. The central mark is the median, and the edges of the boxes are the 25th and 75th percentiles. The outliers are marked individually. Each method is run 30 times to produce the box plots.

III. Experimental Results And Analysis

We focus on experimental results and data analysis on the LabelMe dataset and the Caltech101 dataset in this section. The preprocessing of experimental data and network structure are introduced on the first part. The specific network parameter settings and the specific experimental results are introduced on the second and third part.

A. Datasets

1) LabelMe Dataset: The LabelMe is a natural scene images class dataset, it contains 8 classes: coast, mountain, forest, open country, street, inside city, tall buildings and highways. We randomly select 210 images for each class, of which 100 images, 100 images and 10 images where are used for the training dataset, test dataset and validation set, respectively. The total number of images is 1680.

2) Caltech101 Dataset: The Caltech101 is a digital images dataset, it was created in 2003 by the California Institute of Technology, Li Fei Fei and Marco Andreetto. The dataset contains 9,146 images and 101 classes, it including ”human face”, ”animal”, ”mechanical”, ”landscape” and so on. Each category has 40 to 800 images, most of which have about 50 images, and each image is approximately 300×200 pixels.

B. Network Structure And Parameter Settings

We use a single-input, multi-output network architecture on the LableMe and the Caltech101 datasets. One branch is the classifier of the VGG16 network, we freeze the network parameters. The other is the classifier of fully connected layers, and the number of hidden layers is set to 32. After training, we used the validation set to select the hyperparameter. Thus, our hyperparameter a is set to 0.45 on the LabelMe dataset, and it is set to 1 on the Caltech101 dataset.

To fine-tune the right branch of the network, we use the RMSprop optimization algorithm with a momentum of 0.9 and set the batch size to 32, the initial learning rate is set to 0.0001; the loss function uses the cross-entropy loss; and the number of epochs to 200. The L2 parameter is set to 0.005 when the dataset is LableMe, and it is 0.0005 when the dataset is Caltech101. And we select the weight of the minimum loss on the train set as the final weight after iteration 200 epochs.

C. Classification Accuracies

We compare four methods, Snapshot, Dropout, VGG16 and Ours on the LabelMe and the Caltech101 dataset, each method is run 30 times. VGG16 is the network that we remove the the left branch and leave the right branch. The results of accuracy(acc) and standard deviation (std.) are shown on Table I.

As shown in Table I, on the LableMe dataset, the accuracy and standard deviations are 0.9201 and 0.0036 for Ours, 0.9112 and 0.0054 for VGG16, 0.9077 and 0.0036 for Dropout, and 0.9088 and 0.0044 for Snapshot. On the Caltech101 dataset, the accuracy and standard deviations are 0.9289 and 0.0024 for Ours, 0.9238 and 0.0027 for VGG16, 0.9206 and 0.0026 for Dropout, and 0.9188 and 0.0025 for Snapshot. Our method has improved about 1% to 2% compared with another methods. The values of standard deviation are also lower. The experimental results show that the proposed method achieves competitive performance on the LabelMe and the Caltech101 datasets. To further investigate the effect of the proposed method on the robustness and stability, we present box plots of the accuracies obtained by Snapshot, Dropout, VGG16 and Ours in Figure 2. In Figure 2, we can observe that on the LabelMe dataset, the box plot of Ours is more compact than that of VGG16, Snapshot and Dropout. Meanwhile, the maximum, median and minimum accuracies of Ours are higher than others. Therefore, compared with other methods, the proposed method have better stability.

D. Effect On The Different L2 Norm And Learning Rate

We also change the learning rate and the L2 norm to further compare the performance of each method. Where the L2 norm is a method of regularization. We first fix the learning rate of the network, and change the L2 parameters (0.0005, 0.005, 0.05, 0.5) by a factor of ten. Similarly, we change the learning rate (0.0001, 0.001, 0.01, 0.1) by a factor of ten after fix the L2 (0.0005) parameter. The results are shown in Table II and Table III.

TABLE II
Comparison of the accuracies obtained by VGG16, Ours on the LabelMe dataset and the Caltech101 dataset at different L2 norm values. The learning rate(Lr) is 0.0001.

Dataset	Lr	L2	VGG16	Ours
LabelMe	0.0001	0.0005	0.9112	**0.9185**
		0.005	0.9100	**0.9201**
		0.05	0.9108	**0.9169**
		0.5	0.9087	**0.9097**
Caltech101		0.0005	0.9238	**0.9289**
		0.005	0.9232	**0.9279**
		0.05	0.9069	**0.9172**
		0.5	0.5626	**0.8323**

TABLE III
Comparison of the accuracies obtained by VGG16, Ours on the LabelMe dataset and the Caltech101 dataset at different learning rate(Lr). The L2 norm value is 0.0005.

Dataset	L2	Lr	VGG16	Ours
LabelMe	0.0005	0.0001	0.9112	**0.9185**
		0.001	0.9048	**0.9134**
		0.01	0.9006	**0.9019**
		0.1	0.3167	**0.8362**
Caltech101		0.0001	0.9238	**0.9289**
		0.001	0.8806	**0.9048**
		0.01	0.4353	**0.7760**
		0.1	0.0956	**0.7442**

As shown in Table II and Table III, we can observe that when the learning rate is 0.0001 and L2 changes from 0.0005 to 0.5, the accuracy of the VGG16 and Ours varies little on LabelMe dataset. While on the Caltech101 dataset, the accuracy of the VGG16 has a distinct changed. More specifically, when the L2 is 0.5, the accuracy of the VGG16 is 0.5626, and the accuracy of Ours is 0.9097. Similarly, when the L2 is 0.0005 and the learning rate changes from 0.0001 to 0.1, the accuracy of the VGG16 has a distinct changed, but the accuracy of Ours changes slightly. Especially when the learning rate is 0.1, the accuracy of the VGG16 is 0.3167 and Ours is 0.8362 on the LabelMe dataset. While on the Caltech101 dataset, The accuracy of Ours is 0.7442, the VGG16 is 0.0956.

From the above analysis, our method has better stability in the network than VGG16. The reason may be that the branch based on prototype learning of the network has a certain corrective effect on the learning of its branch based on margin learning in our method. When the learning effect of the branch based on margin learning is worse, the more influence of the branch based on prototype learning. Therefore, the accuracy of the network can be improved.

E. Ablation Study

To show more experimental details, we show the accuracy and standard deviation values of the branch based on prototype learning, the branch based on margin learning, and Ours. The details are shown in Table IV.

TABLE IV
Accuracy and standard deviation of the two branches. Ours-Left: the branch based on prototype learning of the network. Ours-Right: the branch based on margin learning of the network. Ours: combine of the prototype learning and margin learning.

Dataset		Ours-Left	Ours-Right	Ours
LabelMe	Mean	0.8238	0.9113	**0.9200**
	Std.	0.008	0.0056	0.0034
Caltech101	Mean	0.7451	0.9212	**0.9279**
	Std.	0.0042	0.0021	0.0024

As shown in Table IV, on the LabelMe dataset, the accuracy of the Ours-Left is 0.8238, the accuracy of the Ours-Right is 0.9113, and the accuracy of Ours is 0.9200. The standard deviation of Ours has decreased about 0.0022. On the Caltech101 dataset, the accuracy of the VGG16-Left prediction is 0.7541, the prediction accuracy of the Ours-Right is 0.9212, and the accuracy of Ours is 0.9279. Our method does not decrease the variance. In summary, the proposed method improves the accuracy and stability of the entire network model on the small-sample datasets.

F. Discussion

From the experimental results on the LabelMe dataset and the Caltech101 dataset, the proposed method has a better performance compared with other methods. First, our method works better on small-sample datasets. Second, a comparison of the standard deviations and box plots indicates that our method has more stable performance compared to other methods.

The proposed method possesses these advantages for the following reasons, we constructed an ensembling network with two branches, one branch is based on prototype learning, another branch is based on margin learning. The two branches have large variance, and each branch has a high accuracy. So when combining the two branches, the model can achieve better classification performance.

IV. Conclusions

This paper proposes a new ensembling method with two branches, in which one branch is a classifier based on prototype learning, and the other is a classifier based on margin learning. The two branches have larger diversity, and each branch has a high accuracy. The experimental results show that compared with Snapshot, Dropout and VGG16, our method (1) has a better performance on the small-sample dataset of the LableMe and the Caltech101; (2) the model has a better stability.

ACKNOWLEDGEMENT

This work was partly supported by the National Key Research and Development Program of China under Grant 2018YFC0807205, the National Natural Science Foundation of China (NSFC) grant No.61563030, No.61763028, No.61773071, No.61922015 and No.61906080, the Natural Science Foundation of Gansu Province, China, grant No.17JR5RA125, the Beijing Nova Program Interdisciplinary Cooperation Project No. Z181100006218137, Beijing Nova Program No. Z171100001117049 and by the Hong-liu Outstanding Youth Talents Foundation of Lanzhou University of Technology.

REFERENCES

[1] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 1846–1854.

[2] T. Durand, T. Mordan, N. Thome, and M. Cord, "Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[6] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, 2015.

[7] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM review*, vol. 40, no. 3, pp. 636–666, 1998.

[8] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8609–8613.

[9] J. Surh, H.-G. Jeon, Y. Park, S. Im, H. Ha, and I. So Kweon, "Noise robust depth from focus using a ring difference filter," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6328–6337.

[10] Y. Wei, F. Yang, and M. J. Wainwright, "Early stopping for kernel boosting algorithms: A general analysis with localized complexities," in *Advances in Neural Information Processing Systems*, 2017, pp. 6065–6075.

[11] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *2013 International Conference on Machine Learning*, 2013, pp. 1058–1066.

[12] T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1249–1258.

[13] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural computation*, vol. 12, no. 8, pp. 1869–1887, 2000.

[14] M. Moghimi, S. J. Belongie, M. J. Saberian, J. Yang, N. Vasconcelos, and L.-J. Li, "Boosted convolutional neural networks." in *BMVC*, 2016, pp. 24–1.

[15] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[16] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *arXiv preprint arXiv:1702.08835*, 2017.

[17] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[18] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[19] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[20] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[21] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[22] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.