INTERACTIVE BOOSTING OF NEURAL NETWORKS FOR SMALL-SAMPLE IMAGE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks have recently shown excellent performance on numerous classification tasks. These networks often have a large number of parameters and thus require much data to train. When the number of training data points is small, however, a network with high flexibility will quickly overfit the training data, resulting in a large model variance and a poor generalization performance. To address this problem, we propose a new ensemble learning method called InterBoost for small-sample image classification. In the training phase, InterBoost first randomly generates two complementary datasets to train two base networks of the same structure, separately, and then next two complementary datasets for further training the networks are generated through interaction (or information sharing) between the two base networks trained previously. This interactive training process continues iteratively until a stop criterion is met. In the testing phase, the outputs of the two networks are combined to obtain one final score for classification. Detailed analysis of the method is provided for an in-depth understanding of its mechanism.

1 INTRODUCTION

Image classification is an important application of machine learning and data mining. Recent years have witnessed tremendous improvement in large-scale image classification due to the advances of deep learning (Simonyan & Zisserman, 2014; Szegedy et al., 2015; Krizhevsky et al., 2012; Gu et al., 2015). Despite recent breakthroughs in applying deep networks, one persistent challenge is classification with a small number of training data points (Santoro et al., 2016). Small-sample classification is important, not only because humans learn a concept of class without millions or billions of data but also because many kinds of real-world data have a small quantity. Given a small number of training data points, a large network will inevitably encounter the overfitting problem, even when dropout (Srivastava et al., 2014) and weight decay are applied during training (Zhang et al., 2016). This is mainly because a large network represents a large function space, in which many functions can fit a given small-sample dataset, making it difficult to find the underlying true function that is able to generalize well. As a result, a neural network trained with a small number of data points usually exhibits a large variance.

Ensemble learning is one way to reduce the variance. According to bias-variance dilemma (Geman et al., 1992), there is a trade-off between the bias and variance contributions to estimation or classification errors. The variance is reduced when multiple models or ensemble members are trained with different datasets and are combined for decision making, and the effect is more pronounced if ensemble members are accurate and diverse (Granitto et al., 2005).

There exist two classic strategies of ensemble learning (Zhou et al., 2002; Schwenk & Bengio, 1998). The first one is Bagging (Zhou, 2012) and variants thereof. This strategy trains independent classifiers on bootstrap re-samples of training data and then combines classifiers based on some rules, e.g. weighted average. Bagging methods attempt to obtain diversity by bootstrap sampling, i.e. random sampling with replacement. There is no guarantee to find complementary ensemble members and new datasets constructed by bootstrap sampling will contain even fewer data points, which can potentially make the overfitting problem even more severe. The second strategy is Boosting (Schwenk & Bengio, 2000; Moghimi et al., 2016) and its variants. This strategy starts from a classifier trained on the available data and then sequentially trains new member classifiers. Taking

Adaboost (Zhou, 2012) as an example, a classifier in Adaboost is trained according to the training error rates of previous classifiers. Adaboost works well for weak base classifiers. If the base classifier is of high complexity, such as a large neural network, the first base learner will overfit the training data. Consequently, either the Adaboost procedure is stopped or the second classifier has to be trained on data with original weights, i.e. to start from the scratch again, which in no way is able to ensure the diversity of base networks.

In addition, there also exist some "implicit" ensemble methods in the area of neural networks. Dropout (Srivastava et al., 2014), DropConnect (Wan et al., 2013) and Stochastic Depth techniques (Huang et al., 2016) create an ensemble by dropping some hidden nodes, connections (weights) and layers, respectively. Snapshot Ensembling (Huang et al., 2017) is a method that is able to, by training only one time and finding multiple local minima of objective function, get many ensemble members, and then combines these members to get a final decision. Temporal ensembling, a parallel work to Snapshot Ensembling, trains on a single network, but the predictions made on different epochs correspond to an ensemble prediction of multiple sub-networks because of dropout regularization (Laine & Aila, 2017). These works have demonstrated advantages of using an ensemble technique. In these existing "implicit" ensemble methods, however, achieving diversity is left to randomness, making them ineffective for small-sample classification.

Therefore, there is a need for new ensemble learning methods able to train diverse and complementary neural networks for small-sample classification. In this paper, we propose a new ensemble method called InterBoost for training two base neural networks with the same structure. In the method, the original dataset is first re-weighted by two sets of complementary weights. Secondly, the two base neural networks are trained on the two re-weighted datasets, separately. Then we update training data weights according to prediction scores of the two base networks on training data, so there is an interaction between the two base networks during the training process. When base networks are trained interactively with the purpose of deliberately pushing each other in opposite directions, they will be complementary. This process of training network and updating weights is repeated until a stop criterion is met.

In this paper, we present the training and test procedure of the proposed ensemble method and evaluate it on the UIUC-Sports dataset (Li & Fei-Fei, 2007) and the LabelMe dataset (Russell et al., 2008) with a comparison to Bagging, Adaboost, SnapShot Ensembling and other existing methods.

2 THE PROPOSED INTERBOOST METHOD

In this section, we present the proposed method in detail, followed by discussion.

2.1 INITIALIZATION OF COMPLEMENTARY TRAINING DATASETS

We are given a training dataset $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$, d$\in \{1, 2, ..., D\}$, where $\boldsymbol{y}_d$ is the true class label of $\boldsymbol{x}_d$. We assign a weight to the point $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$, which is used for re-weighting the loss of the data point in the loss function of neural network. It is equivalent to changing the distribution of training dataset and thus changing the optimization objective of neural network. We randomly assign a weight $0 < W_{1d} < 1$ to $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$ for training the first base network, and then assign a complementary weight $W_{2d} = 1 - W_{1d}$ to $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$ for training the second base network.

2.2 INTERBOOST TRAINING

The core idea of the InterBoost method is to train two base neural networks interactively. This is in contrast to Boosting, where base networks are typically trained in sequence, namely the subsequent network or learner is trained on a dataset with new data weights that are updated using the error rate performance of the previous base network.

As shown in Figure 1, the procedure contains multiple iterations. It first trains a number of epochs for two base networks using two complementary datasets $\{\boldsymbol{x}_d, \boldsymbol{y}_d, W_{1d}^{(1)}\}$ and $\{\boldsymbol{x}_d, \boldsymbol{y}_d, W_{2d}^{(1)}\}$, $d \in \{1, 2, ..., D\}$, separately, and then iteratively update data weights based on the probabilities that the two base networks classify $\boldsymbol{x}_d$ correctly, namely $P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_1^{(1)})$ and $P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_2^{(1)})$, where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are parameters of the two base networks. During the iterative process, the weights always have the

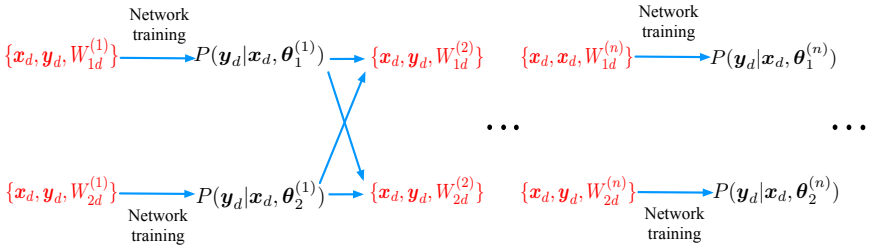

Figure 1: InterBoost training procedure. n is the number of iteration. W_{1d} and W_{2d} are the weights of data point $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$, d$\in \{1, 2, ..., D\}$ for two base networks. $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the parameters of two base neural networks. $W_{1d}^{(n)} + W_{2d}^{(n)} = 1$ and $0 < W_{1d}^{(n)}, W_{2d}^{(n)} < 1$. $P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_i^{(n)}), i \in \{1, 2\}$ is the probability that the ith base network can classify $\boldsymbol{x}_d$ correctly after nth iteration.

constraints $W_{1d} + W_{2d} = 1$ and $0 < W_{1d}, W_{2d} < 1$. That is, they are always kept complementary to ensure the trained networks are complementary. Training networks and updating data weights run alternately until a stop condition is met.

To compute $\boldsymbol{\theta}_1^{(n)}$ and $\boldsymbol{\theta}_2^{(n)}$ in the nth iteration, we minimize weighted loss functions are follows.

$$L_1^{(n)} = \sum_{d=1}^{D} W_{1d}^{(n)} L(\boldsymbol{x}_d, \boldsymbol{y}_d, \boldsymbol{\theta}_1^{(n-1)}) \tag{1}$$

$$L_2^{(n)} = \sum_{d=1}^{D} W_{2d}^{(n)} L(\boldsymbol{x}_d, \boldsymbol{y}_d, \boldsymbol{\theta}_2^{(n-1)}) \tag{2}$$

where $L(\boldsymbol{x}_d, \boldsymbol{y}_d, \boldsymbol{\theta}_1)$ and $L(\boldsymbol{x}_d, \boldsymbol{y}_d, \boldsymbol{\theta}_2)$ are loss functions of of $\boldsymbol{x}_d$ for two base networks, respectively.

To update W_{1d} and W_{2d}, we devise the following updating rule: If the prediction probability of a data point in one base network is higher than that in another, its weight in next iteration for training this network will be smaller than its weight for training another base network. In this way, a base network will be assigned a larger weight for a data point on which it does not perform well. Hence the interaction make it be trained on diverse datasets in sequence, which can be considered as "implicit" Adaboost. Moreover, considering the fact that the two networks are always trained based on loss functions with different data weights, this interaction makes them diverse and complementary.

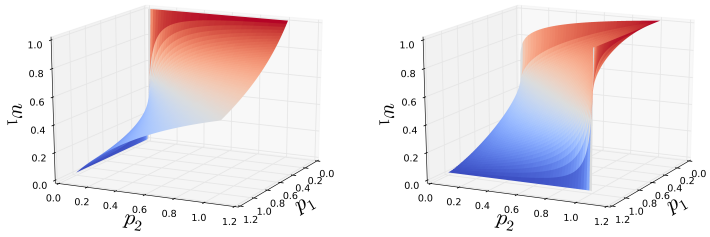

Figure 2: Function $w_1 = p_2/(p_1 + p_2)$ (left) and function $w_1 = \ln(p_1)/(\ln(p_1) + \ln(p_2))$ (right), where $0 < p_1, p_2 < 1$.

To implement the rule of updating data weights, a simple method is to use function $w_1 = p_2/(p_1 + p_2)$, and then assign $W_{1d} = w_1$ and $W_{2d} = 1 - W_{1d}$. Here, for convenience, we use p_1 and p_2 to represent the probabilities that the point $\boldsymbol{x}_d$ is classified by the two base networks correctly. Moreover, this is problematic, as illustrated on the left side of Figure 2. For example, when both p_1 and p_2 are large and close to each other, w_1 will be close to 0.5. In this situation, there will be no big difference between W_{1d} and W_{2d}. In addition, this situation will occur frequently as neural

networks with high flexibility will fit the data well. As a result, the function have difficulty to make a data point have different weights in two base networks.

Instead, we use function $w_1 = \ln(p_1)/(\ln(p_1) + \ln(p_2))$, as shown on the right side of Figure 2, to update data weights. It is observed that the function is more sensitive to the small differences between p_1 and p_2 when they are both large. Specifically, for $\{\boldsymbol{x}_d, \boldsymbol{y}_d\}$, $d \in \{1, 2, ..., D\}$, we update its weights $W_{1d}^{(n)}$ and $W_{2d}^{(n)}$ by Equation (3) and (4).

$$W_{1d}^{(n)} = \frac{\ln P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_1^{(n-1)})}{\ln P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_1^{(n-1)}) + \ln P(\boldsymbol{y}_d|\boldsymbol{x}_d, \boldsymbol{\theta}_2^{(n-1)})} \tag{3}$$

$$W_{2d}^{(n)} = 1 - W_{1d}^{(n)} \tag{4}$$

The training procedure of InterBoost is described in Algorithm 1. First, two base networks are trained by minimizing loss functions L_1 and L_2, respectively. Secondly, weights of data point on training data are recalculated using Equation (3) and (4) on the basis of the prediction results from two base networks. We repeat the two steps until the proposed ensemble network achieves a predefined performance on the validation dataset or the maximum iteration number is reached.

Algorithm 1 InterBoost training procedure

Input:

Training set $\mathbb{X} = \{(\boldsymbol{x}_d, \boldsymbol{y}_d)|d \in \{1, 2, ..., D\}\}$, validation set $\mathbb{V} = \{(\boldsymbol{x}_d, \boldsymbol{y}_d)|d \in \{1, 2, ..., V\}\}$ and maximum number of iterations N.

Steps:

Initialize weights for each data point, $W_{1d}^{(1)}$, $W_{2d}^{(1)}$, and parameters of two base neural networks $\boldsymbol{\theta}_1^{(0)}$ and $\boldsymbol{\theta}_2^{(0)}$. n $\leftarrow 0$, val_acc $\leftarrow 0$.

repeat

n $\leftarrow$ n $+ 1$

Update $\boldsymbol{\theta}_1^{(n)}$ and $\boldsymbol{\theta}_2^{(n)}$ by minimizing (1) and (2)

Update $W_{1d}^{(n+1)}$, $W_{2d}^{(n+1)}$, $d \in \{1, 2, ..., D\}$, according to (3) and (4)

Computing accuracy on $\mathbb{V}$, temp_acc, by (5)

if temp_acc $\geq$ val_acc **then**

val_acc $\leftarrow$ temp_acc

$\boldsymbol{\theta}_1' \leftarrow \boldsymbol{\theta}_1^{(n)}$

$\boldsymbol{\theta}_2' \leftarrow \boldsymbol{\theta}_2^{(n)}$

end if

until val_acc $== 1$ or n $== N$

return Parameters of two base neural networks, $\boldsymbol{\theta}_1'$ and $\boldsymbol{\theta}_2'$

2.3 INTERBOOST PREDICTION

Through the interactive and iterative training process, the two networks are expected to be well trained over various regions of the problem space, represented by the data. In other words, they become "experts" with different knowledge. Therefore, we adopt a simple fusion strategy of linearly combining the prediction results of two networks with equal weights and choose the index class with a maximum prediction value as the final label, as detailed in (5).

$$O(\boldsymbol{x}_{new}) = \arg \max_{c \in \{1,2,...,C\}} \{P\left(c \mid \boldsymbol{x}_{new}, \boldsymbol{\theta}_1\right) + P\left(c \mid \boldsymbol{x}_{new}, \boldsymbol{\theta}_2\right)\}, \tag{5}$$

where $P\left(c \mid \boldsymbol{x}_{new}, \boldsymbol{\theta}_i\right)$, $i \in \{1, 2\}$ is the cth class probability of the unseen data point $\boldsymbol{x}_{new}$ from the ith network, and $O(\boldsymbol{x}_{new})$ is the final classification label of the point $\boldsymbol{x}_{new}$. Because base networks or "experts" have different knowledge, in the event that one base network makes a wrong decision, it is quite possible that another network will correct it.

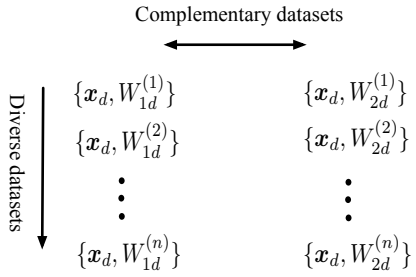

Figure 3: Generated training datasets during the InterBoost training process. The datasets on the left side are for the first base network, and the datasets on the right side are for the second base network.

2.4 DISCUSSION ON INTERBOOST

During the training process, we always keep the constraints $W_{1d}+W_{2d} = 1$ and $0 < W_{1d}, W_{2d} < 1$, to ensure the base networks diverse and complementary. Equation (3) and (4) are designed for updating weights of data points, so that the weight updating rule is sensitive to small differences between prediction probabilities from two base networks to prevent premature training. Furthermore, if the prediction of a data point in one network is more accurate than another network, its weight in next round will be smaller than its weight for another network, thus making the training of individual network on more different regions.

The training process generates many diverse training dataset pairs, as shown in Figure 3. That is, each base network will be trained on these diverse datasets in sequence, which is equivalent to that an "implicit" ensemble is applied on each base network. Therefore, the base network will get more and more accurate during training process. At the same time, the two networks are complementary to each other.

In each iteration, determination of the number of epochs for training base networks is also crucial. If the number is too large, the two base networks will fit training data too well, making it difficult to change data weights of to generate diverse datasets. If it is too small, it is difficult to obtain accurate base classifiers. In experiments, we find that a suitable epoch number in each iteration is the ones that make the classification accuracy of the base network fall in the interval of $(0.9, 0.98)$.

Similar to Bagging and Adaboost, our method has no limitation on the type of neural networks. In addition, it is straightforward to extend the proposed ensemble method for multiple networks, just by keeping $\sum_{i=1}^{H} W_{id} = 1$, $d \in \{1, 2, ..., D\}$, in which H is the number of base networks and $0 < W_{id} < 1$.

3 EXPERIMENTAL RESULTS

3.1 DATASETS AND PREPROCESSING

Considering our focus on small-sample image classification, we choose the following datasets.

- LabelMe datase (LM): A subset of scene classification dataset from (Russell et al., 2008). The dataset contains 8 classes of natural scene images: coast, forest, highway, inside city, mountain, open country, street and tall building. We randomly select 210 images for each class, so the total number of images is 1680.
- UIUC-Sports dataset (UIUC): A 8 class sports events classification dataset [1] from (Li & Fei-Fei, 2007). The dataset contains 8 classes of sports scene images. The total number of images is 1579. The numbers of images for different classes are: bocce (137), polo (182), rowing (250), sailing (190), snowboarding (190), rock climbing (194), croquet (236) and badminton (200).

[1] http://vision.stanford.edu/lijiali/Resources.html

For the LM dataset, we split the whole dataset into three parts: training, validation and test datasets. Both training and test datasets contain 800 data points, in which each class contains 100 data points. The validation dataset contains 8 classes, and each class contains 10 data points.

For the UIUC dataset, we also split the whole dataset into three parts as above. In this dataset, the number of data points in each class, however, is different. We first randomly choose 10 data points for every class to form validation dataset, resulting in 80 data points in total. The remaining parts of the dataset are divided equally into training and test datasets.

For small-sample classification, good discriminative features are crucial. For both LM and UIUC datasets, we first resize the images into the same size of 256×256 and then directly use the VGG16 (Simonyan & Zisserman, 2014) network trained on the ImageNet dataset without any additional tunning, to extract image features. Finally, we only reserve the features of last convolutional layer and simply flatten them. Hence, final feature dimensions for each image is $512 \times 8 \times 8 = 32768$.

3.2 BASE NETWORKS

Considering the small number of data points in the two datasets, we only use fully connected network with two layers. In the first layer, the activation function is Rectified Linear Unit function (***Relu***). In the second layer, the activation function is *Softmax*. We tried different numbers of hidden units, from 1024 to 32, and found overfitting is more severe if the number of hidden units is larger. Finally, we set the number of hidden layer units as 32. We did not adopt the dropout technique, simply because we found there was no difference between with and without dropout, and we set the parameter of L_2 norm about network weights as 0.01. We used minibatch gradient descent to minimize the softmax cross entropy loss. The optimization algorithm is RMSprop, the initial learning rate is 0.001, and the batch size is 32.

3.3 CLASSIFICATION ACCURACIES

In order to evaluate the classification performance of the proposed InterBoost method on LM and UIUC datasets, we use the training, validation and test datasets described above, and compare it with (1) SVM with a polynomial kernel (SVM), (2) Softmax classifier (Bishop, 2006), (3) Fully connected network (FC), (4) Bagging, (5) Adaboost and (6) SnapShot Ensembling (SnapShot) (Huang et al., 2017).

For SnapShot, we adopt the code published by the author of it. For Softmax, FC, Bagging, Adaboost and our method, we implement them based on Keras framework (Chollet, 2015), in which FC is the base network of Bagging, Adaboost, SnapShot and the proposed method. The code of the proposed method and the codes of the other referred methods and the features of two datasets used in the experiment can be found on an anonymous webpage[2] on DropBox .

On the two datasets, we test different epochs ranging from 50 to 800 for Adaboost. It is found that the performance is similar when the epoch number of training base networks is set as 800 and 500. Hence, epoch numbers for FC, Softmax and the base network of bagging are also set as 800 . For SnapShot, we get a snapshot network every 800 epochs, and the number of snapshot is 2. For the base networks of our method, we choose 8 iterations, each iteration has 100 epochs, and thus the total epoch number remains the same as that of Bagging, Adaboost and Snapshot.

SnapShot does not use a validation dataset, so we merge the train and validation datasets into a new training dataset while the test dataset remains unchanged. We run these methods on the two datasets 60 rounds each. The average accuracies are reported in Table 1. Meanwhile, to further evaluate the robustness and stability of the proposed method and other referred methods, we show box plots of accuracies obtained by FC, Bagging, Adaboost, SnapShot and our method on the two datasets in Figure 4.

From Table 1, it can be seen that our method obtains performance with an accuracy of 89.0% on the UIUC dataset, and with an accuracy of 86.4% on the LM dataset. On UIUC dataset, our method performs better than Bagging and Softmax, similarly to FC and worse than Adaboost, Snapshot and SVM. On the LM dataset, our method performs better than FC, Bagging and Softmax, but worse than Adaboost, Snapshot and SVM.

[2]https://goo.gl/9PG3V5

Table 1: Comparison of average accuracies on two datasets: *UIUC-Sports dataset* (UIUC) and *a subset of LabelMe dataset* (LM). Methods include *SVM with a polynomial kernel* , *Softmax classifier*, *Fully connected network* (FC), Bagging, Adaboost, *SnapShot Ensembling* (SnapShot) and *the proposed InterBoost method* (Ours). Each method except for SVM runs 60 rounds, and the mean values of classification accuracies are reported in the Table.

Datasets	**SVM**	**Softmax**	**FC**	**Bagging**	**Adaboost**	**SnapShot**	**Ours**
UIUC	0.897	0.710	0.891	0.872	0.899	0.898	0.890
LM	0.886	0.841	0.860	0.855	0.871	0.880	0.864

In summary, from Table 1 and Figure 4, our method does not have overall superior performance to other baseline methods on both LM and UIUC datasets.

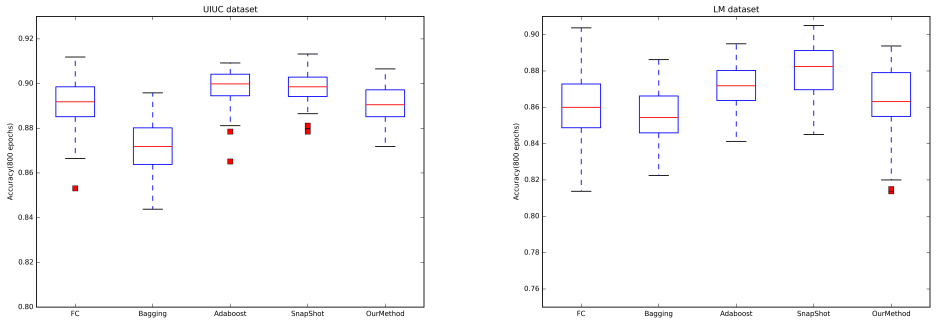

Figure 4: Comparison of accuracies obtained by FC, Bagging, Adaboost, SnapShot and our method via box plot on UIUC and LM datasets. The central mark is the median, and the edges of the box are the 25th and 75th percentiles. The outliers are marked individually. In two boxplots, each method runs 60 rounds. Epoch number of FC and base network in ensemble methods is 800 epochs, in which our method has 8 iterations, and each base network in each iteration is trained 100 epochs.

4 Conclusion

In the paper, we have proposed an ensemble method called InterBoost for training neural networks for small-sample classification and detailed the training and test procedures. In the training procedure, the two base networks share information with each other in order to push each other optimized in different directions. At the same time, each base network is trained on diverse datasets iteratively. Experimental results on UIUC-Sports (UIUC) and LabelMe (LM) datasets showed that our ensemble method does not outperform other ensemble methods. Future work includes improving the proposed method, increasing the number of networks, experimenting on different types of network as well as different kinds of data to evaluate the effectiveness of the InterBoost method.

References

Christopher M.. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

François Chollet. Keras:deep learning library for python. runs on tensorflow, theano, or cntk. *URL https://github.com/fchollet/keras*, 2015.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Pablo M Granitto, Pablo F Verdes, and H Alejandro Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139–162, 2005.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*, 2015.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pp. 646–661. Springer, 2016.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations (ICLR)*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Li-Jia Li and Li Fei-Fei. What, where and who? classifying events by scene and object recognition. In *IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *The British Machine Vision Conference*, 2016.

Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. arxiv preprint. *arXiv preprint arXiv:1605.06065*, 2016.

Holger Schwenk and Yoshua Bengio. Training methods for adaptive boosting of neural networks. In *Advances in Neural Information Processing Systems*, pp. 647–653, 1998.

Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML)*, pp. 1058–1066, 2013.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.