Proceedings of Machine Learning Research 129:769–781, 2020 ACML 2020

PSForest: Improving Deep Forest via Feature Pooling and Error Screening

Shiwen Ni P78083033@GS.NCKU.EDU.TW
Hung-Yu Kao HYKAO@MAIL.NCKU.EDU.TW
Intelligent Knowledge Management Lab, Tainan, Taiwan
Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

In recent years, most of the research on deep learning is based on deep neural networks, which uses the backpropagation algorithm to train parameters of nonlinear layers. Recently, a non-NN style deep model called Deep Forest or gcForest was proposed by Zhou and Feng, which is a deep learning model based on random forests and the training process does not rely on backpropagation. In this paper, we propose PSForest, which can be regarded as a modification of the standard Deep Forest. The main idea for improving the efficiency and performance of the Deep Forest is to do multi-grained pooling of raw features and screening the class vector of each layer based on out-of-bag error. The experiment on different datasets shows that our proposed model achieves predictive accuracy comparable to or better than gcForest, with lower memory requirement and smaller time cost. The study significantly improves the competitiveness of deep forests, further demonstrating that deep learning is more than just deep neural networks.

Keywords: Deep Learning, Deep Forest, non-NN style, Multi-grained pooling, Error Screening

1. Introduction

Currently, the advance of deep learning has demonstrated excellent performance in various applications, particularly in the field of image classification and speech recognition Krizhevsky et al. (2012). Inspired by deep neural networks. Zhou and Feng proposed a deep learning model named deep forest or gcForest Zhou and Feng (2017), which is realized by non-differentiable modules without the backpropagation algorithm. They open a door towards an alternative to deep neural networks.

Essentially, gcForest is a novel decision tree ensemble method with highly competitive to deep neural networks in many machine learning tasks. It combines several ensemble-based methods Dietterich (2000); Zhang and Ma (2012), including random forest and stacking Wolpert (1992). The structure of the deep forest is similar to a multi-layer neural network structure, but each layer in it contains many random forests instead of neurons. Besides, the gcForest is much easier to train because it has much fewer hyper-parameters, and the performance of the model is very robust to the hyper-parameter setting. It has been shown that excellent performance can be achieved by using the default Settings of

hyper-parameter for tasks in different domains. Another critical advantage of gcForest over deep neural networks is that the complexity of its model is not fixed, but automatically determined based on different training data. Therefore, this ensures that gcForest performs well, even on small-scale datasets.

Because of the cascade structure of the deep forest, it has the ability of representational learning. In the cascade structure of the deep forest, each level consists of an ensemble of decision tree forests Breiman (2001); Liu et al. (2008),i.e., an ensemble of ensembles Pang et al. (2018). Each level receives a new set of features as input, which is the output of its preceding level. Each level receives a new set of features as input, which is a combination of its preceding level's output and the original features. For data with sequential and spatial relationships, gcForest further enhances its representational learning ability by a method called multi-grained scanning. Although Zhou and Feng believe that using a larger model may lead to better prediction accuracy, they have not tried a lager model with more grains and larger numbers or forests and trees in each level, which is limited by very high memory requirement and time cost.

We found that multi-grained scanning usually converts the original instance into hundreds or even thousands of new sub-instances, which consumes a lot of memory and time. To address these issues, we proposed a new Deep forest method called PSForest, which improves the deep forest mainly by Feature Pooling and Error Screening. The purpose of Feature Pooling is to extract structural features of data, highlight important features, and reduce the dimension of data features, thus greatly reducing memory and time consumption. In addition, we introduce an Error Screening mechanism in the cascade of deep forest, whit the aim to improve the stability and prediction accuracy of the model. Specifically, the Error Screening filters class vectors generated by each random forest according to the out-of-bagging(OBB) error Matthew et al. (2011). Only the filtered class vectors are used to construct the input feature vectors of the next level.

In a nutshell, we propose an improved the deep forest model called PSForest, which based on Feature Pooling and Error Screening. Our experiments show that PSForest achieves predictive accuracy comparable to or better than gcForest. This is achieved with up to an order of magnitude lower memory requirement and faster runtime. The rest of this paper is organized as follows. A short introduction to deep forest is given in Section 2. Section 3 shows how we accomplish PSForest and introduce the principle of PSForest. In Section 4, we present the experiments and their results. Section 5 provides an overview of the related works. Finally, we conclude in Section 6.

2. Deep Forest

Before considering the PSForest, we introduce gcForest briefly. The gcForest generates a deep forest ensemble, with a cascade structure Zhou and Feng (2017). The gcForest model can be divided into two parts. Representation learning in deep neural networks normally relies on the layer-by-layer processing of raw features. Inspired by this recognition, the first part of the gcForest is a cascade forest structure where each level of a cascade receives feature information processed by its preceding level and outputs its processing result to the next level Zhou and Feng (2017). The second part is the so-called Multi-Grained Scanning, which uses sliding windows of different sizes to scan the raw features. By using multiple

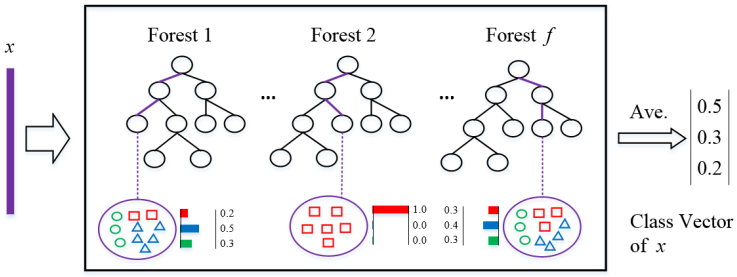

Figure 1: Illustration of class vector generation. (Different marks in leaf nodes imply different classes Zhou and Feng (2017).

sizes of sliding windows, differently grained feature vectors will be generated Zhou and Feng (2017). The representational learning ability of gcForest can be further enhanced by Multi-Grained Scanning when the raw input features are with spatial or sequential relationships.

One of the key ideas underlying the cascade forest structure is a class distribution produced by every tree for each input instance. Given an example, each forest produces an estimate of class distribution by counting the percentage of different classes of examples at the leaf node where the concerned instance falls, as illustrated in Fig. 1, where red color highlights paths along which the instance traverses to leaf nodes.

The usage of the class vector as an output of the random forest classification is highly similar to the idea underlying the stacking method[6]. The stacking algorithm trains the first-level learners using the original training dataset. Then it generates a new dataset for the next-level learners (meta-learners) such that the outputs of the first-level learners are regarded as input features for the second-level learners while the raw labels are still regarded as labels of the new dataset Utkin et al. (2019). In fact, the class vectors in the gcForest can be viewed as the meta-learners. In contrast to the stacking algorithm, the gcForest simultaneously uses the raw feature vector and the class vectors at the next cascade level by means of their concatenation. This means that the feature vector is enlarged and enlarge after every cascade level. After the last level, there are feature representation of the input feature vectors, which can be classified in order to get the final prediction. In addition, each level is an ensemble of decision tree forests, i.e., an ensemble of ensembles. And there are different types of forests to encourage the diversity, as it is well known that diversity is significant for ensemble construction Zhou (2012). In contrast to most deep neural networks whose model complexity is fixed, deep forest adaptively decides its model complexity by terminating training when adequate, which enables it to be applicable to different scales of training data, not limited to large-scale ones Zhou and Feng (2017).

3. The Proposed PSForest Model

The new deep forest model PSForest consists of two parts Multi-Grained Pooling and Cascade-Gate Forest. In this section, we will first introduce the Cascade-Gate Forest, and then the Multi-Grained Pooling, followed by the overall architecture of PSForest.

3.1. Cascade-Gate Forest

The Cascade Forest model provides an alternative to deep neural networks to learn hyper-level representations. Instead of learning hidden variables according to complex forward and backward propagation algorithms in deep neural networks, Cascade Forest tries to learn class distribution vectors directly by assembling amounts of random forest under the supervision of input. Moreover, the Cascade Forest model obviously hopes to acquire more precise class distribution vectors. However, it is well known that a random forest needs to select some features when constructing trees, and the process of selecting features is random. This could lead to the overall prediction performance is sensitive to the quantities of decision trees in each random forest, especially on small-scale data, the final prediction result may not be ideal.

We are Inspired by the gate idea in the DNNs, such as Long Short-Term Memory neural networks (LSTM) Gers et al. (1999). In order to solve the problem and improve the performance of Cascade Forest, we propose a modified version of the cascade forest called Cascade-Gate Forest, which is based on the Error Screening mechanism and stack of class vectors. Rather than requiring all class vectors to go through all levels of the cascade, We are screening the class vectors generated by each random forest according to the out-of-bagging(OBB) error. We propose the cascade-gate structure of the deep forest, as shown in Fig. 2.

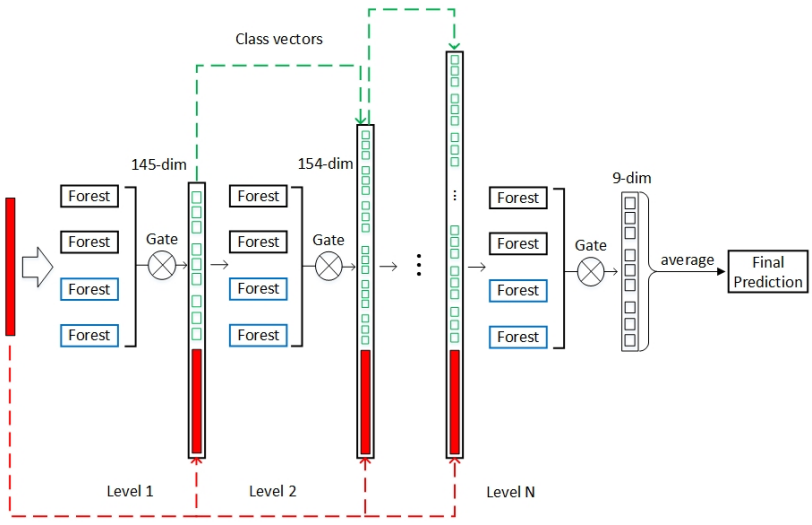

Figure 2: Illustration of Cascade-Gate Forest structure. Suppose each level of the cascade consists of two random forests(black) and two completely-random tree forest (blue). Suppose there are three classes to predict; thus, each forest will output a three-dimensional class vector, which is then concatenated for re-representation of the original input. The class vectors with high OBB error are filtered out at each layer.

Cascade-Gate Forest structure with error screening and stack of class vectors can be defined as $GCF = \{h, f, t, c, s\}$. $h = (1, 2, ..., H)$, h is the index of a layer in the Cascade-Gate Forest, H is for the total number of layers. $f = (1, 2, 3, ...)$, f is the number of forests at each level. The forest at each level includes completely random forest and random forest. $t = (1, 2, 3, ...)$, t is for how many trees each forest contains. $c = (1, 2, 3...)$, c is the number of labels in the sample. $s = (1/f, 2/f, ..., (f-1)/f)$, Note that s is the only additional hyper-parameter compared to the gcForest. It refers to the ratio of the class vector to be filtered out.

In the training stage, each level of cascade forest will generate the class distribution vector of sample x, as shown in formula (1):

$$P^{(t,f)}(x) = (P_1^{(t,f)}(x), P_2^{(t,f)}(x), ..., P_c^{(t,f)}(x)) \tag{1}$$

Here $P^(t, f)$ is the class distribution generated by each decision tree. Then, average the class distribution probability of each tree to get the class distribution vector of each forest.The class distribution vector generated by each forest is

$$V^{(t,f)}(x) = (V_1^{(t,f)}(x), V_2^{(t,f)}(x), ..., V_c^{(t,f)}(x)) \tag{2}$$

The class distribution vector generated for each level is

$$V^{(h,t,f)}(x) = (V^{(h,t,1)}(x), V^{(h,t,2)}(x), ..., V^{(h,t,f)}(x)) \tag{3}$$

Here $V^{(t,f)}$ is of the form:

$$V_c^{(t,f)}(x) = T_f^{-1} \sum_{t=1}^{T_f} P_c^{(t,f)}(x) \tag{4}$$

Then we use error screening mechanism to screen the class vectors. We calculated the out-of-bag error of each forest, then calculated the number of class vectors to be filtered out according to the hyper-parameter s, and finally filtered out the $n(n = s * f)$ class vectors with the maximum OOB error to get the new class vector. The new class vector is set according to the following formulation:

$$V^{(h,t,f,s)}(x) = (V^{(h,t,1,s)}(x), V^{(h,t,2,s)}(x), ..., V^{(h,t,f,s)}(x)) \tag{5}$$

As shown above, the input vectors of each level of the cascade structure of the deep forest is composed of the class vectors of the previous level's output and the initial feature vectors. We can define the input of each level as

$$I^{h,t,f}(x) = (O^{(h-1,t,f)}(x), V_{ini}) \tag{6}$$

Here $I^{h,t,f}(x)$ is the input vectors of level h; $O^{(h-1,t,f)}(x)$ is the output vectors of level $h-1$; and V_{ini} is the initial input feature vectors. Formula (6) is for standard deep forest, with the increasing number of forest layers, the information carried by the feature vector is constantly weakened, which may lead to the fluctuation of the classification result curve. We proposed stack of class vectors method for this problem. As shown in Fig. 2 above, change the input vector of each level of deep forest to a combination of the initial vector and the output class vector of each previous level (The output vector of each level is filtered

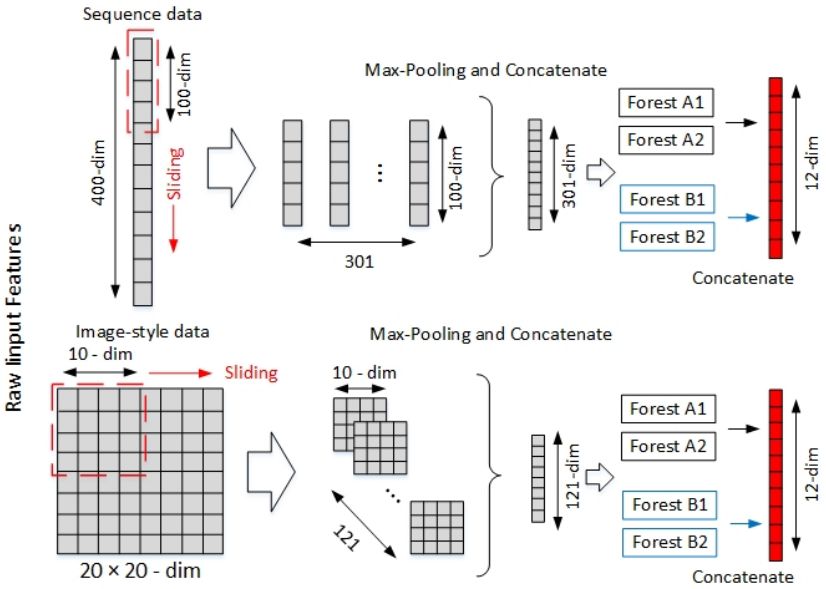

Figure 3: Illustration of Multi-Grained Pooling. (Suppose there are three classes, raw features are 400-dimensional, and sliding window is 100dim, two random forests(black) and two completely-random tree forest (blue).)

by OOB error). More formally, we write the stack of class vectors as follows:

$$I^{h,t,f}(x) = (O^{(1,t,f)}(x), O^{(2,t,f)}(x), ..., O^{(h-1,t,f)}(x), V_{ini}) \tag{7}$$

Finally, the average distribution probability of each forest output is calculated, and then the maximum value is taken as the final prediction result, which is shown below:

$$F = max\{\sum_{1}^{f(1-s)} V_1^{(h,t,f,s)}, \sum_{1}^{f(1-s)} V_2^{(h,t,f,s)}, ..., \sum_{1}^{f(1-s)} V_c^{(h,t,f,s)}\} \tag{8}$$

3.2. Multi-Grained Pooling

It is well known that deep neural networks are powerful in handing feature relationships. For instance, Convolutional neural networks are effective on image data where spatial relationships among the raw pixels are critical LeCun et al. (1998); Hinton et al. (2012); Recurrent neural networks are effective on sequence data where sequential relat-ionships are critical Graves et al. (2013); Cho et al. (2014). Inspired by the pooling layer in the Convolutional neural networks, we enhance Cascade-Gate Forest with a procedure of Multi-Grained Pooling which consumes much less memory and runtime than Multi-Grained scanning in gcForest.

As Fig. 3 illustrates,sliding windows are used to scan the Raw Input Features. Suppose there are 400 raw features and a window size of 100-dimensional features is used. For

sequence data, a 100-dimensional feature vector will be generated by sliding the window for one feature, and We will do max-pooling on this vector(retain the feature with the maximum value) so that the 100-dimensional feature vector will become the 1-dimensional feature vector. If the sliding step size is 1, a total of 301 1-dimensional vectors will be generated.And then, we concatenate 301 1-dimensional vectors and to get a 301-dimensional feature vector. For data with spatial relationships, such as a 20 x 20 panels of 400 image pixels, a 10 x 10 windows will produce 121 10 x 10 dimensional feature vectors. Similarly, we get a 121-dimensional feature vector by max-pooling and concatenation. Then this 301-dimensional or 121-dimensional feature vector will be used to train completely-random forests and random forests, and the class vectors are generated and concatenated as transformed features. As shown in Fig. 3, suppose that there are 3 classes, 4 forest and 100-dimensional windows are used; then 3-dimensional class vector will are produced by each forest, so a 12-dimensional feature vector is generated finally.

The thing to note is that Fig. 3 shows only one size of sliding window, and The feature vector generated by the Multi-Grained Pooling needs to be concatenated with the raw feature vector to from and a new feature vector, which is the input vector of Cascade-Gate Forest. By using multiple sizes of sliding windows, differently grained feature vectors will be generated, as shown in Fig. 4.

3.3. Overall procedure of PSForest

Fig. 4 summarizes the overall procedure of PSForest. Suppose the input is of 100-dimensional data, and three windows sizes are used for Multi-Grained Pooling.For example, there are m instances in training data, a window with size of 100 features will generate a data set of m 91-dimensional training examples.

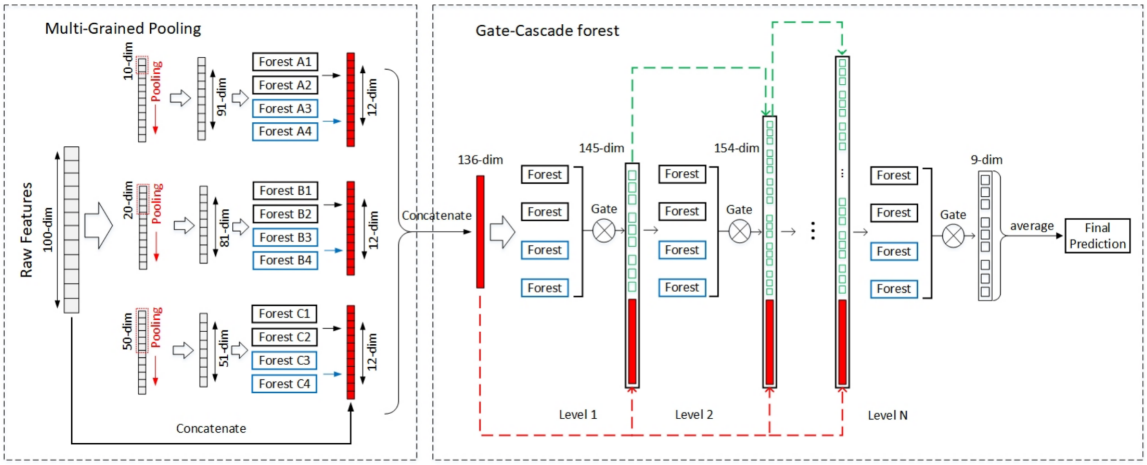

Figure 4: The Over procedure of PSForest. (Suppose there are three class, raw feature are 400-dim,and sliding windows are 10 dim, 20 dim and 30 dim , two random forests(black) and two completely-random tree forest (blue) in each level,and the value of s is 0.25.)

Table 1: Summary of hyper-parameters and default setting. Comparison of the gcForest and PSForest approaches. (d is the dimension of raw input feature)

gcForest	PSForest
Type of forests:	Type of forests:
completely-random forest, random forest	completely-random forest, random forest
Multi-Grained scanning:	Multi-Grained Pooling:
No.Forests: 2	No.Forests: 4
No.Trees in each forest:500	No.Trees in each forest:500
Tree growth:min_samples_split =10	Tree growth:min_samples_split =10
Sliding window size:	Sliding window size:
[d/16], [d/8], [d/4]	[d/16], [d/8], [d/4](MNIST);
Sliding step:	[d/256],[d/128],[d/64],[d/32],[d/16],[d/8]
1(MNIST); 100	Sliding step:
Cascade Forest:	1(MNIST); 100
No.Forests: 8	Cascade-Gate Forest:
Tree growth:min_samples_split =10	No.Forests: 8
	Tree growth:min_samples_split =10
	Filtration rate s : 1/8

These data will be used to train tow completely-random forests and tow random forests, and each forest contains 500 trees. If there are three classes to be predicted, a 12-dimensional class vector will be obtained, as described in Fig. 1. Similarly, sliding windows with sizes of 20 and 50 features will also generate a 12-dimensional class vector, respectively, for each raw training example. The three 12-dimensional class vector generated by the Multi-Grained Pooling will be concatenated with the raw feature vector to from and a 136-dimensional feature vector, which is the raw input vector of Cascade-Gate Forest.Then as shown in Fig. 4, The 136-dimensional primitive vector used to train the four forests of the first level will generate four 3-dimensional class vectors. If the value of s is 0.25, a 3-dimensional class vector will be filtered out, and the remaining 3-dimensional vectors will be combined with the original input vector to form a 145-dimensional vector as the input vector of the second level. All the way to the last layer to get a 9-dimensional class vector, then the average distribution probability of each forest output is calculated, and then the maximum value is taken as the final prediction result.

4. Experiment

The main purpose is to prove that PSForest can achieve predictive accuracy comparable to or better than gcForest with much less memory and time cost. We also did experiments to test the impact of the number of trees on the performance of the model. Specific details of the experiment are as follows:

Table 2: Comparison result (with multi-grained scanning) between gcForest and PSForest on accuracy(%), running time(in seconds) and memory (in megabytes).

Datasets	Method	Accuracy	Running time	Memory
MNIST	PSForest	98.61	106	1783
	gcForest	98.33	755	10232
CIFAR-10	PSForest	44.75	1570	6425
	gcForest	44.70	6109	28820
FASHION	PSForest	87.40	2152	4569
	gcForest	87.89	5868	26491

Table 3: Comparison result (without multi-grained scanning) between gcForest, ImprDF Utkin (2020) and PSForest on accuracy (%), Average is the of all datasets.

Datasets	gcForest	ImprDF	PSForest
Ionosphere	92.42	92.51	**93.16**
Parkinson	91.62	91.62	**93.84**
Yeast	63.45	63.26	**63.48**
Seeds	92.85	**92.94**	**92.94**
Car Evaluation	91.67	91.98	**96.52**
Average	86.40	86.46	**87.98**

Table 4: Impact of the number of trees on accuracy(%), running time(in seconds) and memory (in megabytes).

Trees	500	400	300	200	100
Accuracy	98.61	98.59	98.33	98.21	98.10
Runtime	106	93	77	52	39
memory	1783	1693	1613	1527	1434

Datasets The datasets of this experiment can be divided into two parts: the data set of the first part includes MNIST, CIFAR-10 and FASHION-MNIST, which have sequential or spatial relationship; the other part includes Ionosphere, Parkinson, Yeast, Seeds and Car Evaluation, which have not sequential or spatial relationship. Because the Multi-Grained scanning of gcForest consumes a lot of memory, our computer does not have enough memory, and the MNIST data comes from Sklearn datasets; we chose the data samples of 5000/2000 (train/test) in CIFAR-10 and 20000/10000 (train/test) in FASHION. In addition, the raw features of the three datasets are 64, 784 and 3072, respectively.

Hardware We use a machine with AMD® Ryzen 7 3700x 8-core processor x 16 CPU and 32GB memory.

Parameter Settings In both gcForest or PSForest, the class vector of each forest is generated by three-fold cross-validation. The gcForest and PSForest configurations are

shown in Table 1. Moreover, the number of cascade levels stops increasing when the current level does not improve the accuracy of the previous level for both gcForest and PSForest.And for PSForest, the value of Hyper-parameter s is 1/8. Note that PSForest could adopt more window sizes, which might produce better accuracy. Nevertheless, it is sufficient to use these window sizes for PSForest to achieve satisfactory accuracy.

Evaluation metrics We adopt predictive accuracy as classification performance measurement which is suitable for these balanced datasets. Running time and memory usage are used to evaluate the efficiency.

Results The results with multi-grained scanning, as Table 1 shows that PSForest achieves comparable predictive accuracy than gcForest with less memory and running time. The runtime and memory consumption of PSForest is reduced by several times compared to gcForest, but the performance is more or less the same. For the results without multi-grained scanning, here we conduct the experiment without multi-grained scanning or multi-grained pooling on the datasets that do not hold spatial or sequential relationships among the raw features. We compare the accuracy of gcForest, imprDF and PSForest models, among which imprDF is the latest deep forest variant proposed by Utkin (2020). Since there is no multi-grained scanning in this part of the experiment, it does not need too much memory and time, so we only use the accuracy to prove the performance of PSForest. It should be noted that the accuracy value of imprDF in this paper is directly derived from Utkin (2020). As shown in Table 3, With the exception of Yeast and Seeds datasets, the accuracy of PSForest was higher than that of gcForest and imprDF. Especially in the car Evaluation dataset, PSForest improved by about 5% compared to other models. For average accuracy over the five data sets, PSForest improved by approximately 1.5% over gcForest and imprDF. It can be concluded from the experimental results in Table 3 that the performance of deep forest was improved obviously by the error screening mechanism. As shown in Table 4 (the dataset is the same as the MNIST in Table 2), with the decrease in the number of trees, the running time and memory are significantly reduced, while the decrease in accuracy is not obvious. This implies that our model can reduce a large amount of training time by only consuming a little bit accuracy.

5. Related Work

Following the pioneering work Zhou and Feng (2017), many modifications of the deep forests have been proposed, and the deep forest has been applied to many different tasks. For instance, Guo et al. (2018) propose a modification of the standard deep forest model, so-called BCDForest, to address cancer subtype classification on small-scale biology datasets. They have two main contributions: First, a maned multi-class-grained scanning method is proposed to train multiple binary classifiers to encourage diversity of ensemble. Second, they propose a boosting strategy to emphasize more important features in cascade forests, thus to propagate the benefits of discriminative features among cascade layers to improve the classification performance. Pang and Ting improve deep forest by Confidence Screening Pang et al. (2018) They split the data into easy-to-predict and hard-to-predict at each level of cascade forests. If a sample data is easy-to-predict, its final prediction is produced at the current level; if a sample data is hard-to-predict, it needs to go through the next level, which reduces the number of data passed to the next level and improves the predictive

performance. Utkin and Ryabinin propose a Siamese Deep Forest which can be regarded as an alternative to the well-known Siamese neural networks Utkin and Ryabinin (2018). They modify training set by using concatenated pairs of vectors. In order to deal with the problem of small-scale data, Zhang propose a skip connection deep forest, which uses a skip connection strategy to augment the feature vector and Gradient Boosting Decision Tree as the final classifier to improve the performance Zhang and Zhang (2019). Chen and Li propose a deep forest based predictor for accurate prediction of Self-interacting proteins using protein sequence information Chen et al. (2019). For the classification of ADHD data, Shao and Zhang propose a revised gcForest method that uses a combined multi-grained scanning structure to fuse the two features together, thus a new concatenated feature vector can be formed for each sample Shao et al. (2019). A modification of deep forest for solving classification problems is proposed by Su et al. (2019), and the key idea for improving classification performance of the deep forest is to assign weights to subsets of the lass probability distributions at the leaf nodes computed for every training example. Su et al. (2019) propose a deep cascaded forest model, Deep-Resp-Forest, to classify the anti-cancer drug response as "sensitive" or "resistant". Most recently, Utkin propose an imprecise deep forest(imprDF) classifier Utkin (2020). In the proposed classifier, the precise class probabilities at leaf nodes of decision trees in the deep forest are replaced with interval-valued probabilities produced by Walley's imprecise Dirichlet model Walley (1991, 1996). And the proposed classifier is the first modification of the deep forest, which efficiently deals with small datasets and takes into account a lack of sufficient training data.

6. Conclusion

We propose a more efficient and Deep Forest model called PSForest which has significantly smaller memory requirement and run faster than the standard deep forest proposed by Zhou and Feng. Comparing with deep neural networks, deep forest has much fewer hyper-parameters and its performance is quite robust to hyper-parameter setting. In addition, There is no doubt that tree learners are typically easier to analyze than neural networks. The PSForest retains all the advantages of the standard deep forest. We proposed and used multiple methods such as feature pooling, error screening and stack of class vectors to comprehensively improve deep forest. The experiments show that our new deep forest model achieves higher predictive performance than gcForest, with significantly reduced time cost and memory requirement. Indeed, the important value of this paper lies in the fact that it Greatly promotes research and application of non-NN style deep learning or deep models based on non-differentiable modules. We believe that with the development and deepening of the research, the deep forest will offer an alternative for many tasks when deep neural networks are not superior.

Acknowledgments

We thank anonymous reviewers for their valuable comments and suggestions. Moreover, thanks for senior Jiawen's kind help.

References

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Zhan-Heng Chen, Li-Ping Li, Zhou He, Ji-Ren Zhou, Yangming Li, and Leon Wong. An improved deep forest model for predicting self-interacting proteins from protein sequence using wavelet transformation. *Frontiers in Genetics*, 10, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Yang Guo, Shuhui Liu, Zhanhuai Li, and Xuequn Shang. Bcdforest: a boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data. *BMC bioinformatics*, 19(5):118, 2018.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Fei Tony Liu, Kai Ming Ting, Yang Yu, and Zhi-Hua Zhou. Spectrum of variable-random trees. *Journal of Artificial Intelligence Research*, 32:355–384, 2008.

W Matthew et al. Bias of the random forest out-of-bag (oob) error for certain input parameters. *Open Journal of Statistics*, 2011, 2011.

Ming Pang, Kai-Ming Ting, Peng Zhao, and Zhi-Hua Zhou. Improving deep forest by confidence screening. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1194–1199. IEEE, 2018.

Lizhen Shao, Donghui Zhang, Haipeng Du, and Dongmei Fu. Deep forest in adhd data classification. *IEEE Access*, 7:137913–137919, 2019.

Ran Su, Xinyi Liu, Leyi Wei, and Quan Zou. Deep-resp-forest: A deep forest model to predict anti-cancer drug response. *Methods*, 166:91–102, 2019.

Lev V Utkin. An imprecise deep forest for classification. *Expert Systems with Applications*, 141:112978, 2020.

Lev V Utkin and Mikhail A Ryabinin. A siamese deep forest. *Knowledge-Based Systems*, 139:13–22, 2018.

Lev V Utkin, Maxim S Kovalev, and Anna A Meldo. A deep forest classifier with weights of class probability distribution subsets. *Knowledge-Based Systems*, 173:15–27, 2019.

Peter Walley. Statistical reasoning with imprecise probabilities. 1991.

Peter Walley. Inferences from multinomial data: learning about a bag of marbles. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):3–34, 1996.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

Meiyang Zhang and Zili Zhang. Small-scale data classification based on deep forest. In *International Conference on Knowledge Science, Engineering and Management*, pages 428–439. Springer, 2019.

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

Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*, 2017.