International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-020-01136-0

ORIGINAL ARTICLE

Flat random forest: a new ensemble learning method towards better training efficiency and adaptive model size to deep forest

Peng Liu[1] · Xuekui Wang[2] · Liangfei Yin[3] · Bing Liu[4]

Received: 20 May 2019 / Accepted: 2 May 2020

Abstract
The known deficiencies of deep neural networks include inferior training efficiency, weak parallelization capability, too many hyper-parameters etc. To address these issues, some researchers presented deep forest, a special deep learning model, which achieves some significant improvements but remain poor training efficiency, inflexible model size and weak interpretability. This paper endeavors to solve the issues in a new way. Firstly, deep forest is extended to the densely connected deep forest to enhance the prediction accuracy. Secondly, to perform parallel training with adaptive model size, the flat random forest is proposed by achieving the balance between the width and depth of densely connected deep forest. Finally, two core algorithms are respectively presented for the forward output weights computation and output weights updating. The experimental results show, compared with deep forest, the proposed flat random forest acquires competitive prediction accuracy, higher training efficiency, less hyper-parameters and adaptive model size.

Keywords Ensemble learning · Flat random forest · Training efficiency · Size-adaptive model

1 Introduction

As is known, ensemble learning is an effective learning method to improve the performance of individual models. Typical ensemble learning methods include unsupervised clustering algorithm [1], Bagging [2], Boosting [3], XGBoost [4], Rule aggregation [5] and Random Forest (RF) [6]. Recently, Zhou et al. presented Deep Forest (gcForest) [7] as a counterpart of convolutional neural network (CNN). Compared with CNN, gcForest has some advantages, such as fewer hyper-parameters, shorter training time and lower computation cost. Nevertheless, gcForest still has the following inadequacies:

1. Training Efficiency

In gcForest, the output of each RF layer is input to the next layer, which leads to poor parallelization between different layers during model training. What's more, by means of multi-grained scanning (MGS) [7], gcForest only extracts the locally spatial input features (e.g. sequence data and image) for training each random forest. Thereby, each random forest needs to be trained for EACH scanning window of different sizes, which results in poor training efficiency.

2. Size Adaption

The gcForest adaptively selects the model size by dynamically increasing the depth of model. However, it cannot eliminate the random forests, which contribute little to the output results, and cannot retrain the model accordingly. Thus,

✉ Bing Liu
liubing@cumt.edu.cn

Peng Liu
liupeng@cumt.edu.cn

Xuekui Wang
xuekui.wxk@alibaba-inc.com

Liangfei Yin
13814538110@163.com

[1] National Joint Engineering Laboratory of Internet Applied Technology of Mines, China University of Mining and Technology, Xuzhou 221008, Jiangsu, China

[2] Alibaba Group, Hangzhou 311121, Zhejiang, China

[3] School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, Jiangsu, China

[4] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, Jiangsu, China

Published online: 09 May 2020

this kind of self-adaptive mechanism cannot guarantee that every node of the random forests contributes substantially to the final output. Consequently, it also leads to low training efficiency.

3. Number of Hyper-parameters

The gcForest needs to manually adjust numerous hyper-parameters, including the number of random forests in MGS, the size of sliding window of MGS, the number of random forests in each cascade layer, the number of random trees in each random forest, and so on. Therefore, the number of manually adjusted hyper-parameters is still relatively large, increasing great difficulty for model training.

4. Interpretability

As a deep model, gcForest cannot obtain the contribution of each random forest to the final output after training, thus leading to weak interpretability.

Aiming to solve the abovementioned issues of gcForest, this paper proposes a new ensemble learning method based on RF. First, inspired by densely connected CNN [8], the densely connected deep forest (DCDF) is built by improving the cascade modules of gcForest. Compared with gcForest, DCDF has stronger forward feature-transmission capacity and better performance for large-scale datasets. To further improve the training efficiency, the flat random forest (FRF) is developed by parallelizing DCDF. FRF not only has the ability of learning better representation of the output feature, but also makes parallelization much easier via the trade-off between the depth and width of the model. Compared with gcForest, FRF has the following advantages:

1. *Higher training efficiency* The output weight of FRF can be simply calculated by the proposed forward calculation algorithm. Inspired by [9], the proposed algorithm can update output weights efficiently with the increase of the number of RF. Hence, FRF can be trained efficiently to obtain the final model. Additionally, since the depth of the model is controlled, it is easier to carry out parallelization training for FRF, which enhances the training efficiency considerably, especially for large-scale data. Besides, in this paper, MGS is replaced by principle component analysis (PCA) network (PCANet) [10], which further improves the training efficiency and the classification accuracy.
2. *Size-adaptive model* The proposed FRF can adaptively specify model size through the model training. First, the node number of enhancement-layer can be *dynamically increased* by adding RF nodes. Besides, RF node with small local weight will be replaced by new-trained RF. At last, when the prediction accuracy of the model reaches the goal or it does not rise any more, a size-adaptive model can be obtained.
3. *Less hyper-parameters* Compared with gcForest, the number of hyper-parameters, which need manually fine tuning, is further reduced. For FRF, the hyper-parameters only consist of the number of decision trees in each RF, the node number of the feature-mapping layer and the minimum sample number of the decision tree's leaf nodes. Since the first parameter has little effect upon overall performance, it can be set as an empirical value. Thus, only the rest two parameters need fine tuning for FRF.
4. *Better interpretability* FRF has better interpretability than gcForest based on the average output of each RF. Specifically, the final output only relies on output weights, which makes theoretical analysis easier.

To summarize, the primary contributions of this paper are:

1. Introducing the idea of densely connected CNN to DCDF, which significantly improves the prediction accuracy of gcForest.
2. Utilizing the architecture of functional-link neural networks [11] to increase width and decrease depth of DCDF and proposing a novel ensemble learning method, termed as FRF.
3. Developing two algorithms for training FRF, including forward output weight computation and output weight updating.
4. Integrating PCA Scanning into FRF to further improve the model training efficiency and accuracy.

The remaining paper is organized as follows: Sect. 2 introduces related works of ensemble learning, Sect. 3 provides the details of the flat random forest, Sect. 4 presents experimental results with analysis, and Sect. 5 gives the conclusion.

2 Related works

Conventional ensemble learning mainly includes meta-learning, Rule Ensemble, semi-supervised based ensemble learning, etc. Meta-feature generation methods (meta-learning), such as Stacking [12] and adaptive mixture of experts [13], which use the output of sub-level models as the integrated features to construct advanced model. Rule Ensemble [5] and Bayesian averaging model [14] employ the training set to generate basic models and model combination simultaneously. Refs. [15, 16] study how to select basic models to reach the minimum error rate for an ensemble model. Moreover, there are some studies aiming at using

semi-supervised learning to integrate clustering methods into ensemble classier, including Semiboost [17], Assemble [18] and Try-training [19], etc.

In recent years, many new research directions appear in the field of ensemble learning. Yoon et al. [20] put forward ensemble learning with trees of predictors that generated a different path for each sample. Then the base predictors of each sample were selected from the trees of predictors according to their respective path so as to achieve better performance. Wu et al. [21] generated random forest by different feature subspace selections of minority class and majority class to tackle the problem of imbalanced text categorization. Chakraborty et al. [22] embedded structured random forest and Bayesian Network into a tree structure and designed a dynamic method to decide the depth of tree structure ensemble model. Chakraborty et al. [23] combined clustering and classification for ensemble learning to solve the problem of lack of sufficient manually labeled data. Strauss et al. [24] utilized ensemble learning to solve the problem of deep learning models, which is stated as being highly vulnerable to adversarial perturbations. The above studies improved ensemble learning from various perspectives, including solving the issues that deep learning cannot address.

Inspired by gcForest, this paper proposes FRF, a novel ensemble learning method composed of meta-learning, stacking, and adaptive selection of basic model or combined model. Specifically, FRF leverages stacking architectures and combines the meta-feature generation of the feature-mapping layer with that of the enhancement layers, such as to have satisfactory performance of representation learning and good interpretability. In addition, the proposed node elimination method is adaptive to select and update basic models, while the output weight computing method is efficient to decide the combined model. Finally, the proposed FRF achieves good balance between high efficiency and high classification accuracy, and makes theoretical analysis simpler.

3 Flat random forest

In this section, we first introduce preliminaries that will be useful to understand our model, which mainly cover details of gcForest, the densely connected CNN and functional-link neural network. After that, the densely connected deep forest (DCDF) is presented as the underlying structure of the proposed model. Then, the flat random forest (FRF) is presented by parallelizing DCDF. Finally, PCA Scanning is integrated into FRF to improve the classification accuracy.

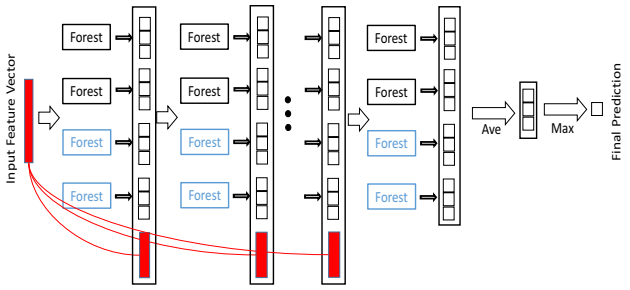

Fig. 1 Cascade module of gcForest

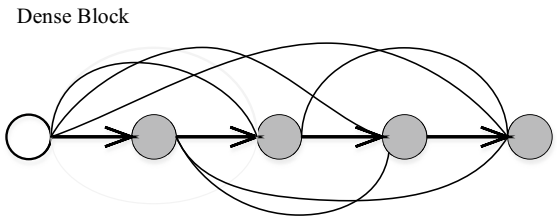

Fig. 2 Dense block in densely connected CNN

3.1 Preliminaries

3.1.1 gcForest

Zhou et al. presented gcForest [7] to overcome the shortcomings of traditional CNN models. As shown in Fig. 1, the output of prior layer in gcForest is just the input of next layer and ONLY the input feature vector can serve as a part of each layer's input. The model training will become more difficult as the depth of gcForest increases.

3.1.2 Densely connected CNN

The major contribution of the densely connected CNN [8] is to put forward Dense Block and apply it to CNN, as shown in Fig. 2. The main idea of Dense Block is to add up all the outputs of prior layers as the input of subsequent neural layers. Motivated by this idea, we apply the dense connection to the construction of Flat Random Forest model. More details are presented in Sect. 3.3.

3.1.3 Functional-link neural network

As illustrated in Fig. 3, the functional-link neural network (FNN) is composed of input layer and enhancement layer, and each layer contains numerous neuron nodes [11]. Thus, FNN leverages the original input and generated features to perform the prediction of the output.

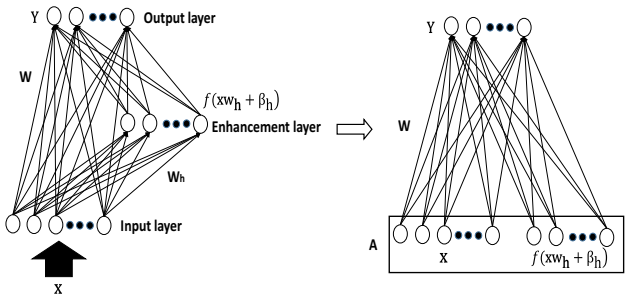

Fig. 3 Functional-link neural network

3.2 Densely connected deep forest

Firstly, combined with the densely connected CNN, gcForest is extended to Densely Connected deep forest (DCDF). As shown in Fig. 4, the outputs of all prior layers in DCDF are concatenated as the input of current layer. Specifically, DCDF is considered as a *widened* gcForest with the depth unchanged. Thus, RF at each level in Fig. 4 is regarded as a whole, and the original model in Fig. 4 can be transformed to the structure in Fig. 5. Notably, these two models are essentially the same.

As illustrated in Fig. 5, if the cascade structure of gcForest, except Level-1, is regarded as ONE node (i.e. Node-N in Fig. 5), DCDF can be viewed as the width extension based on Node-N, i.e., the extended model includes one node (Node-1) in the first layer and N-1 nodes in the second layer. Consequently, DCDF enriches the diversity of feature representations, which contributes to the accuracy improvement of the ensemble model.

From Fig. 5, we can observe that DCDF is actually one special kind of FNN. Node-1 denotes the input layer that is named *feature-mapping layer* in the following part of the paper. The random forest layer composed of Node-2 to Node-N is named as *enhancement layer*. In addition, all enhancement nodes in DCDF, except Node-2, contain more

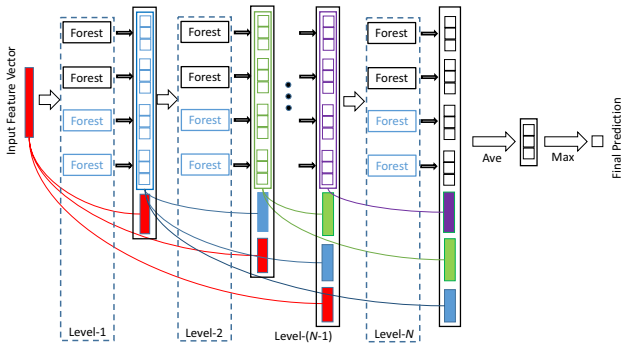

Fig. 4 Densely connected deep forest: gradually wider input of each layer

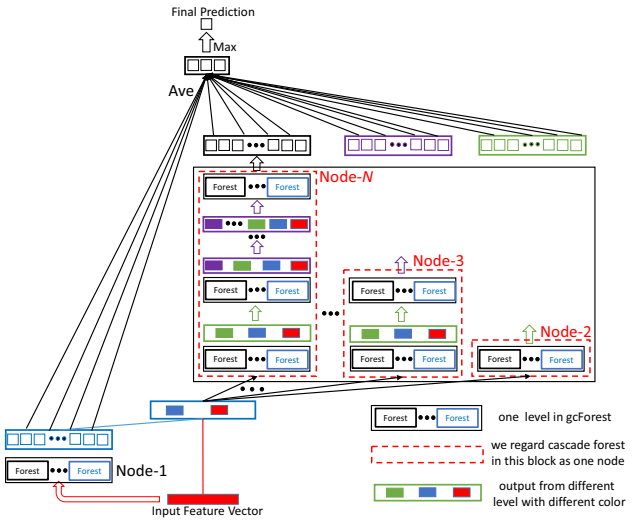

Fig. 5 A flat view of DCDF

than two random forest layers. Nevertheless, it is not hard to find that this multi-layer cascade structure not only increases the difficulty of theoretical analysis but also causes low training efficiency for large-scale data. On one hand, making DCDF wider can improve performance; on the other hand, controlling the depth of DCDF facilitates efficient training and good interpretability. Therefore, the depth and width of DCDF should be balanced to enable the model with better performance, training efficiency and interpretability, which is the original idea of creating FRF.

3.3 System model of FRF

Based on DCDF, the system model of FRF is presented in Fig. 6. As can be seen, Level-1 RF layer, named as the feature-mapping layer (FML), is firstly horizontally expanded. On top of the FML, we design the enhancement layer (EL) that is combined with the FML to form a two-layer RF cascade. Moreover, the output of the FML is concatenated with the input feature vector to serve as the input of the EL, and both outputs are used for the final prediction.

Of note, FRF transfers every neuron node in FNN into RF node, such that it can be regarded as a special kind of FNN. As is well known, it is very important for ensemble algorithms to have model diversity [25]. Therefore, the proposed FRF utilizes a *full-random tree forest* and an ordinary *random forest* to form a composite node to increase model diversity.

Specifically, the feature-mapping layer of FRF is composed of n nodes, shown in Fig. 6, and the input feature vector of this layer is generated by original input data after PCA scanning (as shown in Fig. 9, PCA scanning will be described in Sect. 3.5). Through the feature-mapping layer, a total of n feature mappings can be obtained, which are

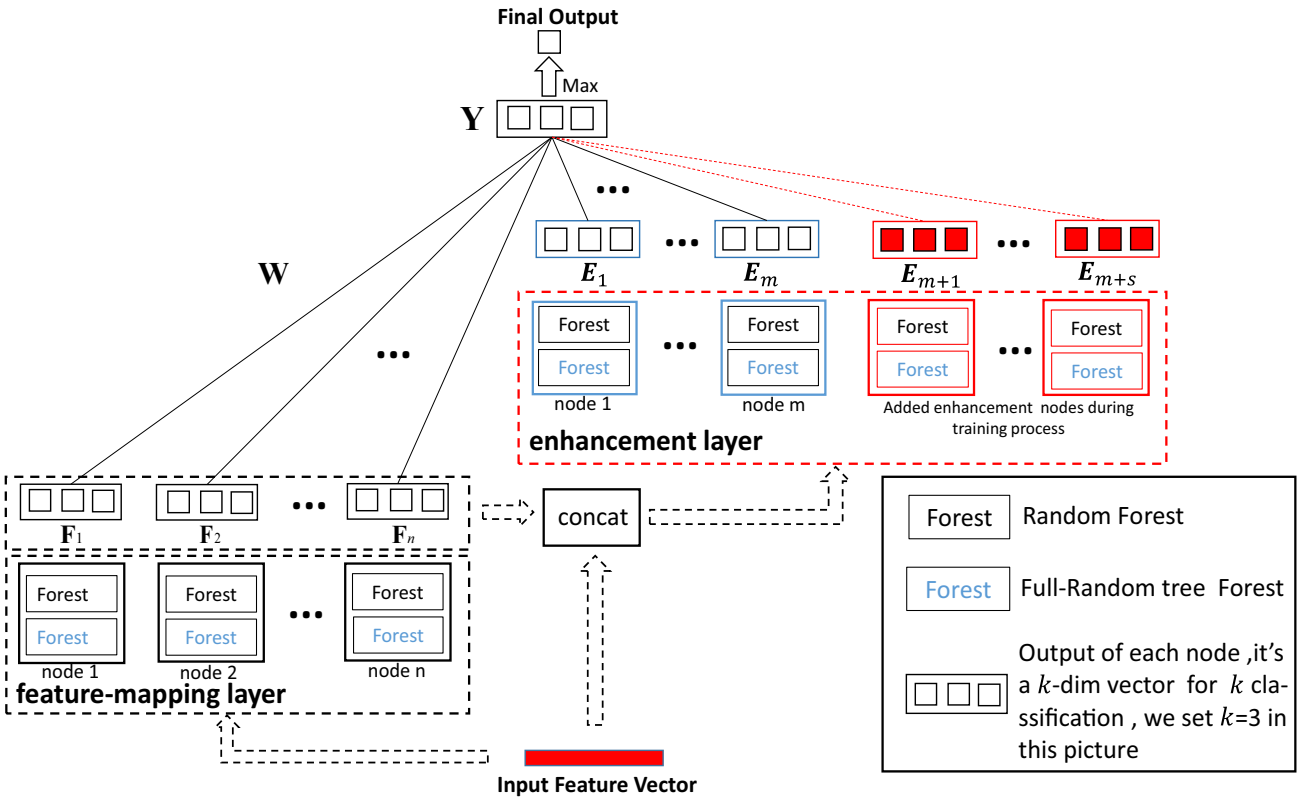

Fig. 6 Flat random forest: a size-adaptive flat model composed of feature-mapping layer and enhancement layer

denoted as $F_1, F_2, \ldots, F_n$. Moreover, each F_i is a k-dim vector (for k classification, $k=3$ in Fig. 6). Then we concatenate the output of feature-mapping layer and the input feature vector to serve as the input of enhancement layer. Next, through the enhancement layer, the outputs can be obtained and denoted as $E_1, E_2, \ldots, E_m, E_{m+1}, \ldots, E_{m+S}$, where m is the initial number of enhancement nodes, s is the dynamically added number of enhancement nodes, and $m+s$ is the final number of enhancement nodes till the end of training process. The output layer uses the weighted sum of all outputs of feature-mapping nodes and enhancement nodes to obtain the final output. Moreover, the output of feature-mapping node and enhancement node are all k-dim vectors (for k classification, $k=3$ in Fig. 6). Each element of the output vector represents the probability of input feature vector belonging to the corresponding class. At last, the final class is the largest element in the final output vector. The computation process of the output for each RF is shown in Fig. 7. It is clear that the output of RF is the average of all decision trees' outputs.

3.4 Training process of FRF

As shown in Fig. 6, the input matrix of output layer is set as the combination of feature-mapping layer and enhancement layer, defined as $\mathbf{A} = [F_1, F_2, \ldots, F_n, E_1, \ldots, E_{m+s}]$, where

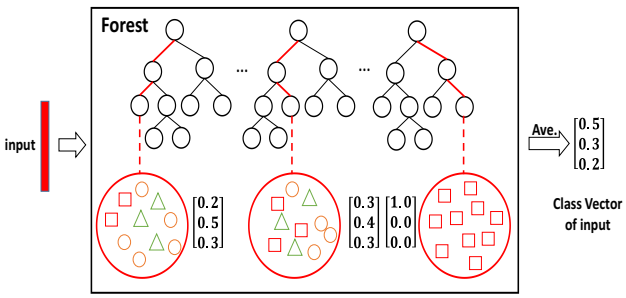

Fig. 7 Computation process of output vector for each RF

$F_1, F_2, \ldots, F_n, E_1, \ldots, E_{m+s}$ are column vectors composed of k elements ($k=3$ in Fig. 6). The output weight vector is defined as $\mathbf{W} = [W_1, W_2, \ldots, W_{n+m+s}]^T$. Hence, the following equation can be obtained:

$$\mathbf{A} \bullet \mathbf{W} = \mathbf{Y} \tag{3-1}$$

where $\mathbf{Y}$ denotes the final output vector of FRF.

The training process of FRF consists of three parts: generation of feature-mapping layer, generation of enhancement layer and computation of output weight. The complete training procedure of FRF is as follows.

Algorithm 1: The Training Procedure of Flat Random Forest

Input: Feature matrix $\mathbf{V}=[\mathbf{V}_1,\mathbf{V}_2,\cdots,\mathbf{V}_N]$ which is produced by PCA scanning (as shown in Fig. 9, PCA scanning is detailed in section3.5). $\mathbf{V}_i(i=1,\cdots N)$ is the feature vector of the i^{th} sample. N denotes the number of training samples.

Output: Weight matrix $\mathbf{W}$ and FRF model

Step 1 **Generation of Feature-Mapping layer**
Set $\mathbf{V}_i(i=1,\cdots N)$ as the input feature vector to generate a feature-mapping layer including n nodes. Then calculate each node's local weight $\omega_i, i=1,2,\cdots,n$. Lastly, regenerate the RF for nodes with the local weight less than $1/k$ (as for k classifications).

Step 2 **Generation of Enhancement layer**
Combine both the output of feature-mapping layer with the input feature vector to form the new feature vector corresponding to m enhancement nodes. Then calculate the local weight of each node. Regenerate the RF for nodes with the local weight less than $1/k$.

Step 3 **Calculation of output weight**
Calculate the output weight $\mathbf{W}$ based on the local weight of each node.

Step 4 **Width increase of Enhancement layer**
Perform step 2 to add s enhancement nodes. Update $\mathbf{W}$.

Step 5 **Determine whether the algorithm is terminated**
Calculate $\mathbf{A}$ of the output layer for each sample. Then calculate the final output vector of each sample by Eq. 3-1 and the final prediction accuracy q. If q does not increase anymore, stop the training, else, go to step 4.

In Algorithm 1, the process of calculation and update for $\mathbf{W}$ is as follows.

First, denote the output matrix of sample set in this node as $\mathbf{O}=[\mathbf{O}_1,\mathbf{O}_2,\ldots,\mathbf{O}_N]^T$, $\mathbf{O}_i=[O_{i1},O_{i2},\ldots,O_{ik}]^T$, $i=1,2,\ldots N$, where O_{ip} denotes the probability of the ith sample belonging to the pth category predicted by this node. For the sample set, its corresponding target category matrix is $\mathbf{C}=[\mathbf{C}_1,\mathbf{C}_2,\ldots,\mathbf{C}_N]$, $\mathbf{C}_i=[c_{i1},c_{i2},\ldots,c_{ik}]^T$, $i=1,2,\ldots N$ (only if the ith sample belongs to the pth category, then $c_{ip}=1$, else, $c_{ip}=0$). Then the local weight is calculated as Eq. 3-2.

$$\omega=\frac{\sum_{i=1}^{N}\mathbf{O}_i^T\bullet\mathbf{C}_i}{N} \tag{3-2}$$

where $\mathbf{O}_i^T\bullet\mathbf{C}_i$ denotes the probability of the ith sample belonging to the target category predicted by this node. To calculate the average probability on the training set, vectorization calculation is employed here. In Eq. 3-2, let $\mathbf{H}=\mathbf{O}\bullet\mathbf{C}$, Eq. 3-3 will be obtained.

$$\omega=\frac{\sum_{i=1}^{N}\boldsymbol{O}_i^T\cdot\boldsymbol{C}_i}{N}=\frac{\sum_{i=1}^{N}\mathbf{H}_{ii}}{N} \tag{3-3}$$

Next, the output weight vector $\mathbf{W}=[W_1,W_2,\ldots,W_{n+m}]^T$ can be calculated through local weights as follows.

$$W_i=\frac{\omega_i}{\sum_{i=1}^{n+m}\omega_i},i\quad=1,2,\ldots n+m \tag{3-4}$$

For new enhancement nodes, Eq. 3-4 shows that each W_i only needs to multiply a coefficient. Denote the number of additional enhancement nodes as s, each node's output weight can be updated as follows.

$$W_i^t=\begin{cases}\dfrac{\omega_i}{\sum_{i=1}^{n+m+s}\omega_i}=\dfrac{\omega_i}{\sum_{i=1}^{n+m}\omega_i}\times\dfrac{\sum_{i=1}^{n+m}\omega_i}{\sum_{i=1}^{n+m+s}\omega_i}=W_i^{t-1}\times\dfrac{\sum_{i=1}^{n+m}\omega_i}{\sum_{i=1}^{n+m+s}\omega_i}, & i=1,2,\ldots n+m\\ \dfrac{\omega_i}{\sum_{i=1}^{n+m+s}\omega_i}, & i=n+m+1,\ldots,n+m+s\end{cases} \tag{3-5}$$

From the discussion mentioned above, it can be seen that the training process of FRF only involves forward operations. In addition, the elimination mechanism and local weights are introduced to guarantee all RFs contribute to the final prediction vector. Moreover, by different values of output weights, the contribution of each node to the final prediction vector can be measured directly. Thus, the theoretical analysis of FRF becomes relatively easier.

3.5 PCA scanning

The gcForest [7] adopts Multi-grained scanning (MGS), as shown in Fig. 8, to extract one-dimensional feature vector that is then used as the input of cascade module. However, through the experiments, it is discovered that this scanning method requires high expense of CPU and memory and is time-consuming meanwhile. In this paper, inspired by PCANet, as shown in Fig. 9, we adopt PCA scanning instead of MGS. The experiment demonstrates the scanning efficiency greatly rises while the model accuracy even further improved by PCA scanning. More details about PCA scanning will be discussed in Sect. 3.6.

3.6 Combination of PCA scanning and FRF

The overall running procedure of proposed model is actually divided into two parts, as shown in Fig. 10, named as PCA scanning and Flat Random Forest (FRF) training.

Suppose we input N training images sized at 32×32 and the size of scanning window is $k_1\times k_2$. Through padding and scanning, $32\times32\times N$ vectors can be generated with dimension of $k_1\times k_2$. Then PCA scanning will be performed after averaging removed to extract L_1 principal component feature

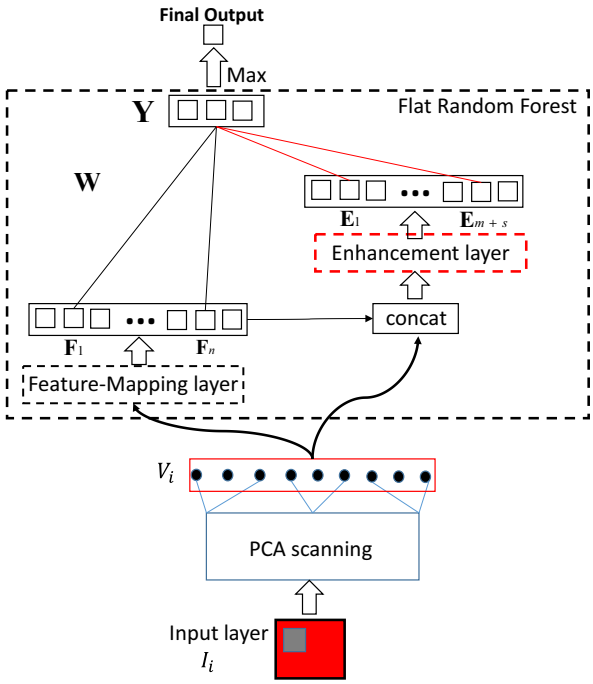

Fig. 10 The pipeline model for classification

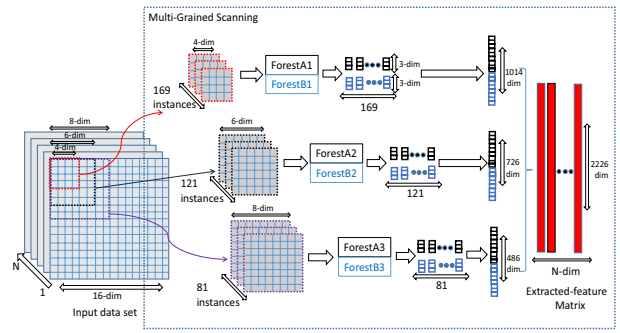

Fig. 8 Multi-grained scanning

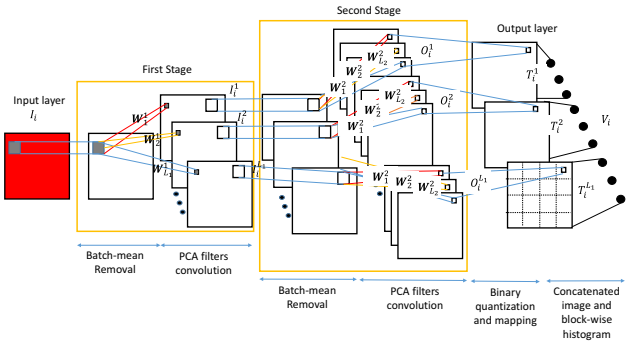

Fig. 9 PCA scanning for FRF: more efficient feature extraction method

vectors via the parameter of L_1 convolution kernels in the first stage (the size of convolution kernel is $k_1\times k_2$). Thus, the original input image is transformed to $N\times L_1$ images of size 32×32 by the first-stage scanning. Moreover, based on the first-stage scanning, the second-stage scanning conducts the same operation to obtain L_2 principal component feature vectors as the parameter of second-stage convolution kernel (the size of convolution kernel is $k_1\times k_2$). The second convolutional layer outputs $N\times L_1\times L_2$ images of size 32×32, which are divided into N groups (each original image corresponds to one group). Then conduct hashing and histogram statistics for each group to obtain the feature vector output with the length of $2^{L_2}*L_1*B$ (where B denotes the block number when partitioning each image to carry out hashing and histogram statistics, which is not detailed in this part. Please refer to PCANet [10]). Thus, after PCA scanning, for N input images, N feature vectors with length of $2^{L_2}*L_1*B$ are obtained.

After PCA scanning, the obtained feature vector and the original input label are used to train the RFs to generate the *feature-mapping layer*. Actually, the feature-mapping layer contains n nodes, and to enhance diversity of ensemble algorithms, each node is composed of a completely random tree forest and an ordinary RF. The output of each node is the mean value of two random forests' output. Through the feature-mapping layer, each training sample can obtain $\mathbf{F}=[\mathbf{F}_1,\mathbf{F}_2,\ldots,\mathbf{F}_n]$, in which $\mathbf{F}_i$ is a k-dim prediction vector (in Fig. 6, $k=3$). Then $\mathbf{F}$ and the PCA-obtained feature

vector are concatenated into the new feature (the labels still keep the same) to generate the *enhancement layer* of m enhancement nodes. Similarly, each node is composed of a completely random tree forest and an ordinary random forest, and the output of each enhancement node is the mean value of two random forests' outputs. Through the enhancement layer, each training sample can obtain m instances of k-dim prediction vector. Next, as mentioned in Fig. 6, the proposed training method is adopted to increase the number of enhancement node dynamically until the end of training. Finally, the output weight of each node is obtained.

At the FRF testing phase, each input sample passes through the PCA scanning, feature-mapping layer and enhancement layer to generate the input of output layer, i.e. $\mathbf{A} = [\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_n, \mathbf{E}_1, \ldots, \mathbf{E}_{m+s}]$ (given that the training process ends with $m+s$ enhancement nodes). Then the inner product of $\mathbf{A}$ and $\mathbf{W}$ can be easily computed to obtain the final category prediction $\mathbf{Y}$.

Lastly, we focus on the FRF's hyper-parameters, including the size of PCA scanning window, the number of decision trees contained in each RF, the minimum sample number of decision tree's leaf nodes and the node number of the feature-mapping layer. Among the four hyper-parameters, the former two can be determined empirically and the latter two need manual fine-tuning. More detailed discussions about hyper-parameters will be given in the experiments.

4 Experiments

In this section, to validate the effectiveness of the proposed method, we compare FRF with DCDF, gcForest and some other models on different datasets. Furthermore, the training efficiency of FRF and other methods are compared by using the same ratio of training data to testing data. Finally, three experiments are designed to study the influence of hyper-parameters on FRF.

4.1 Configuration

Firstly, as shown in Table 1, the common hyper-parameters of gcForest, DCDF and FRF in the experiments are listed as follows.

Secondly, the detailed hyper-parameters for FRF only are shown as follows:

1. For PCA scanning, the corresponding hyper-parameters are set as $k_1 = k_2 = 5$, $L_1 = L_2 = 8$, $B = (H/8) \times (W/8)$ (refer to Sect. 3.6) while those of MGS are set just the same as gcForest.
2. For the feature-mapping layer, the node number is 8, and each node consists of a completely random tree forest and a RF. Each RF contains 200 decision trees and the minimum sample number of decision tree's leaf nodes is 40.
3. For the enhancement layer, correspondingly, each RF includes 200 decision trees and the minimum sample number of decision tree's leaf node is 40.

Next, for other machine learning methods, we mainly refer to the experiment data of gcForest reported in [7], and the relevant hyper-parameters are specified.

Finally, we divide the training sample set into three parts in the proportion of 60%, 20% and 20%, respectively. Then, 60% training data is utilized to generate RF. Then 80% (60% plus 20%) data is used to calculate the local weight of each node. Lastly, the left 20% is adopted to carry out accuracy test. When the accuracy reaches the expected value or the accuracy does not increase any more, the model training process is terminated.

Table 1 Summary of common hyper-parameters

GcForest	Densely-connected deep forest	Flat random forest
Type of forests	*Type of forests*	*Type of forests*
Complete random tree forest, random forest, etc.	Complete random tree forest, random forest, etc.	Complete random tree forest, random forest, etc.
Forest in multi-grained scanning	*Forest in multi-grained scanning*	*Forest in feature Mapping layer*
No.Forests:{2}	No.Forests:{2}	No.Forests:{8}
No.trees in each Forest:{30}	No.trees in each Forest:{30}	No.trees in each Forest:{200}
Tree growth: till pure leaf, or $\leq$20 instances	Tree growth: till pure leaf, or $\leq$20 instances	Tree growth: till pure leaf, or $\leq$40 instances
Sliding window size: $\{\lfloor d/16 \rfloor, \lfloor d/9 \rfloor, \lfloor d/4 \rfloor\}$	Sliding window size: $\{\lfloor d/16 \rfloor, \lfloor d/9 \rfloor, \lfloor d/4 \rfloor\}$	
Forest in cascade	*Forest in cascade*	*Forest in Enhancement layer*
No.Forests:{4}	No.Forests:{4}	No.Forests: self-adaption during training process
No.trees in each Forest:{1000}	No.trees in each Forest:{1000}	No.trees in each Forest:{300}
Tree growth: till pure leaf, or $\leq$10 instances	Tree growth: till pure leaf, or $\leq$10 instances	Tree growth: till pure leaf, or $\leq$40 instances

Table 2 Prediction accuracies on various image data sets

	Digits (%)	MNIST	Cifar10 (%)	Cifar100 (%)
Flat random forest (PCA scanning)	**99.81**	99.36%	**89.01**	**74.32**
Flat random forest (multi-grained scanning)	99.75	99.01%	71.52	62.09
Densely-connected deep forest	99.73	99.02%	68.55	59.14
gcForest	99.69	98.96% [7]	63.82	51.00
BRF	98.98	98.75%	65.12	58.37
EC3	99.46	**99.61%**	70.36	67.45
DBN [13]	99.31	98.75%	62.27	55.77
DNN (Le-Net5)	99.77	99.05% [7]	80.59	68.73
Random forest	98.73	96.80% [7]	50.23	32.25

Bold indicates the best results

Table 3 Prediction accuracies on ORL face data set

	1 image	5 images	9 images
Flat random forest (PCA scanning)	66.94%	**96.95%**	**99.30%**
Flat random forest (multi-grained scanning)	66.38%	96.10%	98.50%
Densely-connected deep forest	65.83%	95.05%	98.60%
gcForest	63.06% [7]	94.25% [7]	98.30% [7]
BRF	66.48%	92.79%	98.18%
EC3	**68.16%**	96.32%	99.08%
Random forest	61.70% [7]	91.20% [7]	97.00% [7]
DNN (CNN)	3.30% [7]	86.50% [7]	92.50% [7]
SVM (RBF kernel)	57.90% [7]	78.95% [7]	82.50% [7]

Bold indicates the best results

4.2 Experimental results

4.2.1 Accuracy

4.2.1.1 Image classification The following four kinds of data sets are employed for image classification. Digits data set is composed of 1797 samples, where each sample is a handwriting of 0–9. MNIST is also a handwriting recognition data set sized at 28×28 that includes 60,000 training samples and 10,000 test samples. Cifar10 data set contains 50,000 training images sized at 32×32 and 10,000 test images of the same size, in which the number of object category is 10. Moreover, for Cifar100 data set, the number and size of training images and test images are the same as Cifar10 except that it contains 100 object categories. In the experiment, we use these four kinds of data sets to test RF, gcForest, DCDF and FRF. Meanwhile, the experiment results of LeNet-5 in gcForest [7], EC3 [23], DBN [26] and BRF [27] are also compared together as shown in Table 2.

From Table 2, it can be observed that the accuracy of DCDF is 1.94% higher than that of gcForest on average, which means DCDF, with dense connection, has stronger feature representation ability than gcForest. Most importantly, compared with DCDF and LeNet-5, FRF achieves more competitive accuracy (11.5% higher than DCDF and 3.34% higher than LeNet-5 on average), which proves effectiveness of the transformation from depth to width. In

Table 4 Small-scale classification data sets

	Instances	Features	Classes
Iris	150	4	3
Letter	20,000	16	26
Adult	48,842	14	2
Yeast	1484	8	10
Zoo	101	17	7
Breast	106	10	6
Echocardiogram	132	12	2
Wine	178	13	3
Vertebral	310	6	3
Cvr	435	16	2
Bands	512	39	2
WDBC	569	32	2
Land-cover	675	148	9
Credit	690	15	2
Transfusion	748	5	2
Vehicle	946	18	4
Mammographic	961	6	2
CMC	1473	9	3
Car	1728	6	4
Image	2310	19	7
Madelon	2600	500	2
Chess	3196	36	2
ADS	3279	1558	2
Abalone	4177	8	29

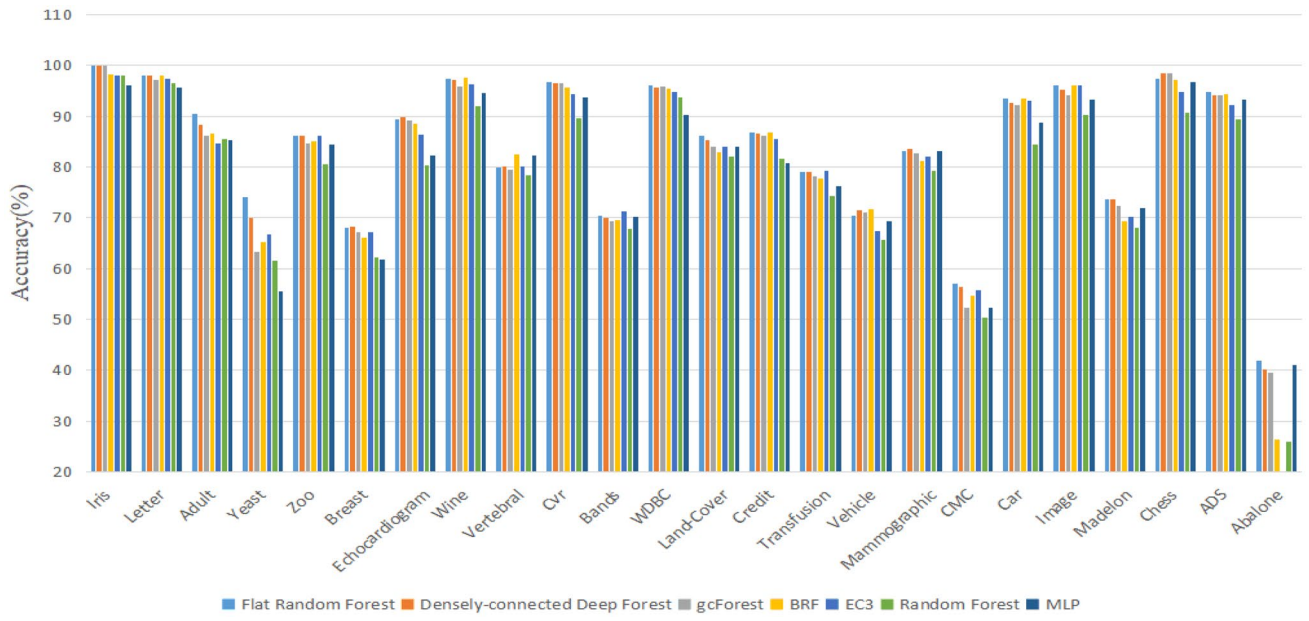

Fig. 11 Prediction accuracies on small-scale data sets

addition, for FRF, compared with MGS, the accuracy is further improved based on PCA scanning.

4.2.1.2 Face recognition ORL data set [28] consists of faces of 40 people of different ages, genders and races. Each person has 10 images sized at 92×112, and there are 400 grayscale images in total with black background. Each person's facial expression is different, such as smiling or not, eyes opening or closing, wearing or not wearing glasses, etc. The face pose is different too, of which the depth and plane rotation can both reach 20 degree. Moreover, the maximum variation of face size can be 10%. For ORL data set, we compare the test results of random forest, gcForest, SVM, DCDF and FRF (with PCA scanning and MGS respectively). In the experiment, for each object, this paper selects 1, 5 and 9 face images to conduct model training, while the remaining images are used to test. After careful fine-tuning of hyper-parameters, the node number of feature-mapping layer in FRF is set as 15. Table 3 gives the test results.

As can be seen from Table 3, EC3 displays a superior test accuracy when the training data are relatively insufficient. This is mainly because that EC3 has a stronger learning ability when dealing with insufficient manually labeled data. However, with sufficient training data, the proposed FRF is superior to all the other methods.

4.2.1.3 Small-scale data sets Classification experiments are further carried out on twenty-four small-scale data sets. The results are shown in Table 4 [29] and displayed in Fig. 11, respectively. In Fig. 11, the X-axis represents various models on various data sets and the Y-axis denotes prediction accuracies (percentage).

As shown in Table 4 and Fig. 11, compared with gcForest, DCDF displays a slight rise of accuracy rate in most datasets (a rise by 0.13% on average), and shows more superiority over other competitive algorithms. The accuracy rate of FRF is quite similar to that of DCDF in these small-scale data sets, which demonstrates the accuracy does not get worse by decreasing the model depth while increasing the model width.

4.2.2 Training efficiency

To validate the high training efficiency of FRF, we compare the running time of FRF, BRF, EC3, DNN, MLP and gcForest on the same data sets (detailed in the next paragraph), and on the same computing resources as GPU GTX1080ti X 2 with 22G video memory, CPU Intel E5-2640 v4, Memory 32G and Hard Disk 1 TB.

Six image datasets and two small non-image datasets are selected for this experiment. The six image datasets include Digits, Minist, Cifar10, Cifar100, Fer and Adience; the non-image datasets include Adult dataset and Letter dataset. Hereinto, compared with the preceding experiments, this section adds two new data sets, Fer and Adience, where Fer is a Facial Expression Recognition data set with 35,888 pictures and Adience has 26,580 human images with distributed ages between 0 and 60. During the experiment, whenever the accuracy variation tends to be stable, the training time and the accuracy rate are recorded respectively, as shown in Fig. 12.

As can be seen from Fig. 12, in both large datasets and small datasets, while maintaining competitive accuracy

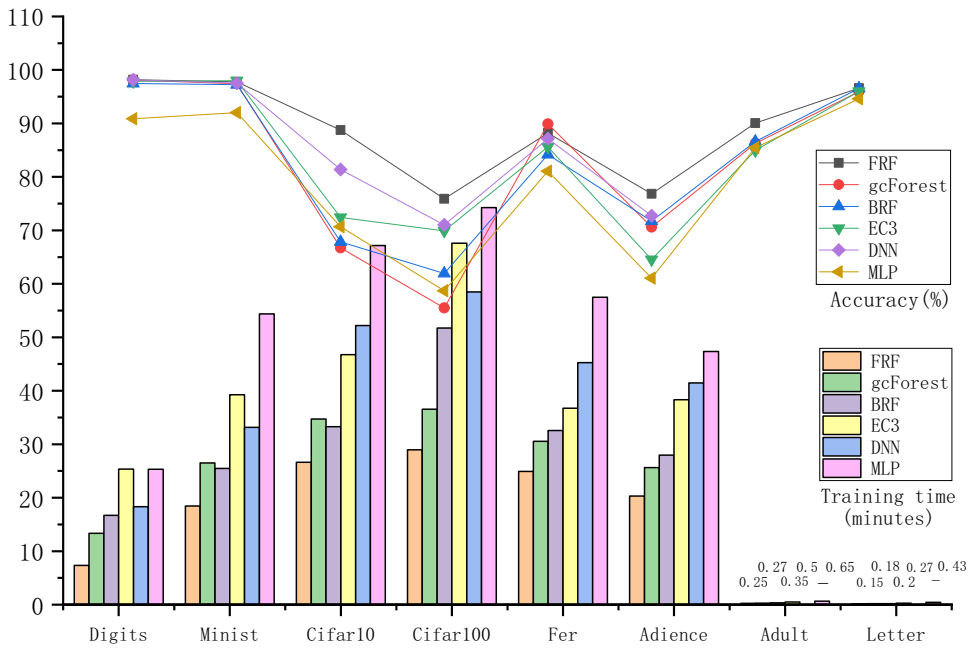

Fig. 12 Training efficiency with accuracy on various data sets

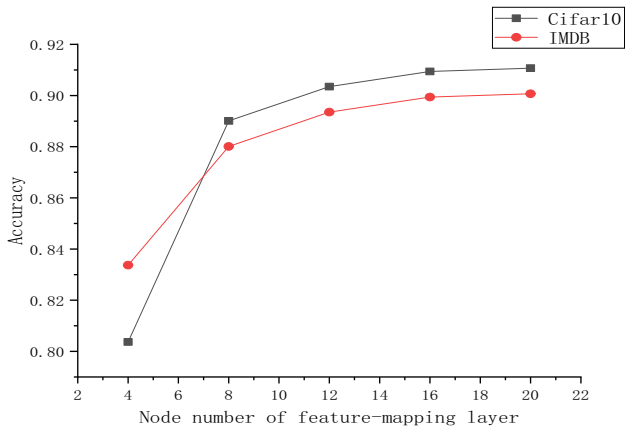

Fig. 13 Accuracy influence by node number of feature-mapping layer

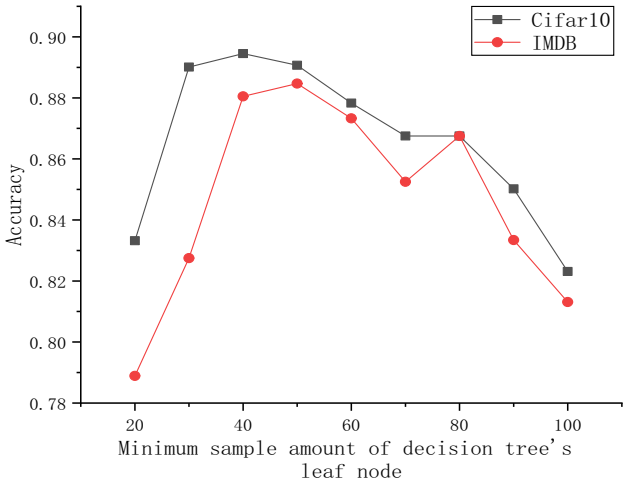

Fig. 14 Accuracy influence by minimum sample amount of decision tree's leaf node

rates, FRF demonstrates a sensible advantage over other methods in running time. Specifically, compared with gcForest, the running time of FRF has been shortened by 24.3% on average. Compared with that of other algorithms, the running time of FRF is much faster. Therefore, it can be concluded that FRF model has excellent scalability in terms of running time. It is noted that the accuracies of FRF on Cifar10 and Cifar100 are significantly better than that of gcForest, due to the advantage of PCANet (for FRF) over MGS (for gcForest) on large-scale image data sets.

4.2.3 Influence of hyper-parameters

Three experiments are conducted to study the influence of hyper-parameters on FRF, including node number in feature-mapping layer, minimum samples of decision tree's leaf nodes and number of decision trees in RF, respectively. For each experiment, cifar10 and IMDB data set are employed to obtain two curves of accuracy variation. See Figs. 13, 14 and 15.

As shown in Fig. 13, when the node number of feature-mapping layer is not large enough, the accuracy rate can be improved by increasing the node number. However, when the node number reaches a threshold value, the improvement of accuracy rate is no longer obvious. Therefore, during the experiments, it is suggested that the node number of feature-mapping layer should keep increasing until the performance becomes steady.

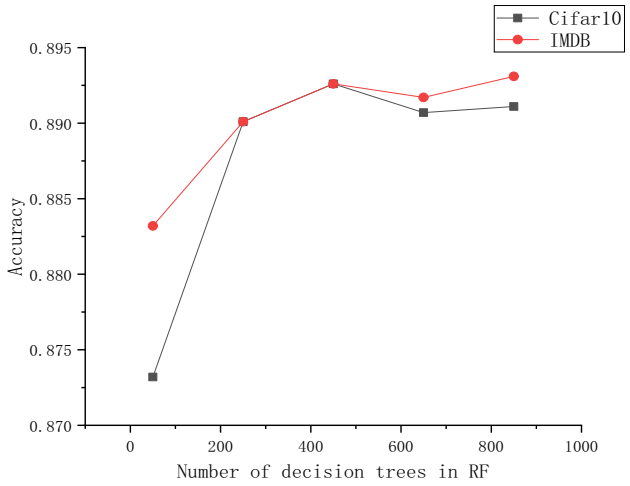

Fig. 15 Accuracy influence by number of decision trees in RF

We can observe in Fig. 14 that the minimum sample amount of decision tree's leaf node should be in appropriate range. If it is smaller than 30, the model will tends to overfit. On the contrary, if it is larger than 80, under-fitting will occur to the model. Thus, it is necessary to adjust this parameter for different data sets to appropriate ranges, respectively.

As illustrated in Fig. 15, the generalization ability of the model is poor when the number of decision trees in RF is smaller than 200. Hence, the final accuracy rate is low. When the number surpasses a threshold (≥ 200), the final accuracy rate tends to be relatively stable. Thus, based on our experiment settings, the number of decision trees in RF should not be smaller than 200.

5 Conclusion

In summary, this paper proposes a novel ensemble-learning model to overcome the shortcomings of gcForest. By virtue of dense connection and FNN, the proposed Flat Random Forest model has the ability of adaptively learning the model size, which enables the balance between the model depth and width. Experimental results show that compared with other popular methods our method achieves sensibly higher training efficiency while maintaining competitive accuracy.

In the process of applying FNN to RF ensemble learning, we find that more studies need to be done in this area. Could FRF be adopted to replace the fully connected layer of CNN to solve the issue of too many parameters? Could size-adaptive training of the model be implemented in other way? How to use FRF to conduct transfer learning? And so on. Moreover, the parallelization and theoretical analysis of FRF still need to be investigated further.

Acknowledgements This work was supported by the Fundamental Research Funds for the Central Universities (2017XKQY082). The authors would like to thank the anonymous reviewers and the associate editor for their valuable comments.

References

1. Dimitriadou E, Weingessel A, Hornik K (2001) Voting-merging: an ensemble method for clustering. In: International conference on artificial neural networks, pp 217–244
2. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
3. Schapire RE (1999) A brief introduction to boosting. In: Sixteenth international joint conference on artificial intelligence, pp 1401–1406
4. Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 785–794
5. Friedman JH, Popescu BE (2008) Predictive learning via rule ensembles. Ann Appl Stat 2(3):916–954
6. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
7. Zhou ZH, Feng J (2017) Deep forest: towards an alternative to deep neural networks. arXiv:1702.08835
8. Huang G, Liu Z, Laurens VDM, Weinberger KQ (2017) Densely connected convolutional networks. In Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR), vol 1(no. 2), p 3
9. Chen CLP, Wan JZ (1999) A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. IEEE Trans Syst Man Cybern B Cybern 29(1):62–72
10. Chan TH, Jia K, Gao S, Lu J, Zeng Z, Ma Y (2014) PCANet: a simple deep learning baseline for image classification. IEEE Trans Image Process 24(12):5017–5032
11. Chen CLP, Liu Z (2018) Broad learning system: an effective and efficient incremental learning system without the need for deep architecture. IEEE Trans Neural Netw Learn Syst 29(1):10–24
12. Payne DB, Stern JR (1985) Wavelength-switched passively coupled single-mode optical network. In: Proceedings of the IOOC–ECOC, vol 85, p 585
13. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural Comput 3(1):79–87
14. Hoeting JA, Madigan D, Volinsky RCT (1999) Bayesian model averaging: a tutorial. Stat Sci 14(4):382–401
15. Bagui SC (2005) Combining pattern classifiers: methods and algorithms. Technometrics 47(4):517–518
16. Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36(1–2):105–139
17. Mallapragada PK, Jin R, Jain AK, Liu Y, Mallapragada PK, Jin R et al (2009) Semiboost: boosting for semi-supervised learning. IEEE Trans Pattern Anal Mach Intell 31(11):2000–2014
18. Bennett KP, Demiriz A, Maclin R (2002) Exploiting unlabeled data in ensemble methods. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, pp 289–296
19. Zhou ZH, Li M (2005) Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans Knowl Data Eng 17(11):1529–1541
20. Yoon J, Zame WR, Mihaela VDS (2018) ToPs: ensemble learning with trees of predictors. IEEE Trans Signal Process 66(8):2141–2152

21. Wu Q, Ye Y, Zhang H et al (2014) ForesTexter: an efficient random forest algorithm for imbalanced text categorization. Knowl Based Syst 67:105–116
22. Amiri S, Mahjoub MA, Rekik I (2018) Dynamic multiscale tree learning using ensemble strong classifiers for multi-label segmentation of medical images with lesions. In: Proceedings of the 13th international joint conference on computer vision, imaging and computer graphics theory and applications, vol 4, pp 419–426
23. Chakraborty T (2017) [IEEE 2017 IEEE international conference on data mining (ICDM)—New Orleans, LA (2017.11.18-2017.11.21)] 2017 IEEE international conference on data mining (ICDM)—EC3: combining clustering and classification for ensemble learning, pp 781–786
24. Strauss T, Hanselmann M, Junginger A et al (2017) Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv:1709.03423
25. Zhou ZH (2012) Ensemble methods: foundations and algorithms. Chapman and Hall/CRC, London
26. Hinton GE, Osindero S, Teh YW (2014) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
27. Wang Y, Xia ST, Tang Q et al (2018) A novel consistent random forest framework: Bernoulli random forests. IEEE Trans Neural Netw Learn Syst 29(8):3510–3523
28. Samaria FS, Harter AC (1994) Parameterization of a stochastic model for human face identification. IEEE Workshop Appl Comput Vis 22:138–142
29. Asuncion A, Newman D (2007) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine