Evolving Deep Forest with Automatic Feature Extraction for Image Classification Using Genetic Programming

Ying Bi, Bing Xue and Mengjie Zhang

School of Engineering and Computer Science,
Victoria University of Wellington, Wellington 6140, New Zealand
{Ying.Bi, Bing.Xue, Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract. Deep forest is an alternative to deep neural networks to use multiple layers of random forests without back-propagation for solving various problems. In this study, we propose a genetic programming-based approach to automatically and simultaneously evolving effective structures of deep forest connections and extracting informative features for image classification. First, in the new approach we define two types of modules: forest modules and feature extraction modules. Second, an encoding strategy is developed to integrate forest modules and feature extraction modules into a tree and the search strategy is introduced to search for the best solution. With these designs, the proposed approach can automatically extract image features and find forests with effective structures simultaneously for image classification. The parameters in the forest can be dynamically determined during the learning process of the new approach. The results show that the new approach can achieve better performance on the datasets having a small number of training instances and competitive performance on the datasets having a large number of training instances. The analysis of evolved solutions shows that the proposed approach uses a smaller number of random forests over the deep forest method.

Keywords: Evolutionary Deep learning, Genetic Programming, Deep Forest, Image Classification, Feature Extraction

1 Introduction

In recent years, deep learning algorithms have achieved a big success in many applications [1]. Image classification is one of the important application areas of deep learning. Many famous deep neural networks (DNNs) have been developed, such as AlexNet, GoogleNet, VGGNet, ResNet, and DenseNet [2–4]. These methods have achieved impressive performance on many large image classification datasets. However, these methods have a number of important limitations. First, rich domain expertise is needed to design a powerful DNN model/architecture. For example, AlexNet and VGGNet are both convolutional neural networks (CNNs) but have different structures and classification performance. Second, DNNs typically require a large amount of training data to train

a model, which often has a huge number of parameters. For example, AlexNet has over 60 million parameters and VGGNet has 138 million parameters, which can only be trained using thousands of training instances. With such parameters, these models cannot be directly applied for solving the tasks with limited training data. In many real-world applications, such as medical applications, the data are difficult or expensive to collect. Learning from limited training data, i.e., also know as few-shot or zero-shot learning [5], for solving a task is important, but the current common DNNs cannot directly solve. Furthermore, interpretability is another problem with them. It is difficult to understand/interpret these models (or the learned features) why they are effective for solving a task. However, in many industrial applications, interpretability is a critical factor for the acceptance and adoption of the model/method.

Motivated by these limitations, many other types of deep models have been developed, such as deep forest (also known as gcForest) [6]. Deep forest builds a deep model based on random forest using a cascade structure and a multi-grained scanning. The cascade structure allows it to have multiple layers of random forests to learn a complex representation of the data for solving a problem. The cascade structure is shown in Fig. 1. The multi-grained scanning scheme uses a sliding window to scan the raw features from sequence data or image data to form a feature vector as the inputs of random forests. Deep forest has been employed to solve many tasks and achieved promising results [6]. However, the deep forest uses fixed structures and parameters of random forests at each layer and the features used are simple without complex transformation. This may limit the performance of deep forest for image classification, especially particular images, e.g., texture images. Although many variants of deep forests have been developed, none of them addressed these two issues.

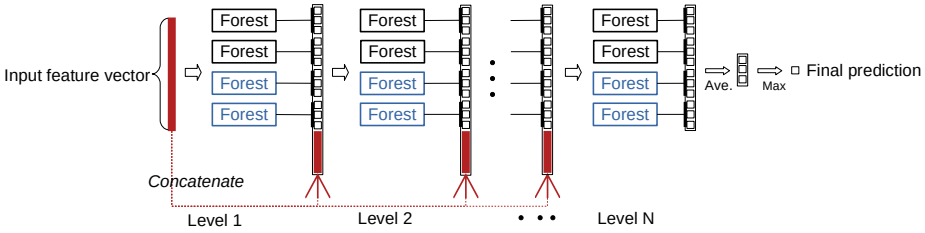

Fig. 1. The cascade forest structure [6].

In this paper, an evolutionary deep forest (EvoDF) approach is developed to automatically extract domain-specific features and find appropriate connections of random forests with effective parameters simultaneously for image classification. Instead of manually designing the structure of forests and choosing its key parameters, EvoDF can automatically find effective structures and parameters of the random forests in deep forest. Feature extraction modules are proposed to perform region selection, descriptor selection and feature description. They allow EvoDF to find effective features, such as local binary patterns (LBP), to solve typical image classification tasks, such as texture classification. Different from deep forest, EvoDF encodes solutions as trees with variable lengths and

uses a population to search for the best solution with fewer parameters. In the experiments, EvoDF achieves better results on the datasets of a small number of training instances and competitive results on the datasets of a large number of training instances than deep forest. Further analysis shows that the solution found by EvoDF has a small number of random forests than those in deep forest.

2 Related Work

Evolutionary Algorithms: Evolutionary algorithms can search for the best solutions via a number of generations based on genetic beam search. Evolutionary algorithms are known for their powerful global search ability and non-differentiable requirement [7]. They have been applied to many tasks in machine learning, such as optimizing the architectures of DNNs [8–10] and finding the optimal model of ensemble [11–13]. However, no evolutionary algorithms have been developed to optimize the structure of deep forest. In this study, we use an evolutionary algorithm, i.e., genetic programming (GP) [14], to achieve automatically search for the structure of forests, extract features and select suitable parameters of the forests for image classification.

Feature Extraction: Feature extraction is essential for solving image classification tasks. Traditional methods to feature extraction are local binary patterns (LBP) [15], histogram of oriented gradients (HOG) [16], and scale-invariant feature transform (SIFT) [17]. In recent decades, automatically extracting/learning features from images becomes increasingly popular. Many DNNs can achieve this by learning deep features. More related work can be found in [18]. In deep forest, the features are from images via multi-grained scanning, which may not be effective for classifying particular images, such as texture images. Therefore, we add a feature extraction process into deep forest to achieve automatic feature extraction for image classification.

Ensemble Methods: Deep forest is a typical ensemble method, using multiple decision trees to perform prediction. Ensemble methods train multiple base learners using traditional machine learning algorithms for solving a task [19]. Typically, an ensemble of diverse and accurate classifiers can obtain a better generalization performance than a single classifier [20]. But the diversity is an open issue in ensemble learning [21]. Various ensembles have been developed such as deep super learner [22] and autostacker [11], which make the ensemble model deeper. Recent applications of ensemble methods in machine learning and deep learning can be found in [12, 20, 23].

Deep Forest Variants: Recently, several variants of deep forests have been developed. Zhou *et al.*[24] extended the deep forest method to generate compact binary code for hashing in image retrieval. Zhu *et al.* [25] proposed an efficient training framework of deep forest on distributed task-parallel platforms. Zhang *et al.* [26] implemented distributed deep forest and tested it on an extra-large dataset of cash-out fraud detection. As a recently proposed method, deep forest is still at early stage and has large research potential. In this study, we aim to improve the performance of deep forest for image classification.

3 The Proposed Approach

In this section, we will describe the proposed approach: evolutionary deep forest (EvoDF). First, we will introduce two main modules, i.e., the forest module and the feature extraction module. Then the encoding and search strategy are presented, followed by the description of the overall algorithm.

3.1 Forest Modules

In deep forest, each layer/level has a predefined number of random forests. To allow EvoDF to search the structures of random forests, we relax this predefined structure of random forests by designing a single module and extending it to deep or wide modules. In EvoDF, the single module is constructed using one random forest, as shown in Fig. 2. The output of the single module is the concatenation of the input and the predictions of the random forest. The predictions of a random forest are the average scores of multiple trees in each class for a classification problem. For example, if the problem has c classes and the dimension of the input feature vector is f (the number of features in the dataset), the dimension of the output is $c + f$.

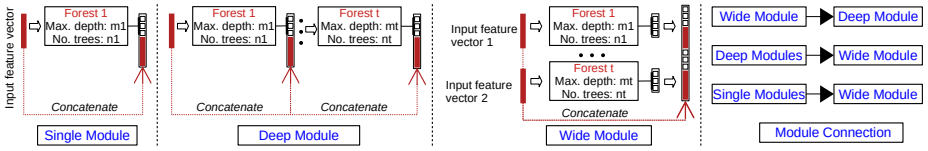

Fig. 2. Forest modules.

The single module can be extended to a deep or wide module, as shown in Fig. 2. The deep module is constructed using multiple single modules sequentially. The output of a single module is the input of the next single module. The final output dimension of a deep module is $f + t \times c$, where t indicates the number of single modules. The wide module is constructed using single modules parallelly. Every single module has various input feature vectors, which can be from different feature extraction modules introduced in the following subsections. The final output dimension of a wide module is $\sum_{i=1}^{t}(f_i + c)$, where f_i represents the input dimension of the ith single module. The deep and wide modules can connect with each other, e.g., using the output feature vector of one module as the input feature vector of another module. This leads to an easy extension of a complex model, which can be constructed by EvoDF.

It is known that the diversity of base learners are important for constructing an ensemble with good generalization ability [19]. The diversity can be enhanced by data sample manipulation, input feature manipulation, learning parameter manipulation, and output representation manipulation [6, 19]. In these forest modules, random forest and extremely randomized trees are employed. These two types of random forests can be automatically selected to construct a module. The key parameters, the number of trees and the maximum number of tree depth

of each forest, can be automatically determined during the learning process of EvoDF, which will be introduced in the later sections. It is also noted that the input feature vectors for the forest in the wide module are different, which further enhances the diversity.

3.2 Feature Extraction Modules

In deep forest, the inputs of random forests are raw data/images or the concatenation of the predictions and raw data. This may not be effective for solving particular image classification tasks, e.g., texture classification. In the EvoDF method, three new feature extraction modules are developed to extract informative features from images. The three possible feature extraction modules are illustrated in Fig. 3 and Fig. 4.

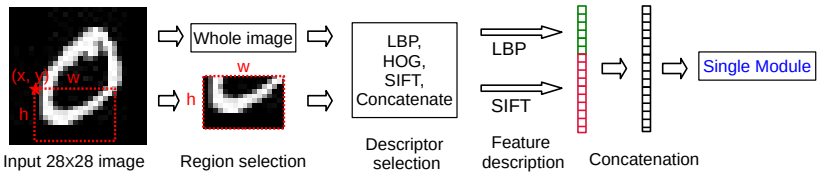

Fig. 3. One possible feature extraction module.

The **first feature extraction module** (module 1 in Fig. 4) consists of descriptor selection and feature description. Four commonly used descriptors, i.e., LBP, HOG, SIFT, and Concatenation [27], are employed to transform an image into features. Descriptor selection is to select one of the four descriptors and feature description is to extract features using the selected descriptor. The output of the feature extraction module is a feature vector. The **second feature extraction module** (module 2 in Fig. 4) has an additional process: region selection, which aims to select a small key region from a large input image. From the selected region, image features can be extracted. To select a region, the position of the top-left point of the region (x, y) and the size of the region (w, h) are needed. These parameters can be automatically selected during the learning process of EvoDF. The **third feature extraction module** (module 3 in Fig. 4) concatenates the features produced by modules 1 or 2 into a feature vector. It can produce a combination of various features.

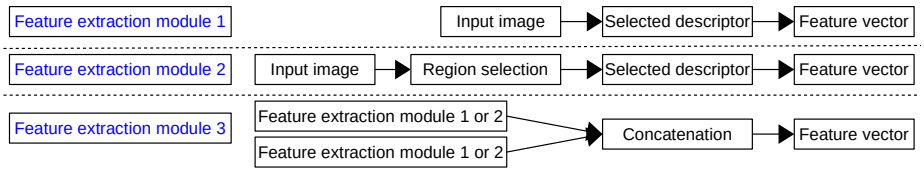

Fig. 4. Feature extraction modules.

3.3 Encoding and Search Strategy

The forest modules and feature extraction modules are connected and represented using a tree structure. A potential advantage is that a tree can be easily

extended to be deeper or broader. The encoding of a solution with forest modules and feature extraction modules is based on strongly typed genetic programming [28]. An example solution and its corresponding encoding are shown in Fig. 5.

In this encoding, the internal nodes are functions/models, including the functions in the feature extraction modules and forest modules. These functions are region selection functions, image descriptors (LBP, HOG, SIFT, and Concatenation), single module, random forest, combine, and average (average2, average3 and average 4). The random forest function is employed after the forest modules, which directly returns the score of each class for a classification problem. The combine function is used in the wide forest module to combine the output of several forests and their predictions. The combine function combines two or three single modules. The average function is used as the root node, where the predictions of various random forests are averaged and the class label is obtained according to the maximum score. In EvoDF, the average function can connect with two, three or four random forests. It is noted that the random forest can be replaced by extremely randomized trees during the learning process of EvoDF.

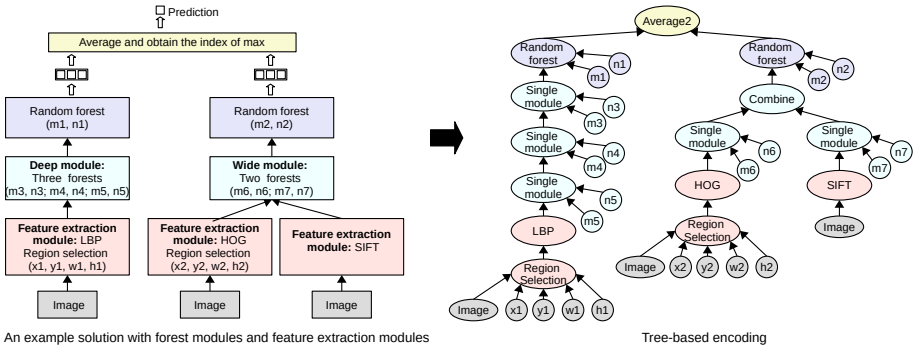

Fig. 5. An example tree/structure with forest modules and feature extraction modules and its encoding in the proposed EvoDF approach.

The leaf nodes are parameters and the input image. The parameters are the coordination (x, y) of the top-left point of the selected region, the size (width w and height h) of the selected region, the number of trees (m) in forests, and the maximum tree depth (n). In the EvoDF method, x is in the range of $[0,\ W-10]$, y is in the range of $[0,\ H-10]$, w and h are in the range of $[10, 20]$, where W indicates the width of the image and H indicates the height of the image. With these settings, region selection functions can select a region with a size from 10×10 to 20×20. The range of m is $[50,\ 1000]$ with a step of 50. The range of n is $[10,\ 100]$ with a step of 10.

In EvoDF, a number (population) of trees/solutions are randomly generated by selecting functions to form the internal nodes and selecting parameters from the predefined ranges to form the leaf nodes. Each tree is evaluated on a training set to obtain its classification performance using the k-fold cross-validation or the hold-out method. Then better trees are selected and the mutation and crossover operations are employed to generate new trees from the selected trees. The crossover operation swaps two branches of two trees based on a selected node

to generate two new trees. The mutation operation replaces the branch of a tree based on a selected node with a new generated branch. These two operations can change the depths of trees, e.g., from wide to deep. The mutation operation can introduce new branches into the current population. In addition, the functions and parameters of a tree can also be changed by these two operations, leading to a search towards the best tree/solution.

3.4 Algorithm Procedure

Training Process: The training/learning procedure of EvoDF is described in Algorithm 1. In this procedure, a population of solutions are generated and evaluated on the training set $\mathcal{D}_{train}$. In the evaluation procedure, the classification performance of the tree/solution is calculated using the k-fold cross-validation or hold-out methods. Then the tree with the best performance is recorded. A selection method is used to select better trees for crossover or mutation. The better trees may have better forest modules or feature extraction modules, which are inherited to the next generation. Then a new population of trees is generated. The overall process is repeated until the maximum number of generation (G) is reached. Finally, the best solution is returned and tested on a test set.

Algorithm 1: Algorithm Training Procedure

Input : $\mathcal{D}_{train}$: the training set.
Output : $Best\_Model$: the best solution.

$P_0 \leftarrow$ Initialise a population of solutions;
$g \leftarrow 0$;
while $g < Maximal\ number\ of\ generations$ **do**
  Evaluate P_g on $\mathcal{D}_{train}$ using the k-fold cross-validation or hold-out methods;
  Update $Best\_Model$ based on P_g;
  Select better solutions from P_g using a selection metod;
  $P_{g+1} \leftarrow$ New solutions generated using crossover and mutation;
  $g \leftarrow g + 1$;
end
Return $Best\_Model$.

The parameters for EvoDF are the population size, the maximum number of generations, the mutation rate, the crossover rate, and the selection method. It should be noted that the number of parameters for EvoDF is smaller than that of deep forest or DNNs, as listed in [6]. Furthermore, the parameter settings for EvoDF can follow the commonly used settings of GP [27, 12]

Test Process: In the test process, the random forest in the best solution is trained using $\mathcal{D}_{train}$ and the solution is tested on the test set. The accuracy of the test set is reported. It should be noted that the test set has never been used in the training or learning process.

4 Experiments and Results

4.1 Configuration

In this section, we compare the proposed EvoDF approach with deep forest (gcForest) and several other algorithms. The settings of most comparison algorithms refer to [6]. In EvoDF, the population size is 100 and the maximum number of generation is 20. Note that we use a smaller number of generation due to the high computational cost. The crossover rate is 0.5 and the mutation rate is 0.5. A larger mutation rate than the commonly used one is expected to increase the diversity of the population. We use the same parameter settings for EvoDF on all the datasets for generality. For the datasets having a small number of training instances, 3-fold cross-validation on the training set is used in the learning process of EvoDF (line 4 of Algorithm 1) to improve the generalisation performance. For the datasets having a large number of training instances, the hold-out method is used in the learning process of EvoDF. The training set is split into two subsets, one for training random forests and one for calculating the accuracy of the random forests in the fitness evaluation process. It is noted that evolutionary algorithms often run 30 or 50 times and report the average results. However, many existing image classification methods often report the results of one run due to the high computational cost. Therefore, the experiments of EvoDF run five independent times and the averaged results are reported.

Six different datasets are employed in the experiments. These datasets are ORL [29], Extend Yale B [30], SCENE [31], KTH [32], MNIST [33], and CIFAR-10 [34]. These datasets represent a variety of image classification tasks, i.e., object classification, face recognition, scene classification, and texture classification, aiming to demonstrate the effectiveness of EvoDF on a wide range of image classification tasks with the same parameter settings.

4.2 Classification Results

Face Recognition: ORL [29] is a face recognition dataset, having 400 images in 40 classes. Each class has 10 images. Following the settings in [6], 5/7/9 images are used for training and the remaining images are used for testing, respectively. Table 1 compares the test accuracy of EvoDF and five baseline algorithms. The results of the baseline methods are from [6]. From Table 1, we can find that the proposed EvoDF method achieves better results than any of the baseline methods, including gcForest, using various numbers of training instances. The performance of EvoDF does not degrade significantly when the number of training instances is decreased. Compared with other baseline methods, EvoDF is less affected by reducing the number of training instances.

Face Recognition: Extend Yale B [30] contains 2,414 facial images sampled from 38 different people under various illumination conditions. In the experiments, we randomly select 10/20/30 images for training and the remaining images for testing, respectively. Two gcForests, one with four forests and one with eight forests at each level, are used for comparisons. The random forest and the

Table 1. Comparison of test accuracy (%) on **ORL**

	5 images	7 images	9 images
EvoDF	**97.00 (96.30±0.57)**	**97.50 (97.17±0.46)**	**100 (99.0±1.37)**
gcForest	91.00	96.67	97.50
Random Forest	91.00	93.33	95.00
CNN (five layers)	86.50	91.67	95.00
SVM (rbf kernel)	80.50	82.50	85.00
kNN (k=3)	76.00	83.33	92.50

architecture of CNN are the same as that in [6]. The batch size of CNN is set to 10, 20 and the number of epochs is set to 50, 100, respectively. The SVM with a linear kernel achieves better performance than with an RBF kernel. The number of neighbours in k-NN is tuned and the best results obtained when k=1.

The results on the Extend Yale B dataset are listed in Table 2. The EvoDF approach achieves better results than any of the baseline methods using 20 or 30 training images. In the first case, EvoDF achieves worse results than SVM and better results than the remaining algorithms. The results show that EvoDF achieves better results than gcForest with four or eight forests at each level.

Table 2. Comparison of test accuracy (%) on **Extend Yale B**

	10 images	20 images	30 images
EvoDF	81.36 (80.66±0.86)	**95.37 (94.09±1.33)**	**98.29 (97.93±0.47)**
gcForest (8 forests)	72.21	91.71	96.88
gcForest (4 forests)	72.06	91.05	96.11
Random Forest	75.44	91.47	96.65
CNN (20, 100)	69.96	86.96	94.94
CNN (10, 50)	71.52	92.19	94.31
SVM (linear kernel)	**82.19**	92.55	95.25
kNN (k=1)	39.92	54.57	62.46

Scene and Texture Classification: The **SCENE** [31] dataset has 3,859 natural scene images of 13 classes. The images are sampled under different conditions and have high image variations. The **KTH** [32] dataset is a texture classification task of 10 classes. The total number of images is 810. The images are sampled in nine scales with three poses under four illumination conditions. In the experiments, 100 images per class of the SCENE dataset are used for training and the remaining images are used for testing. For KTH, 40 images per class are used for training and the remaining images are used for testing. kNN uses k=1 on SCENE and k=5 on KTH after tuning. CNN uses 20 batch size and 100 epochs.

The results obtained by EvoDF and the baseline methods on these two datasets are listed in Table 3. The EvoDF method achieves better results than any of the compared methods on these two datasets. Specifically, EvoDF increases the accuracy by 14.42% on SCENE and by 11.95% on KTH. It is also noticeable that EvoDF achieves better results than gcForest. The EvoDF approach can extract LBP and SIFT features, which are typically for texture, shape and appearance description. This may increase the classification performance of EvoDF on the texture and scene datasets.

Table 3. Comparison of test accuracy (%) on **SCENE** and **KTH**

Dataset	SCENE (100 images)	KTH (40 images)
EvoDF	**67.96 (63.07±3.85)**	**87.80 (85.17±2.79)**
CNN	53.54	75.85
gcForest (8 forests)	39.62	64.63
Random Forest	36.19	60.98
SVM (linear kernel)	19.42	41.46
kNN	22.78	36.34

MNIST [33] is the task of handwritten digit classification. It has 60,000 training images and 10,000 testing images. In the learning process of EvoDF, 30,000 images of the training set are used to train forests and the remaining 30,000 images are used to evaluate. In the testing process, the full training set is used to train the forests and the model is tested on the test set. The results of the baseline methods are from the corresponding references on the same test sets. The results are listed in Table 4. It can be found that the accuracy obtained by EvoDF is very close to that by gcForest, i.e., 0.34% lower. The EvoDF approach achieves better results than CascadeForest, deep belief net, deep forest-based hashing, SVM, deep super learner, and random forest.

Table 4. Comparison of test accuracy (%) on **MNIST**

gcForest	99.26 [6]
LeNet-5	99.05 [6]
EvoDF	98.92 (98.83±0.10)
Deep Belief Net	98.75 [35]
SVM (rbf kernel)	98.60 [6]
Deep forest-based hashing	98.50 [24]
Deep super learner	98.42 [22]
CascadeForest	98.02 [6]
Random Forest	96.80 [6]

CIFAR-10 [34] is a complex object classification dataset. CIFAR-10 has 50,000 32×32 color training images and 10,000 colour testing images. In the experiments, we use gray-scale images to reduce computational cost. In the training process of EvoDF, 30,000 images of the training set are used for training and 20,000 images are used for evaluation. Table 5 lists the test accuracy of all the methods. It can be found that EvoDF achieves better results than the random forest, deep forest-based hashing, multilayer perceptron, logistic regression, and SVM. The performance of EvoDF is inferior to state-of-the-art DNNs and a little behind the gcForest (default). The performance of gcForest is further improved by increasing grains and using gradient boosting decision tree (GBDT) [36] as gcForest (gbdt) achieves better results than gcForest (default). The performance of EvoDF can be further improved by using GBDT, or using a large number of population size and generations, or using 3-fold cross-valuation in training. However, due to the limitation of computational resources, we have not tested the performance of EvoDF under that configuration. In addition, it is noteworthy

that EvoDF can achieve better results than gcForest or other methods on the image classification datasets using a small number of training instances.

Table 5. Comparison of test accuracy (%) on **CIFAR-10**

ResNet	93.57 [3]
AlexNet	83.00 [2]
gcForest (gbdt)	69.00 [6]
gcForest (5grains)	63.37 [6]
Deep Belief Net	62.20 [37]
gcForest (default)	61.78 [6]
EvoDF	61.27 (60.84±0.33)
Deep forest-based hashing	55.90 [24]
Random Forest	50.17 [6]
Multilayer Perceptron	42.20 [38]
Logistic Regression	37.32 [6]
SVM (linear kernel)	16.32 [6]

5 Further Analysis and Discussions

5.1 Discussions of the Classification Results

From Tables 1-5, it can be found that the EvoDF achieves better results than any of the baseline methods on the datasets having a small number of training instances and comparable results on the datasets having a large number of training instances. The results indicate that EvoDF is effective for various types of image classification tasks, especially when the training data is limited. Compared with the original deep forest, the proposed EvoDF approach can automatically search for the structures of deep forests, find appropriate parameters and select effective features as the inputs of the forests. With these designs, the performance of deep forest on image classification has been improved. Although the performance of EvoDF on the large dataset, i.e., CIFAR10, is a little inferior, the comparisons show the potential of EvoDF by automatically searching the structures of forest and extracting features from images. To sum up, EvoDF is suitable and effective for solving tasks with a small number of training instances, such as in the medical, security, or biological domains.

It is also noted that the classification performance of gcForest and EvoDF is worse than deep CNNs, i.e., ResNet and AlexNet, on the large dataset (CIFAR10). These deep CNNs are well-developed algorithms and the area of CNNs for image classification has been developed for over ten years. These methods require a large number of computing resources (GPU) to obtain the current performance on the large datasets. In contrast, deep forest is a newly developed algorithm and its potential has not been comprehensively investigated. The current implementation of deep forest is based on CPU rather than GPU, which may limit its performance. The area of deep forest still has a large research space and needs further investigation in the future.

5.2 Analysis of the Evolved Solution

An example solution found by EvoDF is visualised to show what features are extracted and why it achieves good performance. The best solution is evolved on MNIST, as shown in Fig. 6. It achieves 98.92% test accuracy. It has three branches and the final prediction decision is made from three random forests. Each branch uses a particular feature extraction module to generate features from the input image. The features include SIFT features, LBP features, HOG features, and raw pixels. It is noted that each random forest uses different inputs and has different parameters, which increases the diversity of the classifiers in the ensemble. The solution shows that the connections between the forest modules and the feature extraction modules are very flexible. For example, the output of a single forest module and the output of feature extraction modules can be concatenated to form the input of another random forest.

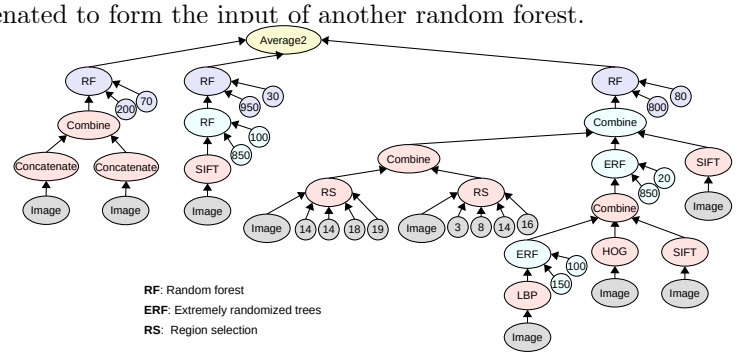

Fig. 6. The solution found by EvoDF on the **MNIST** dataset.

Although EvoDF achieves slightly worse results than gcForest on MNIST, the number of forests in the solution of EvoDF is much smaller than that in gcForest. gcForest uses at least eight random forests at each level (it could use more than 16 random forests), while the solution of EvoDF only uses seven random forests totally. This indicates that EvoDF can improve the utilisation of random forests, i.e., use a small number of random forests to achieve competitive performance by finding an effective structure of forest and feature extraction.

6 Conclusions

In this study, we developed an evolutionary deep forest approach with automatic feature extraction and structure search for image classification. Compared with deep forest, the proposed EvoDF approach found effective connections of forests with various parameters and extracted domain-specific features for image classification. In addition, the new approach used fewer parameters than deep forest and DNNs. The results showed that EvoDF achieved better performance than deep forest and other algorithms on the datasets having a small amount of training data and comparable performance on the datasets having a large number of training data. Further analysis showed that EvoDF can find effective connections of a small number of random forests to achieve competitive performance. In the future, we will further improve the performance of this approach by implementing it distributedly on large datasets.

References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436 (2015)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
4. Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S.: A survey of the recent architectures of deep convolutional neural networks. arXiv preprint arXiv:1901.06032 (2019)
5. Wang, Y., Yao, Q., Kwok, J., Ni, L.: Few-shot learning: A survey. arXiv preprint arXiv:1904.05046 (2019)
6. Zhou, Z.H., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: Proceedings of International Joint Conferences on Artificial Intelligence. pp. 3553–3559 (2017)
7. Al-Sahaf, H., Bi, Y., Chen, Q., Lensen, A., Mei, Y., Sun, Y., Tran, B., Xue, B., Zhang, M.: A survey on evolutionary machine learning. Journal of the Royal Society of New Zealand 49(2), 205–228 (2019)
8. Sun, Y., Xue, B., Zhang, M., Yen, G.G.: Evolving deep convolutional neural networks for image classification. IEEE Transactions on Evolutionary Computation (2019)
9. Bi, Y., Xue, B., Zhang, M.: An evolutionary deep learning approach using genetic programming with convolution operators for image classification. In: Proceedings of IEEE Congress on Evolutionary Computation. pp. 3197–3204 (2019)
10. Baioletti, M., Milani, A., Santucci, V.: Learning bayesian networks with algebraic differential evolution. In: Proceedings of International Conference on Parallel Problem Solving from Nature. pp. 436–448 (2018)
11. Chen, B., Wu, H., Mo, W., Chattopadhyay, I., Lipson, H.: Autostacker: A compositional evolutionary learning system. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 402–409 (2018)
12. Bi, Y., Xue, B., Zhang, M.: Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification. IEEE Transactions on Cybernetics (2020)
13. Bi, Y., Xue, B., Zhang, M.: An automated ensemble learning framework using genetic programming for image classification. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 365–373 (2019)
14. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT press, Cambridge (1992)
15. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
16. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. vol. 1, pp. 886–893 (2005)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
18. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. arXiv preprint arXiv:1809.02165 (2018)

19. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC (2012)
20. Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(4), 1–19 (2018)
21. Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems. pp. 1–15. Springer (2000)
22. Young, S., Abdou, T., Bener, A.: Deep super learner: A deep ensemble for classification problems. In: Proceedings of Canadian Conference on Artificial Intelligence. pp. 84–95 (2018)
23. Ding, C., Tao, D.: Trunk-branch ensemble convolutional neural networks for video-based face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(4), 1002–1014 (2017)
24. Zhou, M., Zeng, X., Chen, A.: Deep forest hashing for image retrieval. Pattern Recognition (2019)
25. Zhu, G., Hu, Q., Gu, R., Yuan, C., Huang, Y.: Forestlayer: Efficient training of deep forests on distributed task-parallel platforms. Journal of Parallel and Distributed Computing (2019)
26. Zhang, Y.L., Zhou, J., Zheng, W., Feng, J., Li, L., Liu, Z., Li, M., Zhang, Z., Chen, C., Li, X., et al.: Distributed deep forest and its application to automatic detection of cash-out fraud. ACM Transactions on Intelligent Systems and Technology 10(5), 1–19 (2019)
27. Bi, Y., Xue, B., Zhang, M.: An effective feature learning approach using genetic programming with image descriptors for image classification [research frontier]. IEEE Computational Intelligence Magazine 15(2), 65–77 (2020)
28. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
29. Samaria, F.S., Harter, A.C.: Parameterisation of a stochastic model for human face identification. In: Proceedings of 1994 IEEE Workshop on Applications of Computer Vision. pp. 138–142 (1994)
30. Lee, K.C., Ho, J., Kriegman, D.J.: Acquiring linear subspaces for face recognition under variable lighting. IEEE Transactions on Pattern Analysis & Machine Intelligence (5), 684–698 (2005)
31. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 524–531 (2005)
32. Mallikarjuna, P., Targhi, A.T., Fritz, M., Hayman, E., Caputo, B., Eklundh, J.O.: The kth-tips2 database. Computational Vision and Active Perception Laboratory, Stockholm, Sweden pp. 1–10 (2006)
33. LeCun, Y., Cortes, C., Burges, C.J.: The mnist database. URL http://yann. lecun. com/exdb/mnist (1998)
34. Krizhevsky, A., Nair, V., Hinton, G.: The cifar-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html 55 (2014)
35. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
36. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 785–794 (2016)
37. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
38. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Advances in neural information processing systems. pp. 2654–2662 (2014)