Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2013181, 9 pages
https://doi.org/10.1155/2022/2013181

Research Article

Image Classification and Recognition Based on Deep Learning and Random Forest Algorithm

Erhui Xi

School of Engineering, Guangzhou College of Technology and Business, Foshan, 528137 Guangdong, China

Correspondence should be addressed to Erhui Xi; 11109822143@stu.wzu.edu.cn

Received 12 April 2022; Revised 30 April 2022; Accepted 5 May 2022; Published 3 June 2022

Academic Editor: Kalidoss Rajakani

Convolutional neural network (CNN) is an important way to solve the problems of image classification and recognition. It can realize effective feature representation and make continuous breakthroughs in the field of image recognition, but it needs a lot of time in the training process. At the same time, random forest (RF) has the advantages of fast training speed and high classification accuracy. Aiming at the problem of image classification and recognition, this paper proposes a hybrid model based on CNN, which inputs the features extracted by CNN into RF for classification. Since the random weight network can also obtain valid results, the gradient algorithm is not used to adjust the network parameters to avoid consuming a lot of time. Finally, experiments are conducted on MNIST dataset and rotated MNIST dataset, and the results show that the classification accuracy of the hybrid model has improved more than RF, and also, the generalization ability has been improved.

1. Introduction

Handwritten digit recognition is a kind of pattern recognition, which is included in the character recognition technology, and the key technology for processing some data information is handwritten digit recognition, such as financial statements, postal codes, and various bills [1]. In 1998, Lecun et al. proposed the handwritten digit recognition model LeNet I5, which was widely used to recognize handwritten digits of U.S. bank checks; in literature [2], the K-nearest neighbor classification (KNN) algorithm was used to achieve a classification error rate of 2.83% on a MNIST dataset. The K-nearest neighbor (KNN) algorithm in [3] achieved a classification error rate of 2.83% on the MNIST dataset; support vector machine (SVM) and its improved algorithms were widely used in classification tasks. In 2012, Niu et al. proposed a hybrid model CNN-SVM applied to digital recognition [4], using CNN for feature extraction and SVM as a classifier, combining the respective advantages of both sides and achieving good experimental results in image classification tasks; Luo et al. [5] in 2014 proposed a hybrid method ELM-SRC (sparse representation-based classification), which combined the advantages of SRC in processing noisy images and the fast training speed of ELM, and conducted experiments on USPS handwritten dataset, which both improved the classification accuracy and ensured the time efficiency.

CNN is a deep learning algorithm, which is widely used in many fields, such as target recognition, scene classification, and face recognition [6]. CNN learns by layer by layer, and each layer automatically extracts different features from the input image, which works very well and is considered as one of the representatives of general-purpose image recognition systems [7]. Usually, the neurons in the convolutional layer are connected to the upper layer by local perceptual fields, and the features of that local area are obtained by convolution, and the secondary features are extracted by pooling in the pooling layer [8]. The structure of alternating convolutional and pooling layers makes it possible to tolerate input samples with certain distortions [9, 10]. However, the CNN requires a BP algorithm to adjust the parameters. Random forest (RF) is proposed in 2001, which has high accuracy in classification and regression and fast training speed and is not prone to overfitting problems, and also performs well in noise immunity [11, 12]. The existing RF-based classifiers rely on hand-selected features; however, manual feature selection is very time-consuming and requires expertise background, and whether good results can be achieved depends on a certain amount of experience and luck. In literature [13, 14], it is proposed that the network structure can achieve good results even with randomly unpretrained weights.

The paper [15, 16] generated realistic original images using a random initialized network reduction without any training; the paper [17, 18] proposed a deep neural network and RF model to detect retinal vessels in fundus images in 2015 and achieved an accuracy of 93.27% on the DRIVE dataset. Based on the above issues, a hybrid model hybrid model is proposed in the paper. In the hybrid model, the features are extracted with a CNN with random weights and then handed over to the RF to complete the classification, which makes the model take much less time in extracting features, overcoming the problem of long training time of CNN and solving the drawback of manual feature selection by RF [19, 20].

2. Convolutional Neural Networks

The combination of convolutional, downsampling, and fully connected layers constitutes the most classical network structure of CNN. The convolutional layer uses a linear filter kernel to perform linear convolution, and then, a nonlinear activation function is added to compute the extracted features.

Figure 1 shows an example of a classical CNN neural network structure.

2.1. Convolutional Layers. The convolutional layer is the most critical component of CNN, also called filter or kernel, which is used for low-dimensional feature extraction on high-dimensional data. The parameters are a set of trainable convolutional kernels, each of which has a relatively small size (length $\times$ width) in order to extract the right size feature map without losing useful information. In each convolutional layer, there are a certain number of convolutional kernels, which is a hyperparameter in the CNN and needs to be empirically specified artificially, and each kernel computes. A feature map means that we have extracted some features of the input image; i.e., the original three-dimensional image becomes a two-dimensional feature map. The combination of all the feature maps is our output data, which can be used for further feature extraction or as the final feature extraction result. Multiple convolutional kernels are used to extract different aspects of features, such as color, contour, and background.

The depth of a normal deep neural network is mainly reflected in the deep layers of the network, which leads to a dramatic increase in parameters, while the most important feature of convolutional layers compared to the most common fully connected neural networks is that the parameters can be reduced substantially, even by orders of magnitude. If an $n \times n$ convolution kernel does the convolution operation on an $m \times m$ image with the same depth of the image, we get a new image of size $((m+n)/l)+1$, which is the feature map. The l divided by l is the step size we set for the convolution. The step size can be set to other values according to our needs, but if the division is not exhaustive, we need to fill the image so that it can take an integer number of steps, so the most important thing is to be able to divide the whole thing.

2.2. Pooling Layers. CNNs generally alternate between a convolutional layer followed by a pooling layer. Its most intuitive function is to reduce the dimensionality, so that the number of parameters in this layer is also reduced, making the computation simpler and faster, extracting important features and conforming to invariance. The most common way to do this is to downsample the input using a filter of size 2×2, where the four pixel values are combined into one pixel value. Each max-pooling operation takes the largest of the four numbers (some 2×2 region of the input image). The depth of the image does not change, and the image size is reduced while preserving as much of the original information as possible.

As shown in Figures 2 and 3, max-pooling has the advantage of not increasing the number of parameters to be adjusted and is generally more accurate than other methods, highlighting features, while mean-pooling tends to be more smooth.

2.3. Training Process. The CNN performs supervised guided training, and the process is roughly as follows.

2.3.1. FC Layer

(1) Calculate the output for the fully connected layer l as a function of

$$x^l = f\left(u^l\right), \text{where } u^l = W^l x^{l-1} + b^l, \tag{1}$$

where $f(\bullet)$ represents the activation function, and here we use the sigmoid function

The error loss of the training sample is

$$E^n = \frac{1}{2}\sum_{k=1}^{c}\left(t_k^n - y_k^n\right)^2 = \frac{1}{2}\left\| t^n - y^n \right\|_2^2, \tag{2}$$

where c indicates that there are a total of c classes in the multicategory problem.

(2) Weight update

For the convenience of presentation, we define the derivative of the error with respect to the basis as the sensitivity, whose expression is

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta. \tag{3}$$

The sensitivity of layer i in back propagation can be expressed as

$$\delta^i = \left(W^{i+1}\right)^T \delta^{i+1} \circ f'\left(u^i\right). \tag{4}$$

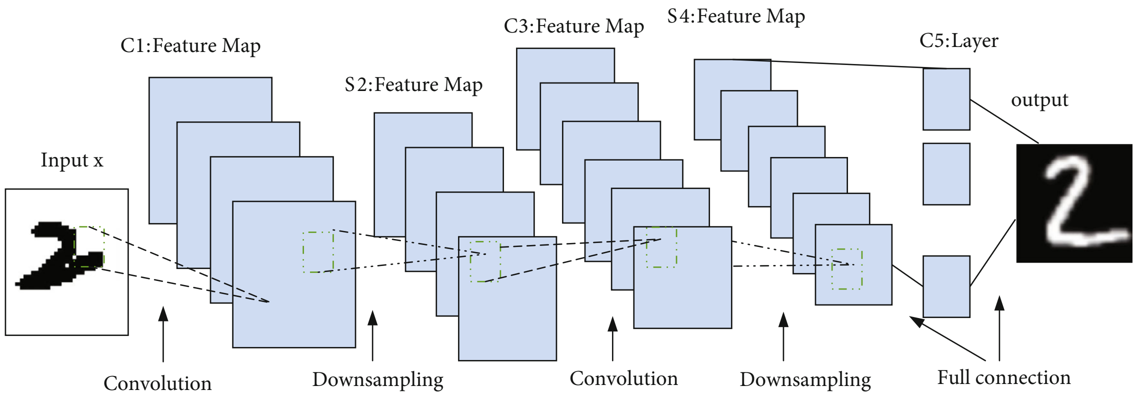

Figure 1: Schematic diagram of LeNet-5 model.

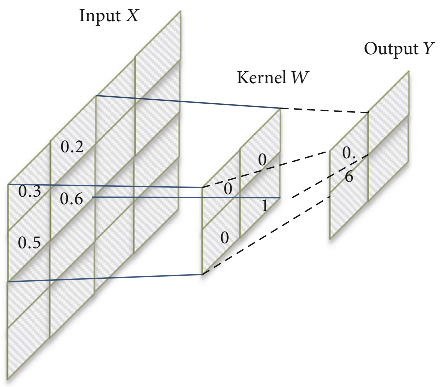

Figure 2: Max-pooling operation.

For the base value b in the output layer (at each layer, b is a vector), the partial derivative to the base value b for layer l is

$$\frac{\partial E}{\partial b^l} = \frac{\partial E}{\partial u'}\frac{\partial u^l}{\partial b^l} = \frac{\partial E}{\partial f(u^l)}\frac{\partial f(u')}{\partial u} = -\sum\left(t - f(u')\right)f'(u'). \tag{5}$$

For the weights $\mathbf{W}$ (at each level, $\mathbf{W}$ is a matrix), find the partial derivatives.

$$\frac{\partial E}{\partial \mathbf{W}^l} = \frac{\partial E}{\partial u^l}\frac{\partial u^l}{\partial \mathbf{W}} = \frac{\partial E}{\partial u^l}x^{l-1} = x^{l-1}\left(\delta^l\right)^T. \tag{6}$$

Here, x^{l-1} is the input of the l layer which is also the output of the upper layer, so the final amount of change in the weights is

$$\Delta W^l = -\eta\frac{\partial E}{\partial W^l} = -\eta x^{l-1}\left(\delta^l\right)^T. \tag{7}$$

2.3.2. Convolution Layer. In the convolution layer, the output is obtained after the filter kernel is calculated.

$$D_k, \theta_k = (x_1 x_2, \cdots, x_m), T. \tag{8}$$

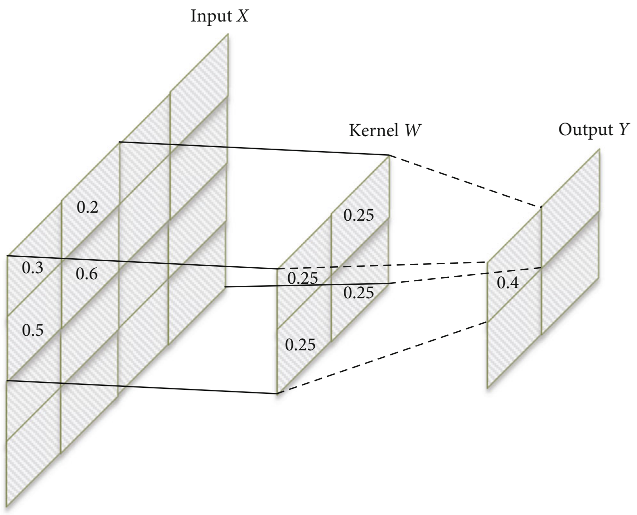

Figure 3: Mean-pooling operation.

From equation (4), we know that if we want to get the change value of the weight of each neuron in layer l, we must first get its corresponding sensitivity δ. In order to find this sensitivity, we need to first sum the sensitivity of the nodes in the next layer (to get δ^{l+1}) and use equation (4) to calculate the sensitivity δ^l corresponding to each neuron node in the current layer l. Next, the weights are updated.

Because the downsampling operation is often done after the convolutional layer, the feature map obtained in the pooling layer does not match the size of the input. The pooling layer is needed. The sensitivity of node j in the lth layer is given by taking a β for all the weights of the pooling map.

$$\delta_j^l = \beta_j^{l+1}\left(f'\left(u_j^l\right)^\circ up\left(\delta_j^{l+1}\right)\right), \tag{9}$$

where $up(x)$means upsampling x, depending on the pooling method described earlier. Now, for a given map, we can compute the gradient of the base value b as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v}\left(\delta_j^l\right)_{uv}. \tag{10}$$

Finally, the gradient of the weights W of the convolution kernel is calculated using Matlab's convolution function.

$$\frac{\partial E}{\partial K_{ij}^{l}} = \operatorname{rot180}\left(\operatorname{conv2}\left(x_i^{l-1}, \operatorname{rot180}\left(\delta_j^l\right), '\text{valid}'\right)\right). \quad (11)$$

2.3.3. Pooling Layer. For the pooling layer, the number of input feature maps is not changed, but only the input feature maps are made smaller: the

$$x_j^l = f\left(\beta_j^l \operatorname{down}\left(x_j^{l-1}\right) + b_j^l\right), \quad (12)$$

where down(•) denotes a pooling function. Each output feature map corresponds to a weight β and a bias b of its own.

Here, again we have to obtain the sensitivity before we can update the weights β and the basis b. If this layer is fully connected to the next layer, the gradient of the layer can be calculated directly by BP. However, when the sensitivity of the convolution kernel is not fully connected as equation (9) shows, the sensitivity of the convolution kernel is calculated here again with the help of the convolution function.

$$\delta_j^l = f'\left(u_j^l\right)^{\circ} \operatorname{conv2}\left(\delta_j^{l+1}, \operatorname{rot180}\left(K_j^{l+1}\right), '\text{ full}\right). \quad (13)$$

The gradient of the base value b is then calculated as in the convolution layer, as in equation (10). The sensitivity to the weights β is calculated as follows.

$$\frac{\partial E}{\partial b_j} = \sum_{u,v}\left(\delta_j^{l^{\circ}} d_j^l\right)_{uv}, \quad (14)$$

where $d_j^l = \operatorname{down}(x_j^{l-1})$.

3. Random Forest

RF consists of k classification trees, and its basic idea is to set multiple weak classifiers into one strong classifier. The classification tree consists of different nodes, where the root node represents the training set, each internal node represents a weak classifier that divides the samples according to a certain attribute, and each leaf node is a labeled training or test set that classifies the input data into several subsets. The final decision result of RF is the optimal result chosen by voting on all classification trees.

The Gini index is used to decide the optimal binary cut point for that feature. The Gini index $G_{\text{gini}}(D)$ represents the uncertainty of the set D. In the classification problem, suppose there are N classes, and for a given set of samples D, the Gini index is defined as

$$G_{\text{gini}}(D) = 1 - \sum_{n=1}^{N}\left(\frac{|C_n|}{D}\right)^2, \quad (15)$$

where C_n is the subset of samples in D that belong to the nth class. If the sample set D is divided into two parts D_1 and D_2 according to whether the value of feature A takes a or not, i.e.,

$$D_1 = \{(x, y) \in D \mid A(x) = a\}, D_2 = D - D_1. \quad (16)$$

Conditional on characteristic A, the Gini index of the set D is defined as

$$G_{\text{gini}}(D, A) = \frac{|D_1|}{|D|} G_{\text{gini}}(D_1) + \frac{|D_2|}{|D|} G_{\text{gini}}(D_2). \quad (17)$$

$G_{\text{gini}}(D, A)$ represents the uncertainty of the set D after the partitioning by $A = a$. When constructing a classification tree, the feature with the smallest Gini index and its corresponding optimal binary cut point are selected. The RF is constructed using the Gini index minimization criterion with the following steps.

(1) Using the bootstrap resampling method, the kth sample set is drawn back from the original sample set D. The kth sample set is denoted as D_k and a random vector θ_k, θ_k is generated for the kth classification tree that is independently and identically distributed with the previous random vectors. In this paper, we use $h(D_k, \theta_k)$ to represent the kth classification tree model

(2) Build classification trees for each of the k samples. The generation of the classification tree is the process of recursively building a binary classification tree, using the feature with the smallest Gini index to split the binary tree

(3) The final classification result is voted based on the results of each classification tree

The flow of constructing the random forest is shown in Figure 4.

4. Hybrid Model Based on Deep Learning and Random Forest

4.1. Model Structure. The hybrid model structure is shown in Figure 5, and the main improvement is to use the features of the output layer of CNN to do classification by RF. First, the feature extraction of the image is done with the convolutional and pooling layers with random weights, and the extracted features are fed into the RF classifier to get the classification results. The number of filters in the convolutional layer greatly affects the generalization ability of the model; based on experience, the values of N1 and N2 of the model are 10 and 20, respectively.

In CNN, part of the image area (local perceptual field) is used as the input of the bottom layer of the network and then transmitted to each layer of the network in turn, and each layer is filtered by multiple filters to the most significant features which are computed in each layer through multiple filters. Because the local receptive fields of the image allow

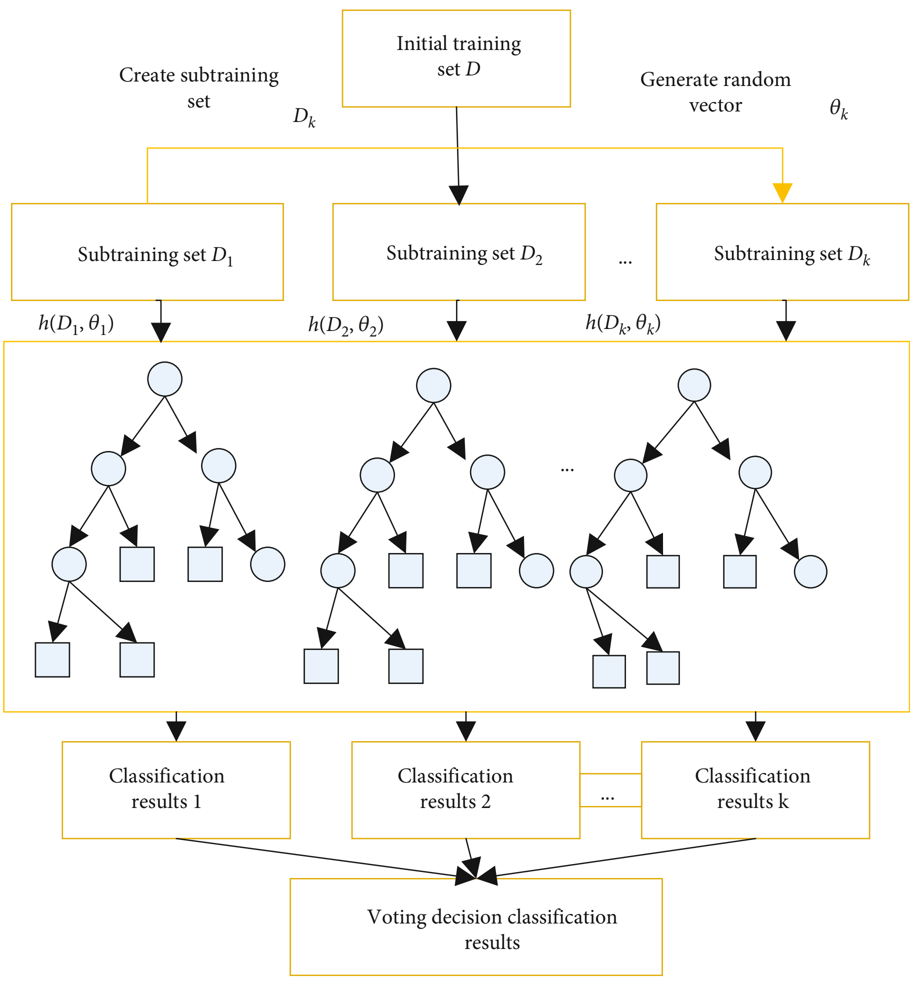

Figure 4: Random forest classifier flow.

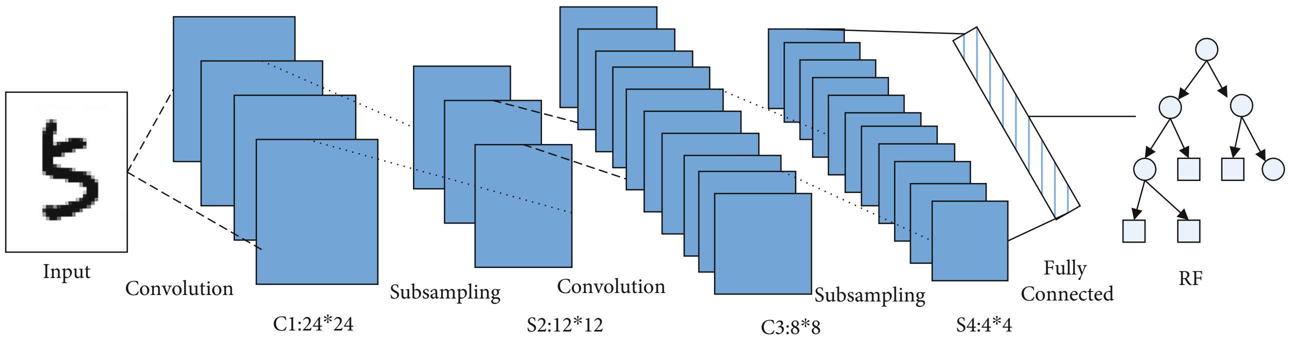

Figure 5: Hybrid model structure.

Input: sample set $D = (x_1 x_2, \cdots, x_m)$; number of split attributes.
Step1: Select n samples from the sample set D using Bootstrap sampling.
Step2: Randomly select k attributes and choose the best split attributes to build CART decision tree.
Step3: Repeat Step1 and Step2 for m times to build m CART decision trees.
Step4: Form a random forest with m CART trees, for the test set T decide which class the data belongs to by voting on the results, and the percentage that is different from the correct classification label is the classification error rate of RF.
Output: Random forest of m trees.

Algorithm 1

Table 1: Error rates (%) for RF and hybrid models on the MINST dataset.

Ntree	RF	Hybrid-RF
100	3.02	2.16
200	3.03	2.12
300	2.95	2.06
400	2.94	2.00
500	2.90	1.98
600	2.89	2.02
700	2.85	2.08
800	2.88	2.12

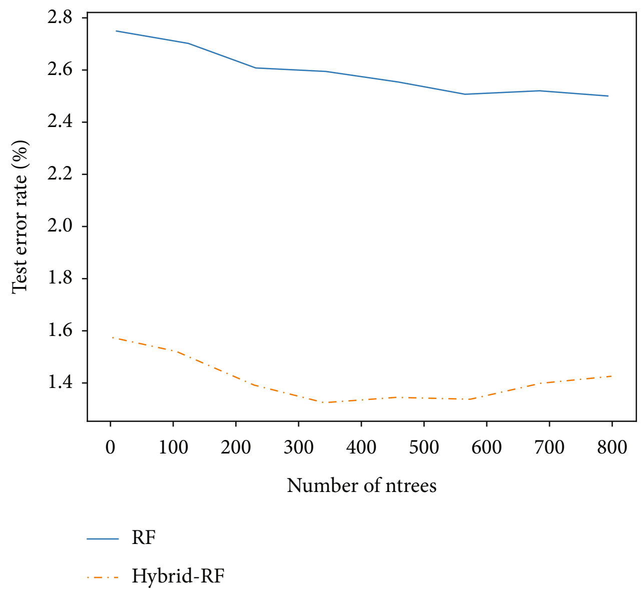

Figure 6: Comparison of the experimental results of RF and hybrid model on MNIST.

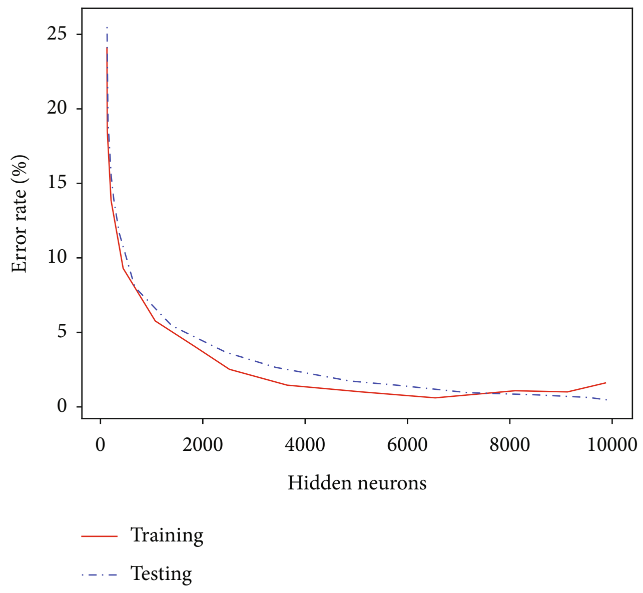

Figure 7: Results of ELM on MNIST dataset.

Table 2: Comparison of experimental results on MNIST dataset.

Datasets	Algorithms	Error rates (%)
MNIST	ELM	2.49
	DAE_s	2.14
	CKELM	3.18
	CNN-0	18.41
	CNN-1	2.01
	RF	2.89
	Hybrid-RF	1.97

the neurons to detect the most basic features of the image, such as edges or corners, the local receptive fields of the image are used as the input. The local perceptual field of the image allows the neurons to detect the most basic features of the image, such as edges or corners, and the neurons in each layer share the weights. A secondary pooling layer for extracting features is after each convolutional layer. This unique approach is able to obtain salient features for data that are invariant to translation, scaling, skew, and rotation.

Whether the features are designed manually or obtained by deep learning, everyone aims to obtain good features that reflect the nature of the original data, which is very much in line with the research intuition that using good features always leads to good results in various ways. ELM (extreme learning machine) does a random projection of the original data, projecting the original information into a certain space randomly, giving up the pursuit of good features for the improvement of the solution speed, and has obtained very good results in some tasks, so the classification of random features is worth exploring.

RF is a combinatorial classifier that solves the defect of decision tree overfitting and is more resistant to noise and anomalies; and it runs relatively fast and remains efficient for large amounts of data.

Our hybrid model exploits the advantages of CNN in feature extraction and RF in terms of speed and less overfitting and uses CNN with random weights to automatically extract the features and use it as the input of the RF classifier, which avoids the CNN consuming a lot of time in the training process, while obtaining better classification accuracy.

4.2. Training Process

4.2.1. Extraction of Features

Step 1. Network initialization and random initialization of weights and biases.

Step 2. Convolutional layer extracts features. For example, the convolution kernel is operated to obtain the output.

$$x_j^l = f\left(\sum_{i\in M} x_i^{l-1} * k_{ij}^l + b_j^l\right). \tag{18}$$

Here, M_j represents the set of input feature maps.

Table 3: Experimental results of the rotated MNIST dataset on RF and hybrid models.

Ntree	RF	Hybrid-RF
100	11.16	10.55
200	10.55	9.88
300	10.31	9.67
400	10.25	9.50
500	10.25	9.22
600	10.11	9.22
700	10.07	9.21
800	10.09	9.21

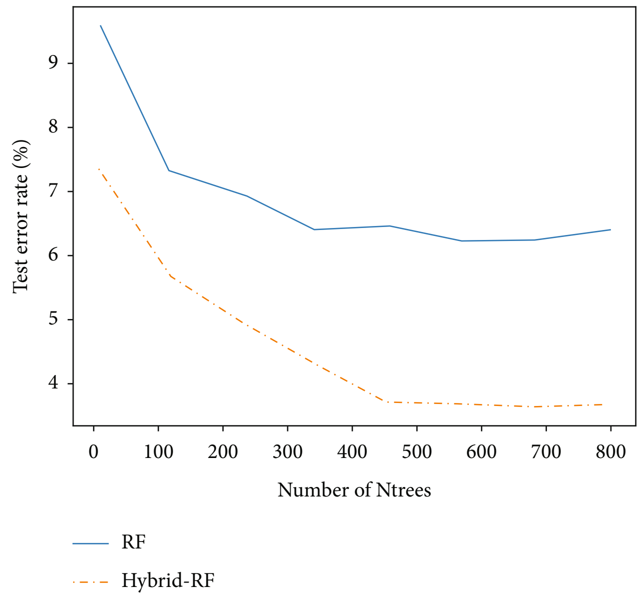

Figure 8: Comparison of the experimental results of RF and hybrid models on rotated MNIST.

Table 4: Comparison of experimental results on rotated MNIST dataset.

Algorithms	Error rates (%)
CNN-0	35.42
CNN-1	9.67
RF	10.02
Hybrid-RF	9.18

Step 3. The pooling layer extracts features. For the pooling layer, the number of feature maps remains the same, but the input feature maps are made smaller.

$$x_j^l = f\left(\beta_j^l \text{down}\left(x_i^{l-1}\right) + b_j^l\right), \tag{19}$$

where down(•) denotes a pooling function.

4.2.2. Replacement Classifier. The output of the C5 layer is used as the extracted features as the training set D and the test set T to build a random forest as follows.

5. Experiments and Results

To evaluate the classification performance of our hybrid model, we conducted experiments on the well-known MNIST and rotated MNIST datasets.

5.1. MNIST Dataset. Assuming that Ntree is the number of trees in RF, when Ntree is taken too small, the classification accuracy will not reach the desired result. Since RF is not prone to overfitting problem, we can make the value of Ntree as large as possible to ensure the classification accuracy, but it will take much time to build RF, so the value of Ntree has an important significance to the performance and complexity of RF. In order to avoid the problem of long training time of CNN, we use the method of random weights, after the CNN extracts the features, the extracted features are input to RF for classification; in order to compare, we do the experiments under different Ntree numbers. Table 1 shows the test error rate under different Ntree values.

From Table 1, it is seen that the classification accuracy of the hybrid-RF model is better than that of the RF at each Ntree.

Figure 6 shows the results of RF and hybrid-RF model, from which we can see that the test error rate of the hybrid model is better than RF at different Ntree.

Figure 7 shows the relationship between the classification error value and the number of hidden layer neurons of ELM (extreme learning machine) classifier on the MNIST dataset; the final training error is 0.66%; the test error is 2.47%.

Many methods have been experimented on the MNIST dataset, and Table 2 lists the performance of several different methods on the MNIST dataset, among which CKELM (convolutional extreme learning machine with kernel) is a convolutional neural network with random weights to extract the number of features, and the classifier is replaced with kernel extreme. The error rate on the MNIST dataset is 3.20%; the network structure of DAEs is $200 \times 200 \times 200$; CNN-0 is a convolutional neural network with random weight filtering kernels, because the weights are not adjusted, and the error rate that can be seen from the table in CNN-1 is the convolutional neural network after 50 training iterations.

Under the given hardware conditions, the CNN takes about 190 s for one iteration, and 50 iterations have consumed close to 3 hours. The RF itself runs very fast, taking only 20 minutes to train on the MNIST dataset (when Ntree = 400), and the hybrid model takes less time to train than the original RF because the dimensionality of the data is less than the original. Our model greatly reduces the time for feature extraction and the accuracy is guaranteed.

Due to the effectiveness of random weights, we use a CNN with random weights, which does not require gradient descent algorithm to adjust the parameters, also overcomes the problem of sensitive learning rate selection, and combines the advantages of fast and efficient RF, while the experimental results also prove the effectiveness of the hybrid model.

5.2. Rotated MNIST Dataset. To further illustrate the effectiveness of our model, we selected a rotated MNIST dataset from the variant MNIST dataset for the comparison test, which rotates the digital images in the MNIST dataset uniformly between 0 and 2π. Here, we randomly select 50,000 data from the rotated MNIST dataset as training data and 10,000 as test data and do the same experiments as the MNIST dataset for the RF and hybrid models with different Ntree numbers. Table 3 shows the error rates of the RF and hybrid models for the rotated MNIST dataset.

Figure 8 shows the experimental results of RF and our hybrid model in the rotated MNIST dataset, from which we can see that our hybrid model outperforms RF for different Ntree values, which again validates the effectiveness of our model and also has better generalization ability than RF.

Table 4 shows the comparison of the experimental results on the rotated MNIST dataset. CNN-0 is the CNN with random weights and CNN-1 is the CNN after 50 iterations of training.

6. Conclusion

For the image classification and recognition problem, we propose a hybrid model in this paper. In the hybrid-RF model, CNN extracts features with random weight and then completes the classification combined with RF. Therefore, the model greatly reduces the time spent in the process of feature extraction, effectively overcomes the problem of long CNN training time, and avoids the problem of manual feature selection of RF. Sufficient experimental results show that our proposed hybrid-RF model has superior performance and can effectively solve the problems of image classification and recognition. In the hybrid model, the features are extracted using CNN with random weights and then handed over to RF to complete the classification, which makes the model take much less time to extract features, overcomes the problem of long training time of CNN, and solves the drawback of manual feature selection by RF.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflicts of interest regarding this work.

References

[1] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 1, pp. 20–28, 2021.

[2] M. Xin and Y. Wang, "Research on image classification model based on deep convolution neural network," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, 11 pages, 2019.

[3] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *The Stata Journal*, vol. 20, no. 1, pp. 3–29, 2020.

[4] E. Kremic and A. Subasi, "Performance of random forest and SVM in face recognition," *International Arab Journal of Information Technology*, vol. 13, no. 2, pp. 287–293, 2016.

[5] A. Smith, "Image segmentation scale parameter optimization and land cover classification using the random forest algorithm," *Journal of Spatial Science*, vol. 55, no. 1, pp. 69–79, 2010.

[6] B. C. Ko, S. H. Kim, and J. Y. Nam, "X-ray image classification using random forests with local wavelet-based CS-local binary patterns," *Journal of Digital Imaging*, vol. 24, no. 6, pp. 1141–1151, 2011.

[7] J. Xia, N. Falco, J. A. Benediktsson, P. Du, and J. Chanussot, "Hyperspectral image classification with rotation random forest via KPCA," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 4, pp. 1601–1609, 2017.

[8] M. Han, X. Zhu, and W. Yao, "Remote sensing image classification based on neural network ensemble algorithm," *Neurocomputing*, vol. 78, no. 1, pp. 133–138, 2012.

[9] D. R. Nayak, R. Dash, and B. Majhi, "Brain MR image classification using two-dimensional discrete wavelet transform and AdaBoost with random forests," *Neurocomputing*, vol. 177, pp. 188–197, 2016.

[10] M. Li, S. Zang, B. Zhang, S. Li, and C. Wu, "A review of remote sensing image classification techniques: the role of spatio-contextual information," *European Journal of Remote Sensing*, vol. 47, no. 1, pp. 389–411, 2014.

[11] A. Ellahyani, M. El Ansari, and I. El Jaafari, "Traffic sign detection and recognition based on random forests," *Applied Soft Computing*, vol. 46, pp. 805–815, 2016.

[12] F. Zhu, L. Shao, and M. Lin, "Multi-view action recognition using local similarity random forests and sensor fusion," *Pattern Recognition Letters*, vol. 34, no. 1, pp. 20–24, 2013.

[13] H. Guan, J. Li, M. Chapman, F. Deng, Z. Ji, and X. Yang, "Integration of orthoimagery and lidar data for object-based urban thematic mapping using random forests," *International Journal of Remote Sensing*, vol. 34, no. 14, pp. 5166–5186, 2013.

[14] E. Adam, O. Mutanga, J. Odindi, and E. M. Abdel-Rahman, "Land-use/cover classification in a heterogeneous coastal landscape using RapidEye imagery: evaluating the performance of random forest and support vector machines classifiers," *International Journal of Remote Sensing*, vol. 35, no. 10, pp. 3440–3458, 2014.

[15] C. Hu, Y. Chen, L. Hu, and X. Peng, "A novel random forests based class incremental learning method for activity recognition," *Pattern Recognition*, vol. 78, pp. 277–290, 2018.

[16] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, "An assessment of the effectiveness of a random forest classifier for land-cover classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 67, pp. 93–104, 2012.

[17] P. Du, A. Samat, B. Waske, S. Liu, and Z. Li, "Random forest and rotation forest for fully polarized SAR image classification using polarimetric and spatial features," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 38–53, 2015.

[18] A. O. Ok, O. Akar, and O. Gungor, "Evaluation of random forest method for agricultural crop classification," *European Journal of Remote Sensing*, vol. 45, no. 1, pp. 421–432, 2012.

[19] Y. Zhang, G. Cao, X. Li, and B. Wang, "Cascaded random forest for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 4, pp. 1082–1094, 2018.

[20] Y. Ye, Q. Wu, J. Z. Huang, M. K. Ng, and X. Li, "Stratified sampling for feature subspace selection in random forests for high dimensional data," *Pattern Recognition*, vol. 46, no. 3, pp. 769–787, 2013.