remote sensing

MDPI

Article

SAR Target Classification Based on Deep Forest Model

Jiahuan Zhang [1,2], Hongjun Song [1,*] and Binbin Zhou [1,2]

[1] Department of Spaceborne Microwave Remote Sensing System, Aerospace Information Research Institute Chinese Academy of Sciences, Beijing 100190, China; zhangjiahuan15@mails.ucas.ac.cn (J.Z.); zhoubinbin15@mails.ucas.ac.cn (B.Z.)

[2] School of Electronics, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: hjsong@mail.ie.ac.cn

Received: 7 November 2019; Accepted: 31 December 2019; Published: 1 January 2020

Abstract: Synthetic aperture radar (SAR) has become one of the most important means of information acquisition in today's society and shows great potential in many fields. Target identification and classification of SAR images are also the focus of research. With the vigorous development of deep learning, many researchers apply this method to SAR target classification to obtain a more automatic process and more accurate results. In this paper, a novel deep forest model constructed by multi-grained cascade forest (gcForest), which is different from the traditional neural network (NN) model, is employed to classify ten types of SAR targets in the moving and stationary target acquisition and recognition (MSTAR) dataset. Considering that the targets of input images may be off-center and of different sizes in practical applications, two improved models based on varying weights by image features have been put forward, and both obtain better results. A series of experiments have been conducted to optimize model parameters, and final results with the MSTAR dataset illustrate that the two improved models are both superior to the original gcForest model. This is the first attempt to classify SAR targets using the non-NN model.

Keywords: synthetic aperture radar (SAR); target classification; deep forest; gcForest

1. Introduction

As an active microwave remote-sensing device, synthetic aperture radar (SAR) is capable of providing high-resolution images independent of weather conditions and sunlight illumination, and is of great application value in military reconnaissance, environmental monitoring, geological exploration, disaster prediction, and other fields [1–5]. High-resolution SAR images contain a wealth of detailed information, which can immensely widen its application areas [6–9]; however, this also makes SAR image interpretation extremely complicated. Unlike optical images, the same landform or targets in SAR images show completely different information characteristics, so successful classification algorithms for optical images cannot be used directly on SAR images. In addition, a lack of effective characteristics, geometric distortion, false targets, speckle noise, and other issues make SAR images become more difficult to understand [5,10,11]. From the above points, it becomes harder and harder to meet the growing application demand utilizing only regular means of image interpretation. In recent years, with the development of deep learning, increasingly intelligent methods have been applied to SAR image interpretation. Deep learning is regarded as a feature learning tool in which image data can be directly taken as input without additional manual image processing or other complex operations. It can independently extract features that best express the target from SAR image data, and realize image and target classification automatically and efficiently [12–20].

Over the past few years, many different methods of SAR image automatic interpretation have been proposed. Chen employed an unsupervised sparse auto-encoder in SAR target recognition, replacing the classical backpropagation algorithm, to obtain convolutional kernels used in convolutional neural networks (CNN) [21]. Geng and Fan proposed a new approach to extract image features named deep convolutional auto-encoder (DCAE), which can also perform target classification automatically [22]. Chen and Wang presented a new all-convolutional network to reduce overfitting due to being limited by a small number of classified training samples [23]. For the same purpose, Xue and Pei proposed a method based on a heterogeneous CNN ensemble which can reduce training samples and improve the recognition performance and generalization ability of the network [24]. Kang and He developed a classification method built on a multi-layer network and transfer learning [25]. Li divided the CNN into two parts to achieve automatic recognition: One is a convolutional auto-encoder (CAE) which is used to extract advanced features; the other is a classifier named shallow neural network (SNN), and this system can achieve a faster training speed than ever before [26]. Zhang and Song presented a fast training method for largescale SAR sample recognition based on CNN [27]. Zhao and Liu proposed the algorithm of a multi-stream convolutional neural network for SAR automatic target recognition (ATR), which integrated multi-view information from different angles of the same target, maximized the use of limited SAR image data, and improved the recognition performance [28]. Gao and Huang combined the deep CNN and support vector machine (SVM) to enhance the recognition of the moving and stationary target acquisition and recognition (MSTAR) database [29]. Jiang and Zhou put forward a robust recognition method via a hierarchical fusion of CNN and attributed scattering center (ASC) matching [30].

Most deep learning methods used for SAR target classification are based on neural network (NN) systems. NN systems can learn multi-layer non-linear relations of datasets by building multiple hidden layer models, which is beyond the reach of existing nonlearning-based approaches [31]. However, due to the inherent shortcomings of NN-style models, there are many problems in the application of SAR target classification for the following reasons. First, it is well known that most deep learning methods require a large amount of training data to obtain good network performance, while the open-labeled SAR image datasets are still limited because of the excessive cost of labeling. Second, the high computational and time cost in the training process of NN is a major inconvenience for practical usage. Third, network performance relies too much on hyper-parameters. These problems undoubtedly restrict the development of SAR image interpretation and need to be solved urgently. In this paper, a brand new nonNN model proposed by Zhou [32], named deep forest, which is implemented by a multi-grained cascade forest (gcForest), is applied to SAR target classification for the first time. This nonNN deep model consists of two major parts: multi-grained scanning and cascade forest. The latter provides the ability of representation learning through layer-by-layer structure and this ability can be further enhanced by the former, hence gcForest has the potential to be structurally aware. In addition, the deep forest needs much fewer hyper-parameters, and the structure is less dependent on hyper-parameter settings. The most important thing is that deep forest can automatically determine the number of training layers, so it has the ability to deal with small datasets just like SAR datasets and obtain a satisfactory result.

The remainder of this paper is organized as follows. In the second section, the deep forest algorithm and two proposed modified versions are introduced in detail. The third section uses the MSTAR dataset to verify the performance of the deep forest algorithm and the two adjusted algorithms. Then, the relevant results are compared with several methods proposed in articles published in recent years (e.g., Multi-Scale Convolutional Auto-Encoder (MSCAE) [33], Euclidean distance restricted auto-encoder (ED-AE) [34]). In Section 4, a detailed explanation regarding the technical aspects of the deep forest model is provided. The last section is a summary.

2. Methods

2.1. Preprocessing of MSTAR Images

Images from the MSTAR dataset used in this paper are collected by a high-resolution spotlight SAR operating in the X band with single polarization (HH). The images are slices of static targets. According to the SAR image characteristics, continuous strong reflection sources usually correspond to artificial targets [35]. The contrast between the targets and background can be used to identify such objects. However, due to the existence of clutters, the contrast between the targets and the background is usually too low to identify objects and details. So, it is necessary to adjust the images' contrast before they are input in order to classify. This paper carries out two steps of Gamma transformation preprocessing on the original images for enhancement.

$$I_{out}(x, y) = cI_{in}(x, y)^{\gamma} \tag{1}$$

Equation (1) represents a Gamma transformation of a greyscale image I. $I(x, y)$ is the intensity of the pixel at the (x, y) location of the input image; c and γ are positive constants. A Gamma transformation with a certain range of γ maps a narrow range of cramped input greyscale into a broader range of output greyscale, with the opposite being true for higher input. In other words, if $\gamma > 1$, low-intensity pixels are darkened, and vice versa when $\gamma < 1$. In this experiment, a Gamma transformation of $\gamma > 1$ will be done on the images in the first step. This operation can effectively highlight the area containing targets. Then, the obtained image will be dynamically divided according to its gray value to obtain a slice of the target area. Finally, a Gamma transformation with $\gamma < 1$ will be performed on the slice to improve the dark details of the target; this operation can also increase the image's contrast at the low gray level, which is more conducive to distinguishing the image details.

2.2. Introduction of the Deep Forest Construction Method

Multi-grained cascade forest (gcForest) is an approach to construct a deep forest [32]. It is a novel ensemble method with a cascade decision tree structure, which can enhance its representation learning ability. The deep forest is made up of two parts, multi-grained scanning and cascade forest, which will be introduced in the following part. Since gcForest is the main method to achieve a deep forest, gcForest will be mainly used instead of the name deep forest in the following section.

2.2.1. Multi-Grained Scanning

As shown in Figure 1, sliding windows are used to scan the input original images. The original d*d image is partitioned into n*n panes of m*m size, in which m is the side length of a sliding window and n can be computed as follows:

$$n = \frac{d - m}{p} + 1, \tag{2}$$

where p is the step size of sliding windows.

Then, n^2 instances are put into classifiers such as random forest that can obtain classification types. The classification probability of the input instances obtained by the classifier constitutes the first-stage feature vectors. Suppose there are k classes to classify, these k classes can constitute a k-dimensional vector, so each classifier gets n^2 k-dimensional first-stage feature vectors. The first-stage feature vectors of each classifier are sorted according to class k and rearranged into second-stage feature vectors with k n*n-dimensional as shown in Figure 2.

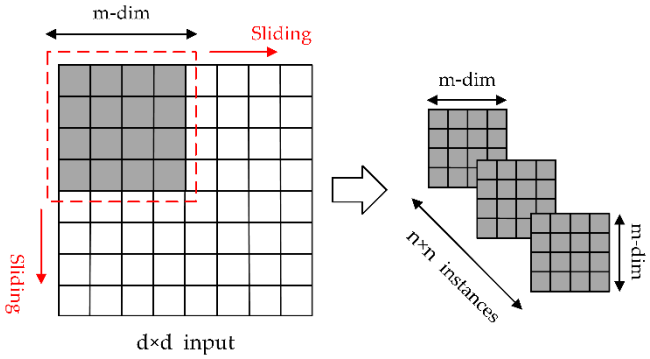


Figure 1. The process of extracting the original image features by sliding the pane.

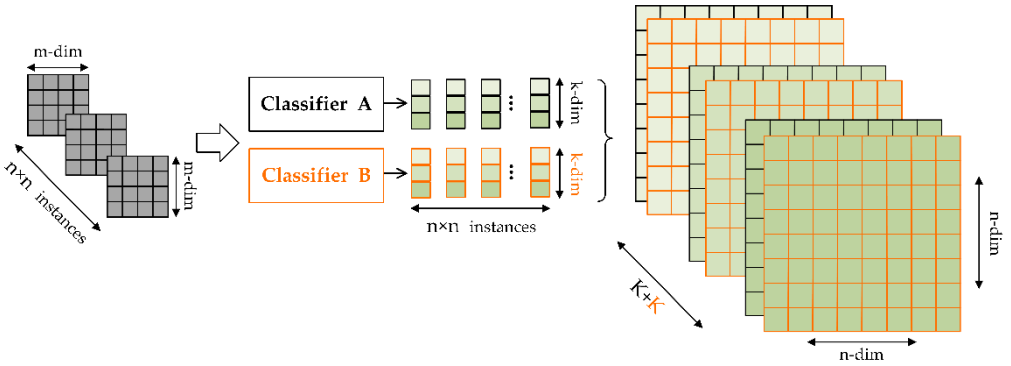


Figure 2. The formation and transformation of second-stage features vectors.

As shown in Figure 3, the second-stage feature vectors are pooled to obtain the final features vectors of the multi-grained scanning stage. The side length of the pooling block (framed by a yellow dotted line in Figure 3) is set as t, and the pooling block is composed of t^2 sliding blocks of the second-stage feature vectors. The final feature vectors can be obtained by averaging the prediction probability of the second-stage feature vectors within the pooling block.

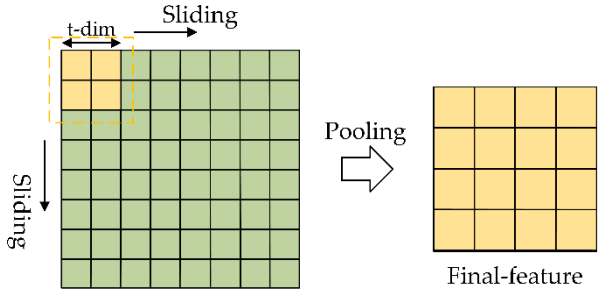


Figure 3. Pooling to obtain final feature vectors.

2.2.2. Cascade Forest

Cascade forest is a cascade structure, as illustrated in Figure 4. This layer-by-layer processing configuration improves the representation learning ability of deep forest. Each layer of the cascade is an ensemble of classifiers, and it receives the feature vectors (i.e., the classification probability) processed by the previous layer classifiers. Then these feature vectors are brought into the classifiers of the current layer for classification and the classification probability is output to the next layer as a result. Theoretically, classifiers used here can be of any form, such as random forest or logistic regression.

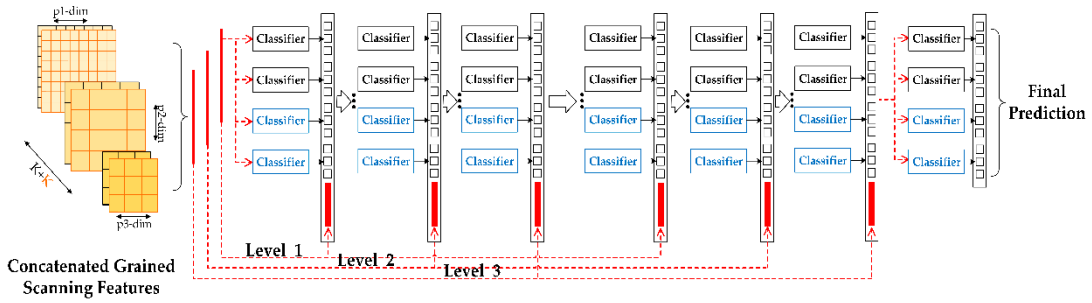


Figure 4. Process of the cascade.

To lower the risk of overfitting, the cascade forest uses *k*-fold cross-validation to generate every class vector. More concretely, each input instance will be used as training data *k*-1 times to generate *k*-1 class vectors, and the output class vector is produced by averaging these *k*-1 class vectors as enhanced features. After cross-validation, a new layer of cascade forest is generated. Since gcForest has a characteristic that automatically determines the number of cascade levels, the training procedure will be terminated if a new layer leads to almost no improvement. The overall framework of gcForest is elucidated in Figure 5.

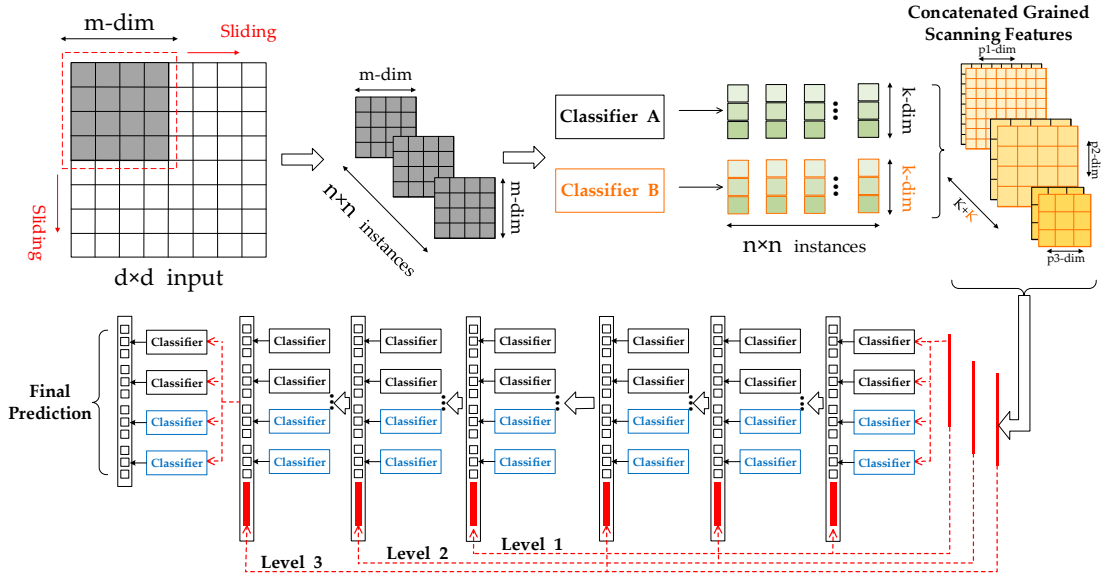


Figure 5. The overall structure of the multi-grained cascade forest (gcForest).

2.3. Optimized Deep Forest

To gather new image features, multi-grained scanning is used to assort sizes of sliding panes to deal with the original image as shown in Figure 6. Thus, the proportion of the object in each sliding pane is different, which affects the classification probability of second-stage feature vectors in the pooling block. Nevertheless, these influences are ignored in the pooling process. Such an average pooling process method is not accurate for samples with the target in the center, which will directly affect whether the generated features are representative enough or accurate enough. In order to overcome this problem, two improved methods for pool layer processing are proposed, which can effectively improve the classification accuracy without affecting the training speed. They are Euclidean distance weighted pooling (distance gcForest) and overlap degree weighted pooling (overlap gcForest).

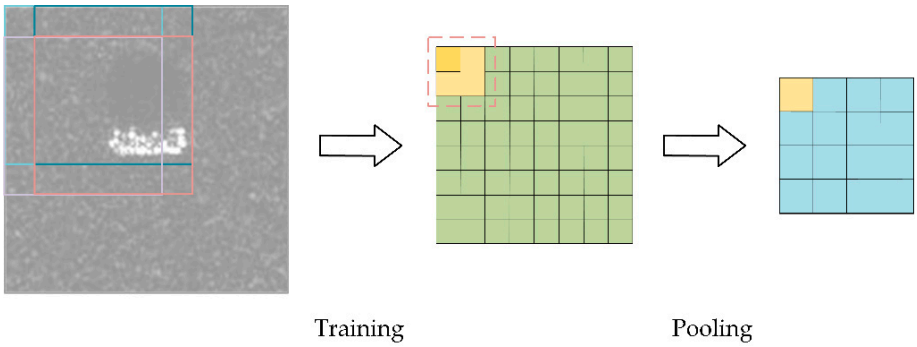
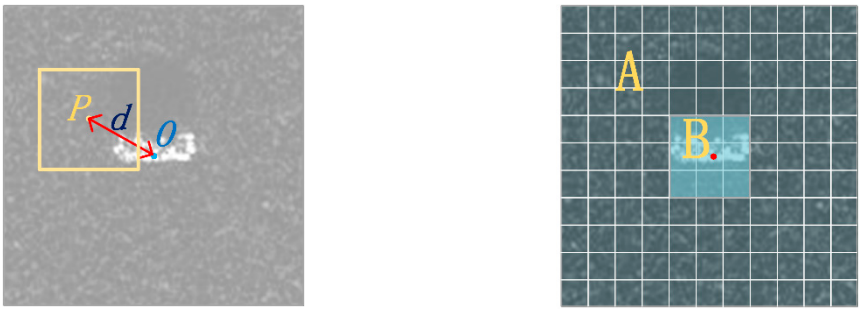


Figure 6. Feature evolution.

2.3.1. Distance gcForest

Distance gcForest measures the proportion of the target in the sliding pane in a relatively straightforward way by calculating the Euclidean distance d between O (the center of the image) and P (the center of the sliding block), as shown in Figure 7a. Assuming that the distance between P and O is less than some value and that the pool block will contain the entire target, there is no need to weight by distance. Accordingly, the second-stage feature vectors in pooling blocks are weighted differently according to the extent of d. The distance range is the minimum to maximum value of d, and this range is divided into two parts as described in the first column of Table 1. l is the side length of the input image and q is a parameter that can be adjusted according to the average proportion of the target to the input image. A and B are weighted areas shown in Figure 7b.



(**a**) (**b**)

Figure 7. Description of Euclidean distance pooling. (**a**) The distance between the center of the image and the center of the pane; (**b**) the weight range partition diagram.

Table 1. Weights for different distance panes.

d	Update Weights
$d > \frac{\sqrt{2}}{q}l$	$w_{d_i} = \frac{u_i}{Z_d} d_i^{-ke_i},\ i = 1, 2, \cdots, N$
$d < \frac{\sqrt{2}}{q}l$	$w_{d_i} = \frac{1}{t^2},\ i = 1, 2, \cdots, N$

Inspired by the weight updating mode of the Boosting classifier [36], the updated weight can be formulated as follows.

The initial weight distribution of each feature vector in the block is

$$W_o = (u_1, u_2, \cdots, u_i),\ u_i = \frac{1}{t^2},\ i = 1, 2, \cdots, N, \tag{3}$$

$$N = m \times n, \tag{4}$$

where m and n are the size of the pooling block (usually choose $m = n$). Then, the feature vector's updated weight distribution is

$$W_d = \left(w_{d_1}, w_{d_2}, \cdots, w_{d_i}\right), \tag{5}$$

$$w_{d_i} = \frac{u_i}{Z_d} d_i^{-ke_i},\ i = 1, 2, \cdots, N, \tag{6}$$

where k is an adjustable parameter, e_i is the misjudgment probability of a second-stage feature vector in the pooling block, and Z_d is the scaling factor,

$$Z_d = \sum_{i=1}^{N} u_i d_i^{-ke_i}, \tag{7}$$

which makes W_d a probability distribution.

In conclusion, the weighting mode divided by d is shown in Table 1.

2.3.2. Overlap gcForest

Overlap gcForest is a method that weights the second-stage feature vector in the pooling block according to the overlap degree. As shown in Figure 8, S_W is the cross area of the sliding pane and the target region (indicated by the yellow area), and S is the target area (indicated by the red box). The overlap degree I is defined as the ratio of S_W to S.

$$I = \frac{S_W}{S}. \tag{8}$$

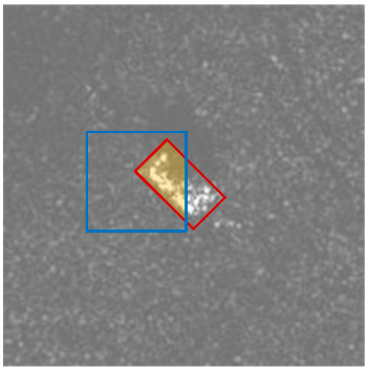


Figure 8. Description of overlap degree weighted pooling.

Similar to the last section, the initial weight u_i is the average number of panes in the pooling block and the updated overlap degree weight distribution is

$$W_I = \left(w_{I_1}, w_{I_2}, \cdots, w_{I_i}\right), \tag{9}$$

$$w_{I_i} = \frac{u_i}{Z_I} I_i^{-ke_i},\ i = 1, 2, \cdots, N, \tag{10}$$

where k is an adjustable parameter and Z_I is the scaling factor,

$$Z_I = \sum_{i=1}^{N} u_i I_i^{-ke_i}, \tag{11}$$

which makes W_I a probability distribution.

3. Results

3.1. MSTAR Dataset

The experimental data used in this paper are provided by the SAR sensor of Sandia national laboratory, collected by a spotlight SAR with 0.3 m × 0.3 m resolution. Defense Advanced Research Projects Agency and the Air Force Research Laboratory co-hosted the data collection, which is a vital part of the MSTAR project. The project gathered hundreds of thousands of SAR images, including target types, azimuth, pitch, barrel turning, and contour configuration changes of various ground military targets.

The dataset used in this paper includes 10 different classes of ground military vehicles, which are 2S1 (self-propelled howitzer), BMP2 (infantry fighting vehicle), BRDM2 (armored reconnaissance vehicle), BTR60 (armored transport vehicle), BTR70 (armored transport vehicle), D7 (bulldozer), T62 (tank), T72 (tank), ZIL131 (cargo truck), and ZSU234 (self-propelled anti-aircraft gun) as shown in Figure 9. The same type of target in the training set and the test set has the same model number, but their imaging elevation angle and azimuth angle are different. The training SAR image was collected at 17 pitch angles and the test SAR image was collected at 15 pitch angles, a difference of 2 pitch angles that can be considered negligible. The sample numbers of each target type in the dataset are different and are listed in Table 2. The initial size of samples is 128 * 128, and then images are cut into size 96 * 96 to reduce the amount of computation.

Figure 9. Ten classes of ground military targets' optical and synthetic aperture radar (SAR) images.

Table 2. Ten classes of moving and stationary target acquisition and recognition (MSTAR) targets and their sample numbers.

	Training Set	Test Set
2S1	299	274
BMP2	233	195
BRDM2	298	274
BTR60	256	195
BTR70	233	196
D7	299	274
T62	298	273
T72	232	196
ZIL131	299	274
ZSU234	299	274
Total	2746	2425

3.2. Classification Results of Optimized gcForest

The hardware environment producing these experiment results in this paper is a computer using an Intel Core I5-6500 CPU of 3.20 GHz with a 64-bit operating system and 24 GB of internal storage. It is worth noticing that the running efficiency of gcForest and optimized gcForest is good without GPU acceleration.

Table 3 shows the confusion matrix for the classification results of the MSTAR dataset using the parameter optimized gcForest, which is described in Section 4. The rows in the confusion matrix

represent the actual target category, and the columns represent the experimentally predicted category. The experiment results demonstrate that deep forest can obtain high classification accuracy in most categories. It is feasible to apply deep forest to SAR target classification, but the algorithm still needs further improvement. The classification accuracies of BMP2 and BTR60 are not satisfactory, and other categories are prone to be misconceived as T72 and ZIL131. The main reason for these cases is that different incident angles of the electromagnetic wave have different contributions to the final imaging results, making the same vehicle object look completely different.

Table 3. Confusion matrix of the MSTAR dataset using gcForest.

	2S1	BMP2	BRDM2	BTR60	BTR70	D7	T62	T72	ZIL131	ZSU234	Recall
2S1	261	3	3	4	0	1	1	0	0	1	95.26%
BMP2	0	170	2	9	2	0	0	12	0	0	87.18%
BRDM2	0	0	268	0	0	0	0	0	1	5	97.81%
BTR60	0	1	4	173	4	0	0	2	11	0	88.72%
BTR70	0	0	0	1	193	0	0	2	0	0	98.47%
D7	0	0	0	0	0	271	0	0	3	0	98.90%
T62	0	0	0	0	0	0	271	0	1	1	99.27%
T72	0	1	2	0	0	0	0	193	0	0	98.47%
ZIL131	0	0	0	0	0	1	0	0	272	1	99.27%
ZSU234	0	0	0	0	0	1	0	0	0	273	99.63%
Avg.						96.70%					

3.3. Effect of Optimized gcForest and Improved gcForests

The MSTAR dataset is then used to evaluate the performance of two improved methods. The first four rows in Table 4 present the classification results for each class using several kinds of gcForests, where the initial gcForest is the model using the original parameters. Comparing the results of the initial gcForest and optimized gcForest, it can be found that the adjustment of parameters has greatly improved the classification accuracy of gcForest. This part will be explained in detail in the next section. The distance and overlap gcForests using optimized parameters further improve the classification accuracy, and the latter one improved BMP2 and BTR60 significantly.

Table 4. Detailed comparison of the classification performance under different methods.

	2S1	BMP2	BRDM2	BTR60	BTR70	D7	T62	T72	ZIL131	ZSU234	Avg.
Optimized gcForest	95.26%	87.18%	97.81%	88.72%	98.47%	98.90%	99.27%	98.47%	99.27%	99.64%	96.70%
Distance gcForest	94.16%	85.64%	97.81%	90.77%	98.47%	99.64%	99.27%	98.98%	99.27%	99.64%	96.74%
Overlap gcForest	94.53%	88.21%	97.45%	91.28%	97.96%	98.91%	98.90%	98.47%	99.27%	99.64%	96.78%
Initial gcForest	56.57%	80.00%	71.17%	84.10%	90.31%	87.23%	63.74%	90.31%	99.64%	99.64%	81.77%
MSCAE	97.44%	85.64%	94.16%	99.48%	97.44%	98.91%	99.49%	92.67%	99.27%	99.64%	96.54%
ED-AE	93.80%	87.90%	96.72%	91.79%	92.86%	98.91%	94.14%	79.55%	94.53%	99.64%	91.29%
NMF	100%	91.01%	97.91%	94.87%	97.87%	98.32%	96.38%	94.70%	97.27%	95.54%	94.20%
MSR	88.1%	76.02%	78.2%	97.8%	97.1%	99.3%	96.9%	99.3%	97.4%	99.6%	92.79%
Riemannian Manifolds	88.3%	96.9%	82.1%	98.5%	96.7%	99.3%	99.6%	97.8%	99.6%	99.6%	94.88%

Table 4 also lists the results of comparative methods, namely, MSCAE, Non-negative Matrix Factorisation (NMF) [37], ED-AE, Sparse Representation of Monogenic Signal (MSR) [38], and Riemannian Manifolds [39]. The training results obtained by the comparison approach in this paper are all based on the dataset without artificial expansion. It is not hard to find that the classification accuracies of the optimized and the two improved gcForests are higher than the other algorithms.

4. Discussion

This section discusses the influence of classifiers and parameter selection on multi-grained scanning and cascade forest respectively, based on the working process of gcForest, and then selects the most suitable training parameters for the MSTAR dataset.

The default parameters of gcForest are listed in Table 5. The training and test set classification accuracies are 98.18% and 81.77%. Training time is divided into two parts: Multi-grained scanning costs 567,858 s, cascade takes 126,179 s, and the total time consumed is 8 days, 47 min, and 27 s. The accuracies using initial parameters are not very ideal and the training is inefficient, there is still a long way to go for a specific category of classification tasks. Therefore, it is necessary to adjust the parameters to achieve a better classifier performance and shorter time cost. Not only that, comparison and analysis of the results can further help to understand the working process of gcForest and prove the feasibility and superiority of the proposed method.

Table 5. Initial experimental parameter setting.

gcForest Models		Parameter Setting
Multi-Grained Scanning	Number of Classifiers	2
	Types of Classifiers	ExtraTreesClassifier RandomForestClassifier
	Condition to Stop Growing Trees	Tree depth reaches 500
	Size of Sliding Panes	12, 24, 48
Cascade	Number of Classifiers	8
	Types of Classifiers	XGBClassifier RandomForestClassifier ExtraTreesClassifier LogisticRegression
	Condition to Stop Growing Trees	Tree depth reaches 500

4.1. Factors Affecting Multi-Grained Scanning

This stage involves feature extraction, which enhances the ability of the network to process input image data. The quality of feature extraction directly affects the accuracy and performance of the model, and it also influences the training time of the whole network. The following subsections explain the influence of the sliding pane size, classifier type and classifier parameters on accuracy.

4.1.1. Size of the Sliding Pane

In this experiment, ExtraTreesClassifier is used for classification to observe the influence of the change of the pane size on the accuracy and training time.

The size of the sliding pane determines the number and completeness of features. Sliding panes that are too small can cause the target to be too scattered, with most of them not even having parts associated with the target. Experiment results listed in Table 6 demonstrate that a small-pane size not only leads to a decrease in accuracy, but also significantly reduces training efficiency. Too large a pane size may contain lots of useless background information, which also reduces training accuracy to some extent but has little effect.

Table 6. Effects of different sliding panes sizes on accuracy and training time.

	12 * 12	**24 * 24**	**36 * 36**	**48 * 48**	**60 * 60**	**72 * 72**	**84 * 84**	**90 * 90**	**96 * 96**
Training Set	57.74%	69.92%	78.08%	83.72%	84.71%	86.72%	89.15%	90.05%	89.15%
Test Set	29.44%	36.54%	45.24%	57.86%	63.55%	68.82%	76.54%	80.54%	79.55%
Training Time (s)	97,024	89,666	63,686	55,896	40,255	11,122	1504	399	19

4.1.2. Type of Classifiers

In theory, gcForest can use any kind of classifier, and it imports various kinds of base classifiers to enhance model diversity. Unlike the original setting, this experiment adds LogisticRegression at the multi-grained scanning stage, the sliding pane size is set as 84, and the parameter settings of the classifiers are formulated in Table 7.

Table 7. Original experimental parameter setting of multi-grained scanning.

Classifier Types	Parameter Setting
RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100, min_samples_leaf = 10
ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100, min_samples_leaf = 10
LogisticRegression	Penalty = 'l2', tol = 0.0001

Table 8 illustrates the accuracy of each classifier. RandomForestClassifier and ExtraTreesClassifier achieve high accuracy on the training set, but the accuracies of their test sets are slightly lower, while LogisticRegression has better accuracy in both sets. However, in the multi-grained scanning phase, LogisticRegression significantly increases the memory footprint and reduces the training speed, so it will not be used on a large scale.

Table 8. Accuracy using different classifiers.

Classifier Types	Training Set	Test Set
RandomForestClassifier	88.64%	72.70%
ExtraTreesClassifier	87.98%	76.62%
LogisticRegression	86.13%	85.65%

4.1.3. Size of Classifier

The parameter adjustment of the classifier mainly focuses on the number of estimators and the max depth, which will affect the classification accuracy and training speed. Five groups of parameters are used in this experiment to observe their effects. The parameters are adjusted up and down proportionally centered on n_estimators = 500, max_depth = 100. Table 9 illustrates their influence on accuracy, and Figure 10 shows the corresponding time consumption.

Table 9. Accuracy of RandomForestClassifier and ExtraTreesClassifier with different sizes.

Parameter Setting	RandomForestClassifier		ExtraTreesClassifier	
	Training Set	Test Set	Training Set	Test Set
n_estimators = 100, max_depth = 20	88.20%	72.49%	88.60%	76.21%
n_estimators = 250, max_depth = 50	87.91%	72.78%	89.29%	76.26%
n_estimators = 500, max_depth = 100	88.93%	72.91%	89.15%	76.54%
n_estimators = 600, max_depth = 120	88.35%	72.78%	87.87%	76.78%
n_estimators = 750, max_depth = 150	89.15%	72.82%	88.75%	76.41%

The results show that for the same classifier, the variation of parameters has little effect on the classification accuracy, but the accuracies of different classifiers using the same parameter are quite different. ExtraTreesClassifier takes less time and performs better than RandomForestClassifier. On the premise of ensuring the classification accuracy, especially that of the test set, the final model in this stage selects a parameter that uses less training time. Finally, RandomForestClassifier and ExtraTreesClassifier choose n_estimators = 500, max_depth = 100 and n_estimators = 600, max_depth = 120 as their parameters, respectively.

Combined with the previous discussion, the classifiers selection and their parameters settings of the multi-grained scanning stage are listed in the following Table 10. As can be seen from Table 17 in Section 4.3, the improvement of multi-grained scanning plays a significant role in shortening the training time.

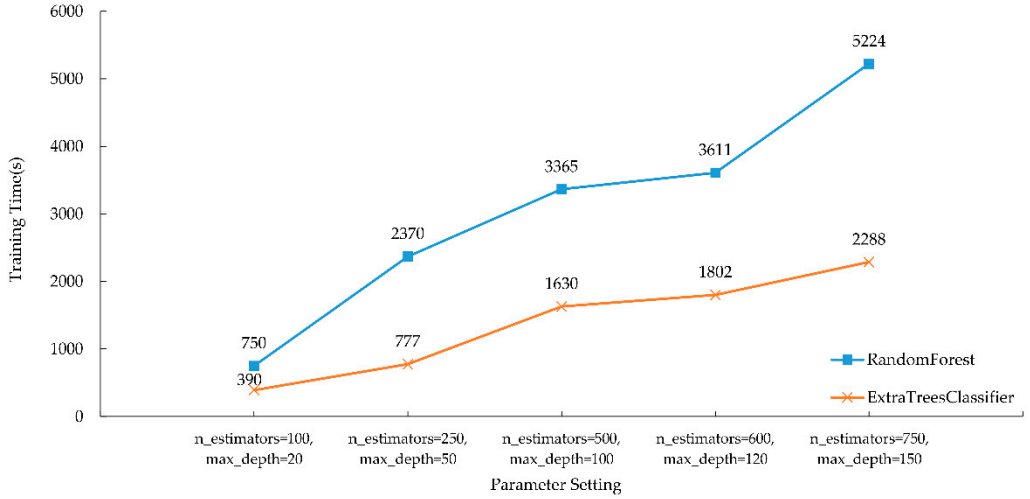


Figure 10. The training time of estimators with different size.

Table 10. Final configuration of multi-grained scanning.

Classifier Types	Parameter Setting	Pane Size
ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 600, max_depth = 120	72, 84, 90, 96
RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100	72, 84, 90, 96
LogisticRegression	Penalty = 'l2', tol = 0.0001	84, 90

4.2. Factors Affecting Cascade

Cascade forest carries the predictive results trained by multi-grained scanning as a new extracted feature for training classification. This section discusses the effects of the type, parameter setting, and quantity of classifiers inside the cascade forest in detail.

4.2.1. Type of Classifier

Based on the ensemble learning theory, the introduction of individual classifiers with differences can improve the overall performance of the network which is also the foundation of the gcForest algorithm. Therefore, a new classifier SGDClassifier is introduced to improve the performance of gcForest. The parameter setting is shown in Table 11.

The average classification accuracies of each layer are shown in Table 12. The accuracies of built-in classifiers are listed in the first five lines, and the last line is that of gcForest configured with these parameters. Figures 11 and 12 records the changes in the accuracy of the training set and test set at each layer of the built-in classifiers and gcForest.

Table 11. Initial experimental parameter setting of the cascade forest.

Classifier Types	Parameter Setting
ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100
RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100
XGBClassifier	base_score = 0.5, booster = 'gbtree', max_depth = 5, learning_rate = 0.1, n_estimators = 500, objective = 'multi: softprob'
LogisticRegression	Penalty = 'l2', tol = 0.0001
SGDClassifier	Penalty = 'l2', loss = 'modified_huber'

Table 12. Mean classification accuracy of gcForest and built-in classifiers.

Classifier Types	Training Set	Test Set
ExtraTreesClassifier	98.82%	94.88%
RandomForestClassifier	98.77%	94.60%
XGBClassifier	98.62%	95.20%
LogisticRegression	98.80%	95.52%
SGDClassifier	97.41%	95.96%
gcForest	98.81%	95.52%

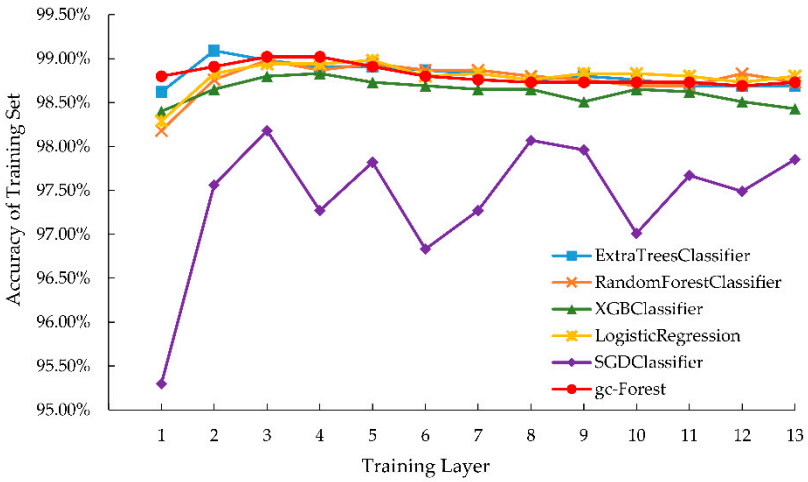


Figure 11. Training set classification accuracy of gcForest and its built-in classifiers in each layer.

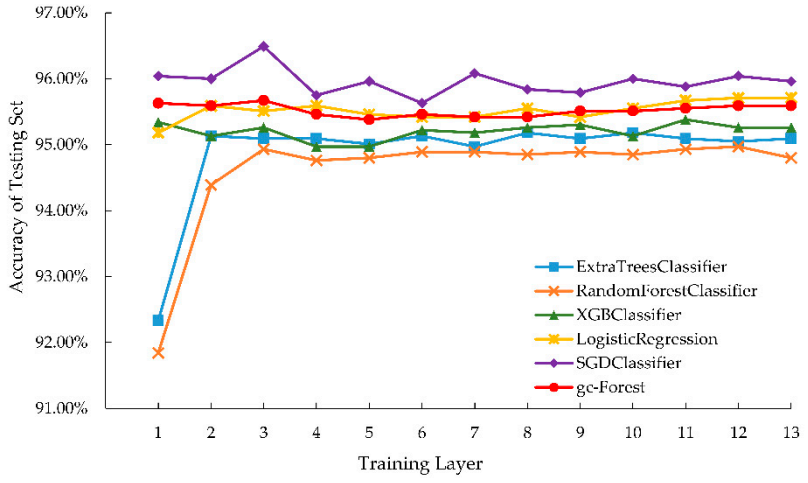


Figure 12. Test set classification accuracy of gcForest and its built-in classifiers in each layer.

During the training stage, the performance of the SGDClassifier is not as good as the other four basic classifiers, but in the test stage, the accuracy of the SGDClassifier is the best. However, two classifiers, ExtraTreesClassifier and RandomForestClassifier, that achieved better performances in the training stage turned out to be worse in testing. GcForest integrates these five kinds of classifiers and achieves high accuracy and stable performance in both stages.

On account of the classification accuracies of ExtraTreesClassifier and RandomForestClassifier falling short of the other three classifiers and being lower than that of gcForest, an experiment was carried out using a new classifier combination. It used each of the above-mentioned classifiers, except for ExtraTreesClassifier and RandomForestClassifier. The gcForest accuracy using this combination was 98.43% in the training set and 95.84% in the test set. This result is just slightly better than the result above, which uses a network without optimization. To some extent, the result also shows that increasing the diversity of classifiers in gcForest is beneficial to improving its performance. The classifiers used in the network support each other and play a complementary role.

4.2.2. Parameter Selection of Built-In Classifiers

This part debates the effects of classifier size and loss function selection on experiment results from three perspectives, respectively.

i. Size of Two Forest Classifiers

Similar to the experiment in the previous section, several combinations of parameters are selected to compare their effects on the test set. Their influence on classification accuracy is shown in Figures 13 and 14, where different numbers of estimators are represented by different colors, their corresponding descriptions are listed below, and the max depths corresponding to estimators' number are listed on the Y-axis. These parameter settings have a slight effect on accuracy, and the training time increases as the estimator number and max depth increase. Except for the first two sets of parameters, the average costing time per layer of RandomForest is 33.4 s and ExtraTreesClassifier is 14.2 s. These results indicated that ExtraTreesClassifier performs quickly. In consideration of time consumption and slight differences in accuracy, RandomForestClassifier selects max_depth = 100, n_estimators = 600 and ExtraTreesClassifier selects max_depth = 100, n_estimators = 500 as the model parameters in the following experiments.

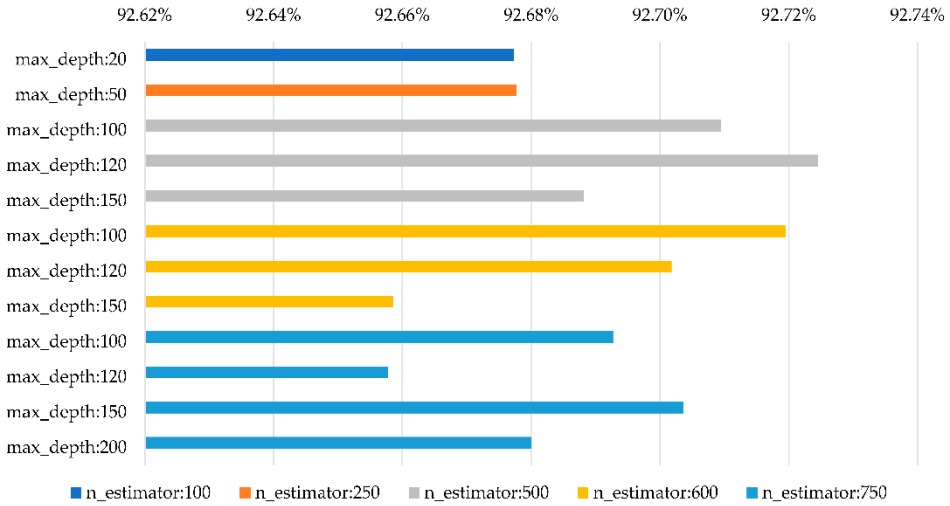


Figure 13. Test set classification accuracy of RandomForestClassifier with different sizes.

ii. Size of XGBClassifier

The performance verification of the XGBClassifier is divided into three groups according to the numbers of estimators, which are 500, 750, and 1000, and they are distinguished by three colors in Figure 15. For each group, there are three kinds of max depth for the estimators, which are 5, 6, and 10.

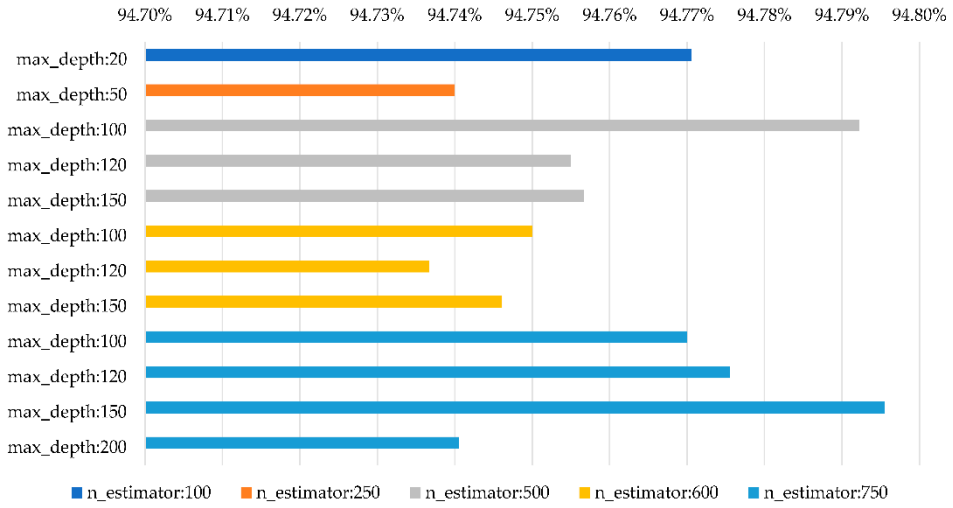


Figure 14. Test set classification accuracy of ExtraTreesClassifier with different sizes.

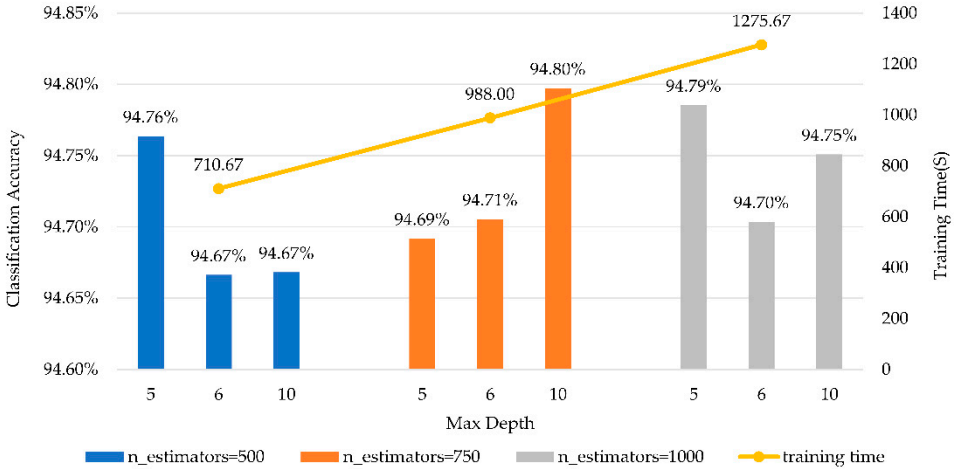


Figure 15. Accuracy and time cost for the test set using XGBClassifier with different sizes.

Figure 15 shows the max depth required for each set of estimators to obtain the best classification results; the bar chart shows the accuracy and the line chart represents the average training time. The training time very closely relates to the number of estimators used in XGBClassifier; classifiers with the same estimator number but different max depths take almost the same amount of time. Therefore, the training time of the classifier with the same number of estimators is averaged to represent the time cost. It is not hard to find from the composite structure diagram that classification accuracy can be improved by using more estimators, but the more estimators are used, the more time it takes. Considering the accuracy and time consumption, the alternative parameters are chosen as max_depth = 10, n_estimators = 750 and max_depth = 5, n_estimators = 500.

iii. Loss Function of SGDClassifier

SGDClassifier is a simple and effective classification method. Since gradient descent is extremely sensitive to the range of data, it is necessary to adjust its parameters. In this experiment, the loss functions are chosen as *modified_huber* and *log*. *Modified_huber* is a smooth hinge loss function, where the hinge is a linear support vector machine, and *log* is a logistic regression function. The loss function *modified_huber* only updates the model parameters when the sample violates the marginal constraint. Table 13 shows that the classification accuracy of *modified_huber* is not as high as that of *log* in both datasets, so *log* is selected as the final loss function of the SGDClassifier.

Table 13. The average training precision of different loss functions of the SGDClassifier.

Parameter Setting	Training Set	Test Set
loss = 'modified_hube'	95.43%	95.23%
loss = 'log'	97.42%	95.37%

4.2.3. Number of Classifiers

It is found from the structure of gcForest that the type and number of classifiers used in the cascade forest affect the results. The accuracy and generalization performance of the gcForest will improve with the increase in the number of individual classifiers inside the model, but too many individual classifiers will lead to a decrease in the diversity among classifiers, which will lead to a performance decline and magnified calculation of the algorithm. In addition, the number of cascade forest classifiers is small, so it produces fewer feature vectors, which leads these features to be overtaken by large quantities of features generated at multi-grained scanning. Therefore, it is particularly important to choose the number and types of classifiers appropriately.

The analysis in the previous two sections provides a reliable basis for us to make targeted adjustments for the appropriate number and types of classifiers. XGBClassifier performs better in gcForest, so its number is raised during the base classifier selection. The choices of the number of each classifier are indicated in Table 14, their parameter selections are listed in Table 15, and the classification accuracies of these different combinations are illustrated in Figure 16.

Table 14. Selection of the number and combination of different classifiers.

Classifier Type	Classifier Number				
	5	**6**	**8**	**9**	**10**
ExtraTreesClassifier	1	1	1	1	1
RandomForestClassifier	1	1	1	1	1
XGBClassifier (max_depth = 10, n_estimators = 750)	1	1	1	2	2
XGBClassifier (max_depth = 5, n_estimators = 500)	0	1	1	1	2
LogisticRegression	1	1	2	2	2
SGDClassifier	1	1	2	2	2

Table 15. Optimized classifier selection and its parameter settings.

Classifier Type	Parameter Setting
ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100
RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 600, max_depth = 100
XGBClassifier	base_score = 0.5, booster = 'gbtree', max_depth = 10, learning_rate = 0.1, n_estimators = 750, objective = 'multi: softprob'
XGBClassifier	base_score = 0.5, booster = 'gbtree', max_depth = 5, learning_rate = 0.1, n_estimators = 500, objective = 'multi: softprob'
LogisticRegression	Penalty = 'l2', tol = 0.0001
SGDClassifier	Penalty = 'l2', loss = 'log'

In order to reach the optimal outcome under the condition of any data size, gcForest will automatically determine its number of training layers. Therefore, the overall training time of gcForest cannot be determined in advance, which is not the focus of attention, so only the average training time of each layer is recorded here.

The results prove that a suitable parameter selection can greatly improve classification accuracy. Moreover, an appropriate increase in the number of XGBClassifier used in cascade forest improves the

classification accuracy to a certain extent, but at the same time, this increase in number also leads to an increase in the training time per layer.

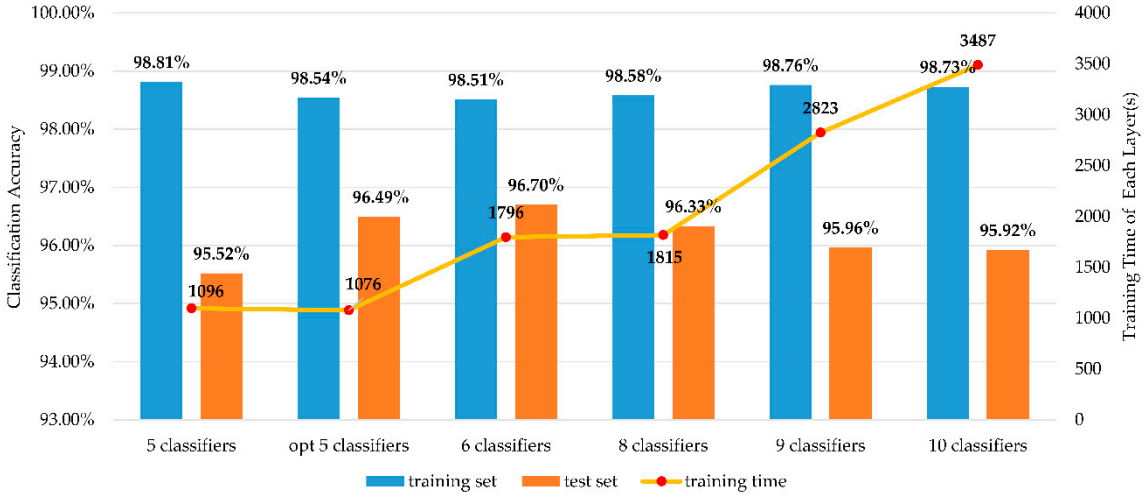


Figure 16. The accuracy and per layer training time of different combinations.

4.3. Results Contrast

The final parameters used in the gcForest model are listed in Table 16. Tables 4 and 17 show the classification results and training time comparison before and after optimization. It can be seen in Table 4 that after parameter optimization, the classification accuracy has been greatly improved. Furthermore, the classification of augmented MSTAR datasets using optimized and improved gcForests is also more accurate than the methods proposed in recent years.

Table 16. Optimized experimental parameter setting.

gcForest Models			Parameter Setting
Multi-Grained Scanning	Number of Classifiers		3
	Size of Sliding Panes		72, 84, 90, 96
	Types of Classifiers	ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 600, max_depth = 120
		RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100
		LogisticRegression	Penalty = 'l2', tol = 0.0001
Cascade	Number of Classifiers		6
	Types of Classifiers	ExtraTreesClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 500, max_depth = 100
		RandomForestClassifier	Criterion = 'gini', max_features = sqrt(n_features), n_estimators = 600, max_depth = 100
		XGBClassifier	base_score = 0.5, booster = 'gbtree', max_depth = 10, learning_rate = 0.1, n_estimators = 750, objective = 'multi: softprob'
		XGBClassifier	base_score = 0.5, booster = 'gbtree', max_depth = 5, learning_rate = 0.1, n_estimators = 500, objective = 'multi: softprob'
		LogisticRegression	penalty = 'l2', tol = 0.0001
		SGDClassifier	penalty = 'l2', loss = 'log'

Table 17. Time consumption of each stage and each layer.

Cost Time (s)	Initial	Optimized	Distance	Overlap
Multi-Grained Scanning (Total)	567,868	67,169	67,169	67,169
Cascade (Total)	126,179	37,954	56,975	56,323
Cascade (Each layer)	9801	1796	1859	1922
gcForest (Total)	694,047	105,123	124,144	123,492

The gcForest terminates its training automatically, which means that the layer number and training time of the cascade forest are uncertain. On this account, although the total time consumption of cascade forest needs to be paid attention to, it is more important to pay attention to the time consumption of each layer. Since the optimized gcForest, distance gcForest, and overlap gcForest use the same features generated by multi-grained scanning, they have the same training time at multi-grained scanning. Although the time consumption of the two optimization algorithms is slightly increased, it brings an improvement in classification accuracy, especially for the two categories of BMP2 and BTR60.

5. Conclusions

In this paper, a novel deep forest method implemented by the gcForest is used to explore the field of SAR target classification. This is the first time that such an experiment applied this method to SAR target classification and achieved satisfactory results. At present, most NN classification methods need to expand the dataset by means of rotation, stretching, and arbitrary clipping to improve their classification accuracy, but the experimental results in this paper use no artificially extended dataset. The gcForest is not sensitive to hyper-parameters and can achieve a good classification effect only by using the original parameters. Of course, as proved by the experiments presented in this paper, a more suitable selection of parameters will improve the classification accuracy for the specific classification task. Compared with NN methods, gcForest's calculation scale is much smaller, the training time is much shorter, and it is much easier to understand the working process of the network. The gcForest model can automatically determine the optimal number of training layers and terminate the classification process. In addition, this paper proposes two preliminary optimized methods, distance gcForest and overlap gcForest, aiming at data features for further improvement, and some results were achieved. Compared with several classification experiments without dataset enlargement, the optimized gcForest and improved gcForests achieve the best accuracy. In future studies, more targeted improvement schemes will be proposed for SAR target characteristics and applied to a wider range of extension conditions.

Author Contributions: Methodology, J.Z.; software, J.Z. and B.Z., validation, J.Z. and B.Z.; investigation, J.Z.; resources, H.S.; writing—original draft preparation, J.Z.; writing—review and editing, J.Z. and B.Z.; visualization, J.Z.; supervision, H.S.; project administration, H.S.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the foundation item of the National Key R&D Programme of China [2017TFB0502700].

Acknowledgments: Many thanks to Yucong Ma for her help in this part of the experiment code.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sacha, J.R. Automatic target recognition in acoustics: An overview. *J. Acoust. Soc. Am.* **2002**, *112*, 2306. [CrossRef]
2. Moreira, A.; Prats-Iraola, P.; Younis, M.; Krieger, G.; Hajnsek, I.; Papathanassiou, K.P. A tutorial on synthetic aperture radar. *IEEE Geosci. Remote Sens. Mag.* **2013**, *1*, 6–43. [CrossRef]
3. El-Darymli, K.; Gill, E.W.; McGuire, P.; Power, D.; Moloney, C. Automatic target recognition in synthetic aperture radar imagery: A state-of-the-art review. *IEEE Access* **2016**, *4*, 6014–6058. [CrossRef]
4. Sun, Y.; Liu, Z.; Todorovic, S.; Li, J. Adaptive boosting for SAR automatic target recognition. *IEEE Trans. Aerosp. Electron. Syst.* **2007**, *43*, 112–125. [CrossRef]
5. Feng, X.; Haipeng, W.; Yaqiu, J. Deep learning as applied in SAR target recognition and terrain classification. *J. Radars* **2017**, *6*, 136–148.
6. Brenner, A.R.; Roessing, L. Radar imaging of urban areas by means of very high-resolution SAR and interferometric SAR. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 2971–2982. [CrossRef]

7. Essen, H.; Fuchs, H.H.; Pagels, A. High resolution millimeterwave SAR for the remote sensing of wave patterns. In Proceedings of the 2007 IEEE International Geoscience and Remote Sensing Symposium, Barcelona, Spain, 23–28 July 2007; pp. 963–966.
8. Li, J.; Wang, C.; Wang, S.; Zhang, H.; Zhang, B. Classification of very high resolution SAR image based on convolutional neural network. In Proceedings of the 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP), Shanghai, China, 18–21 May 2017; pp. 1–4.
9. Zhang, B.; Wang, C.; Wu, F.; Zhang, H. New detector based on patch segmentation for high resolution SAR image. In Proceedings of the 2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS, Melbourne, VIC, Australia, 21–26 July 2013; pp. 907–910.
10. Shao, J.; Qu, C.; Li, J. A performance analysis of convolutional neural network models in SAR target recognition. In Proceedings of the 2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA), Beijing, China, 13–14 November 2017; pp. 1–6.
11. Nie, T.; He, B.; Bi, G.; Zhang, Y.; Wang, W. A method of ship detection under complex background. *ISPRS Int. J. Geo Inf.* **2017**, *6*, 159. [CrossRef]
12. Zhong, C.; Mu, X.; He, X.; Wang, J.; Zhu, M. SAR target image classification based on transfer learning and model compression. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 412–416. [CrossRef]
13. Du, L.; Liu, B.; Wang, Y.; Liu, H.; Dai, H. Target detection method based on convolutional neural network for SAR image. *J. Electron. Inf. Technol.* **2016**, *38*, 3018–3025. [CrossRef]
14. Qi, W.; Wen, G. SAR target classification method based on convolutional neural network. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; pp. 1151–1155.
15. Ding, J.; Chen, B.; Liu, H.; Huang, M. Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 364–368. [CrossRef]
16. Zhou, Y.; Wang, H.; Xu, F.; Jin, Y.Q. Polarimetric SAR image classification using deep convolutional neural networks. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 1935–1939. [CrossRef]
17. Wang, J.; Zheng, T.; Lei, P.; Bai, X. Ground target classification in noisy SAR images using convolutional neural networks. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 4180–4192. [CrossRef]
18. Paladini, R.; Famil, L.F.; Pottier, E.; Martorella, M.; Berizzi, F.; Dalle Mese, E. Point target classification via fast lossless and sufficient Ω–Φ–$ invariant decomposition of high-resolution and fully polarimetric SAR/ISAR data. *Proc. IEEE* **2013**, *101*, 798–830. [CrossRef]
19. Gorovyi, I.M.; Sharapov, D.S. Efficient object classification and recognition in SAR imagery. In Proceedings of the 2017 18th International Radar Symposium (IRS), Prague, Czech Republic, 28–30 June 2017; pp. 1–7.
20. Du, K.; Deng, Y.; Wang, R.; Zhao, T.; Li, N. SAR ATR based on displacement- and rotation-insensitive CNN. *Remote Sens. Lett.* **2016**, *7*, 895–904. [CrossRef]
21. Chen, S.; Wang, H. SAR target recognition based on deep learning. In Proceedings of the 2014 International Conference on Data Science and Advanced Analytics (DSAA), Shanghai, China, 30 October–1 November 2014; pp. 541–547.
22. Geng, J.; Fan, J.; Wang, H.; Ma, X.; Li, B.; Chen, F. High-resolution SAR image classification via deep convolutional autoencoders. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2351–2355. [CrossRef]
23. Chen, S.; Wang, H.; Xu, F.; Jin, Y.Q. Target classification using the deep convolutional networks for SAR images. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4806–4817. [CrossRef]
24. Xue, Y.; Pei, J.; Huang, Y.; Yang, J.; Zhang, Y. Target recognition for SAR images based on heterogeneous CNN ensemble. In Proceedings of the 2018 IEEE Radar Conference (RadarConf18), Oklahoma City, OK, USA, 23–27 April 2018; pp. 0507–0512.
25. Kang, C.; He, C. SAR image classification based on the multi-layer network and transfer learning of mid-level representations. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 1146–1149.
26. Li, X.; Li, C.; Wang, P.; Men, Z.; Xu, H. SAR ATR based on dividing CNN into CAE and SNN. In Proceedings of the 2015 IEEE 5th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR), Singapore, 1–4 September 2015; pp. 676–679.
27. Zhang, Y.; Song, Y.; Wang, Y.; Qu, H. A fast training method for SAR large scale samples based on CNN for targets recognition. In Proceedings of the 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, 13–15 October 2018; pp. 1–5.

28. Zhao, P.; Liu, K.; Zou, H.; Zhen, X. Multi-Stream Convolutional Neural Network for SAR Automatic Target Recognition. *Remote Sens.* **2018**, *10*, 1473. [CrossRef]
29. Gao, F.; Huang, T.; Wang, J.; Sun, J.; Yang, E.; Hussain, A. Combining deep convolutional neural network and SVM to SAR image target recognition. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 21–23 June 2017; pp. 1082–1085.
30. Jiang, C.; Zhou, Y. Hierarchical fusion of convolutional neural networks and attributed scattering centers with application to robust SAR ATR. *Remote Sens.* **2018**, *10*, 819. [CrossRef]
31. Le, Q.V.; Zou, W.Y.; Yeung, S.Y.; Ng, A.Y. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In Proceedings of the CVPR 2011, Providence, RI, USA, 20–25 June 2011; pp. 3361–3368.
32. Zhou, Z.H.; Feng, J. Deep forest: Towards an alternative to deep neural networks. *arXiv* **2017**, arXiv:preprint/1702.08835.
33. Tian, S.; Wang, C.; Zhang, H. SAR object classification with a Multi-Scale convolutional auto-encoder. In Proceedings of the 2019 SAR in Big Data Era (BIGSARDATA), Beijing, China, 5–6 August 2019; pp. 1–4.
34. Deng, S.; Du, L.; Li, C.; Ding, J.; Liu, H. SAR automatic target recognition based on Euclidean distance restricted autoencoder. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 3323–3333. [CrossRef]
35. Henri, M. *Processing of Synthetic Aperture Radar Images*; Willey: Hoboken, NJ, USA, 2008.
36. Freund, Y.; Schapire, R.; Abe, N. A short introduction to boosting. *J. Jpn. Soc. Artif. Intell.* **1999**, *14*, 1612.
37. Cui, Z.; Cao, Z.; Yang, J.; Feng, J.; Ren, H. Target recognition in synthetic aperture radar images via non-negative matrix factorisation. *IET Radar Sonar Navig.* **2015**, *9*, 1376–1385. [CrossRef]
38. Dong, G.; Kuang, G.; Wang, N.; Zhao, L.; Lu, J. SAR target recognition via joint sparse representation of monogenic signal. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 3316–3328. [CrossRef]
39. Dong, G.; Kuang, G. Target recognition in SAR images via classification on Riemannian manifolds. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 199–203. [CrossRef]