Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 16(5): 4069–4081.
DOI: 10.3934/mbe.2019201
Received: 21 January 2019
Accepted: 29 April 2019
Published: 08 May 2019

*Research article*

# A new JPEG image steganalysis technique combining rich model features and convolutional neural networks

**Tao Zhang[1],*, Hao Zhang[2], Ran Wang[2] and Yunda Wu[2]**

[1] School of Computer Science and Engineering, Changshu Institute of Technology, No.99, Hushan Road, Changshu 215500, Jiangsu Province, China
[2] National Digital Switching System Engineering and Technology Research Center, No.62, Kexue Avenue, Zhengzhou 450001, Henan Province, China

* **Correspondence:** Email: Brunda@163.com; Tel: +86-512-5225-1702; Fax: +86-512-5225-1702.

**Abstract:** The best traditional steganalysis methods aiming at adaptive steganography are the combination of rich models and ensemble classifier. In this study, a new steganalysis method for JPEG images based on convolutional neural networks is proposed to solve the high dimension problem in steganalysis from another aspect. On the basis of the original rich model, the algorithm adds different sizes of discrete cosine transform (DCT) basis functions to extract different detection features. Different features are combined at the fully connected layer through inputting 2-D feature values to the neural network convolutional layer for predictive classification. Experimental results show that convolutional neural networks as classifiers do not require a large number of training samples, and the final classification performance is better than that of the original ensemble classifier.

**Keywords:** Steganalysis; rich model; ensemble classifier; convolutional neural networks

## 1. Introduction

Image steganography is a technique in which a secret message is embedded into an image and transmitted through a public channel without gaining the attention of a third party, thereby implementing covert communication. The adaptive steganography algorithm changes the assumption that steganographic information is independent of the carrier content and can adapt to the image content when embedding the secret message. The detection difficulty is greatly improved, which is the hotspot and challenge in the current research. Steganalysis is the reverse process of steganography; its purpose is to determine whether the image to be tested contains a confidential message. As a typical binary classification problem, most steganalysis methods include feature extraction and classifier design. Among the general detection algorithms for adaptive steganography, the use of rich model features [1–3] and ensemble classifier has the most prominent effect [4].

The concept of rich model was first proposed by Fridrich et al. in [5]. A rich steganographic detection feature is proposed from each submodel by building rich submodels for images, and then all the features are combined to form rich model features, which are usually of high dimensions. For example, the dimensions of steganographic features SRM [1] and PSRM [6] for spatial images are 12,753 and 12,870, respectively; and the dimensions of detection features DCTR [1] and PHARM [3] for JPEG images are 8,000 and 12,600, respectively. Rich model features can extract steganographic features from multiple angles, which can capture the statistical characteristic changes caused by message embedding. However, the computational complexity of rich model features is usually large due to their high dimensionality.

The classifier used in steganographic detection is limited compared with that in rich detection features. Support vector machine is the most used classifier for traditional steganographic detection features owing to its solid statistical model and mathematical proof as a support, which can avoid overfitting problems. Although the feature dimension is higher than the number of training samples, better experimental results can still be obtained. However, when the training number and feature dimension are high, the training efficiency will be significantly reduced due to the need to calculate the Gram matrix. Therefore, the detector can only select features with low dimensionality when designing the detection features and the corresponding detection results are limited.

Kodovsky et al. proposed an ensemble Fisher's linear discriminant classifier based on random forest to solve the problem of high dimensionality of rich model features [7]. The ensemble classifier consists of a number of independent sub-classifiers that are trained by randomly selected subspaces in the feature space. For the same training sample, multiple sub-classifiers with different performances can be obtained by selecting the feature subspace multiple times. The classification results of these sub-classifiers are ensembled by voting to obtain the final decision result. Selecting the subspace in the feature space eliminates the constraints of high-dimensional features and the number of training samples, thereby leading to a high training rate. Steganographic detection on the spatial domain and JPEG image shows that the proposed method is more suitable for adaptive steganography detection than the traditional method, and subsequent steganographic detection methods are mostly performed through the combination of rich model and ensemble classifier.

In recent years, deep learning has been widely used as a powerful tool for solving complex problems in many aspects, such as automatic driving, face recognition, and image understanding. The deep-learning-based computer program AlphaGo has even beat top professional Go players for the first time in history. With the rapid development of deep learning, this technology has gradually been applied to the field of steganography. Tan and Li [8] first attempted to use steganographic detection of spatial images through stacked autoencoders in deep learning. The test image set is images embedded using HUGO [9] algorithms under BOSSbase, and the embedding rate is 0.4 bit per pixel (bpp). On the premise of pretraining the parameters of a neural network, the detection accuracy is still 17% lower than that of the SRM feature and the ensemble classifier combination. Although the performance of steganographic detection is much lower than that under the traditional method, it provides a new idea for solving the steganalysis problem by neural network. Subsequently, Qian et al. proposed a steganographic detection model based on convolutional neural networks in [10].

The image to be tested was input as a neural network after being filtered through a 5 × 5 high-pass filter. After five layers of convolutional layer learning, a 256-dimensional detection feature was generated which was classified by a three-layer fully connected layer. The experimental results under the three embedded algorithms of HUGO, WOW [11], and S-UNIWARD [12] show that the detection rate under this structure is only 3%–4% lower than that of SRM, close to the best result of the traditional algorithm. The algorithm proposed by Pibre et al. [13] also needs to filter images, but the basic idea is to use few convolution layers and large convolution kernel sizes. The detection performance of the S-UNIWARD embedding algorithm with an embedding rate of 0.4 bpp is 16% higher than that of the rich model, which can reduce the impact of image source mismatch to a certain extent. Chen et al. [14] proposed CNN steganalysis network supporting phase-aware channel; Zeng et al. [15] proposed hybrid deep learning network for JPEG steganalysis; Xu et al. [16] used 20-layer CNN for JPEG steganalysis, and verified by experiments that it has better performance in J-UNIWARD detection. Boroumand et al. [17] recently proposed a method based on deep residual networks, claiming that it can achieve optimal performance in both spatial domain and JPEG steganography.

Unlike the traditional means of extracting features on images, these methods perform steganographic detection directly on the original image or filtered image based on deep learning, thus avoiding complex feature extraction and classifier design process. However, a great difficulty exists in performing steganographic detection directly on images. Some open-source algorithms [13] show that the use of deep learning for steganographic analysis can achieve good experimental results because fixed secret messages and embedded locations are used when embedding secret messages. These locations can be easily learned and can avoid the influence of carrier source mismatch. Currently, detection methods still cannot avoid including the rich model method when extracting features, and performing deep learning detection directly on the original image remains difficult.

In this study, rich model features with tens of thousands of dimensions are dealt with through the use of deep learning methods. Extracted features are regarded as learning objects to replace traditional ensemble classifiers via the deep learning of rich model features. The experimental results show that the proposed method exhibits better classification performance than the ensemble classifier because it does not require a large number of auxiliary samples, thus avoiding numerous image embedding operations. The GPU platform acceleration operation can avoid long training time.

## 2. Basic structure of convolutional neural networks

A convolutional neural network [18] is an artificial neural network that has become a research hotspot in the field of image processing. Its weight-sharing network structure is similar to that of a biological neural network, in which the complexity of the network model and the number of weights are reduced. Therefore, an image can be directly used as the network input. This process avoids the complicated feature extraction and classifier design process in the traditional classification algorithm. A typical neural network is usually composed of alternating convolutional layers and pooling layers. The last few layers of the network near the output layer are usually fully connected layers, as shown in Figure 1. The training process of its parameters adjusts the weights and offsets by minimizing the residuals through a back-propagation method.

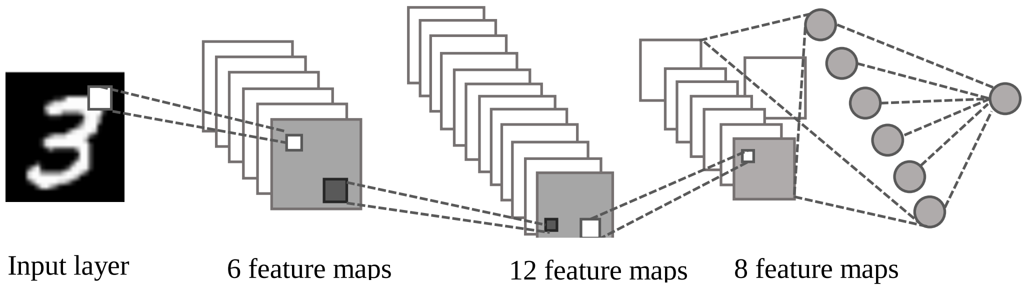

**Figure 1.** Example of a convolutional neural network.

## *2.1. Convolution layer*

The feature map of the *k*th channel of the convolutional layer *l* is $F_k^l$. This feature map can be convolved by the upper-layer feature map and the convolution kernel and then obtained by an activation function, as shown below:

$$X_k^l = \sum_{n=1}^{K^{l-1}} (W_k^l * F_n^{l-1}) \qquad (1)$$
$$F_k^l = f(X_k^l + b_k^l)$$

where $F_n^{l-1}$ is the *n*th feature map of the previous layer, $b_k^l$ is the offset of the feature map after convolution, and the symbol "*" is the convolution symbol. *f* is the activation function, and the commonly used activation functions are Sigmoid, Tanh, and Relu function. $K^{l-1}$ is the number of convolution kernels in the previous layer. When $l = 1$, we have $K^{l-1} = K^0 = 1$ and $F_n^{l-1} = F_1^0 = I$, that is, the input of convolutional layer is the input image *I*. $W_k^l$ is a convolution kernel weight matrix that corresponds to the *k*th feature map. The weight matrix $W_k^l$ determines the size of the convolution region, and its value affects the output feature map size. The size of the convolutional output also depends on two parameters, namely, stride and padding. Stride controls the distance between two adjacent hidden units at the same depth and their connected input areas. The total size of the input unit can be changed by zero padding around the input unit, thereby controlling the size of the output unit. Normally we have equal stride length and padding in both the X and Y dimension. Therefore, in case of a given stride *S* and a zero-padded number *P*, for square filters, the size of the output image after learning for an input image of the same length and width through the convolutional layer is

$$\dim(F_k^l) = \frac{(\dim(F^{l-1}) - \dim(W_k^l) + 2 \times P)}{S} + 1 \qquad (2)$$

## *2.2. Pooling layer*

The pooling layer is also often referred to as the down-sampling layer, and the *l*th feature map of the *k*th channel of pooling layer $F_k^l$ can be calculated by the following formula:

$$\begin{aligned} X_k^l &= \gamma_k^l \,\text{pooling}(F_k^{l-1}) \\ F_k^l &= f(X_k^l + b_k^l) \end{aligned} \tag{3}$$

where $\gamma_k^l$ is the weighting factor of the pooling layer, and $b_k^l$ is the offset term after down-sampling. Pooling is a down-sampling function. The basic principle is that a 2-D feature map is divided into a plurality of subregions with a size of $r \times r$, and the maximum or average value of the region is taken as a feature of the region and transmitted to the next layer. If the subareas do not overlap during the pooling process, then the dimensions of the output image will be 1 / r times the dimensions of the input image. The pooling method can reduce the data while reducing the variance among data. However, the pooling layer will rapidly reduce the amount of data. Small pooling filters are currently adopted, and the pooling layer is never used. The pooling layer is removed due to this factor for the convolutional neural network structure used in this study.

### *2.3. Fully connected layer*

In the fully connected layer, the feature map of a 2-D image is first flattened into 1-D features as the input of the fully connected network. All local features are combined into global features, and the output probability of each class is calculated. The formula is as follows:

$$\begin{aligned} X^l &= W^l F^{l-1} \\ F^l &= f(X^l + b^l) \end{aligned} \tag{4}$$

The feature map of the fully connected layer $l$ is $F^l$, and $W^l$ and $b^l$ are the weight coefficients and offset terms of the corresponding fully connected network, respectively. The output of fully connected layer is obtained by input weighting and by the activation function, and the output layer neurons are fully connected to the input layer neurons.

## 3. DCTR features

Holub et al. proposed the DCTR features in [1]. For a JPEG image, 64 discrete cosine transform (DCT) kernel functions are used to filter the noise residuals after decompressing, and the noise residuals are quantized and truncated. The DCTR feature can be finally obtained by quantizing the first-order statistics of the noise. Compared with other rich model features, the DCTR feature has higher detection performance but lower computational complexity and dimension. The detailed algorithm is as follows.

For a JPEG grayscale image of size $M \times N$, we first decompress it to the spatial domain to obtain pixel data $I$, and 64 DCT transform kernels $\mathbf{B}^{(k,l)}$ of size $8 \times 8$ are defined as shown in Equation (5), where $0 \le k, l \le 7$, $0 \le m, n \le 7$.

$$\begin{aligned} \mathbf{B}^{(k,l)} &= (B_{mn}^{(k,l)}) \\ B_{mn}^{(k,l)} &= \frac{w_k w_l}{4} \cos\left(\frac{\pi l(2m+1)}{16}\right) \cos\left(\frac{\pi l(2n+1)}{16}\right) \end{aligned} \tag{5}$$

where $w_0$ is $\frac{1}{\sqrt{2}}$; for $k > 0$, $w_k = 1$, noise residual $U^{(k,l)}$ can be obtained by convolving $B^{(k,l)}$ with the input image $I$, that is,

$$U^{(k,l)} = I * B^{(k,l)} \tag{6}$$

All elements in $U^{(k,l)}$ are first quantized and truncated by quantization step $q$ and truncation threshold $T$. For JPEG images with the same quality factor, the choices of $q$ and $T$ are fixed values.

$$R^{(k,l)} = \text{abs}(\text{round}(\tfrac{U^{(k,l)}}{q})) \tag{7}$$

$$R^{(k,l)}(R^{(k,l)} > T) = T \tag{8}$$

Finally, 125-dimensional features can be obtained by calculating the first-order statistics of some elements in each residual noise $R^{(k,l)}$, and the results of 64 residual noises can be combined to obtain the final 8,000-dimensional features.

In [1], the size of the filtered convolution kernel is fixed to 8 without considering the influences of different convolution kernel sizes on the detection performance or the influence of the feature combination under different convolution kernels on the final classification results. In a convolutional neural network, different features of the same image can be easily combined to enhance the classification effect. For example, in the processing of a color image, three color components are usually taken as three inputs, and the features are combined and classified at the fully connected layer. The $8\times8$ convolution kernel remains large in image processing under normal circumstances; other smaller convolution kernels are selected to obtain more steganographic detection features.

If the convolution kernel size is $N$, then for the DCT base $B^{(k,l)}$ in Equation (5), the constraint condition of $k$, $l$, $m$ and $n$ is $0 \le k, l, m, n \le N$-1. Each transform base can extract 125-dimensional features; thus, each type of convolution kernel can extract $125\times N^2$ dimensional features. The features under different convolution kernels or the decision results under different features can be combined. The method adopted in this study is to learn the different features of the same image through the convolutional layer, combine them in the fully connected layer, and classify the decision. In Section 4.2, the influence of convolution kernel size on the final steganographic detection results is discussed by combining the detection features under different convolution kernels. We find that the classification performance is the best under the combination of sizes 4 and 8 while the feature dimensions are 2,000 and 8,000, respectively. They are referred to as DCTR4 and DCTR8.

## 4. Network structure in this paper

### *4.1. Structure of the convolutional neural network*

The original extracted DCTR feature is a 1-D vector. If the feature vector is directly used as the input of the fully connected layer and is only trained by the self-encoder in deep learning, the final experimental result is close to a random guess. Therefore, we adopted a more complex convolutional neural network. Figure 2 shows the complete network structure, which mainly includes three parts, namely, feature extraction, convolutional layer, and fully connected layer.

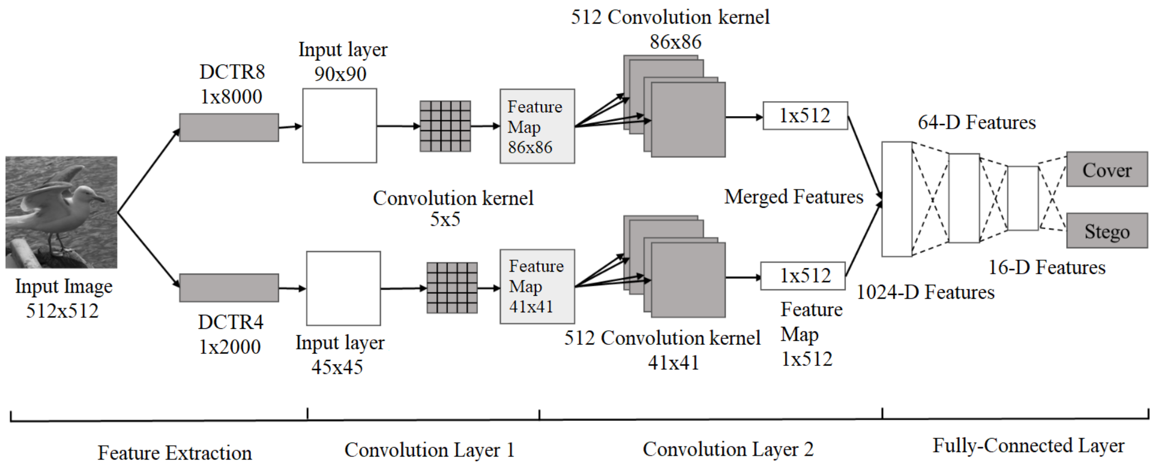

**Figure 2.** Neural network structure in this paper.

For the image to be tested, when the convolution kernel size is $N$, the extracted feature dimension is 125 $\times N^2$ and the size of the transformed 2-D matrix is $\lceil N*\sqrt{125} \rceil$. The extra data are zero-padded and used as input to the convolutional neural network. Therefore, the feature vectors of DCTR8 and DCTR4 can be converted into 2-D matrices of size 90 $\times$ 90 and 45 $\times$ 45, respectively. The additional 100 and 25 data are zero-padded.

The convolutional layer part mainly consists of two layers. The first layer is a single 5 $\times$ 5 convolution kernel whose function is to preprocess the input features. The sizes of padding and stride are 0 and 1, respectively. From Equation (2), the dimensions of the DCTR8 and DCTR4 output feature maps are 86 and 41, respectively. For the second layer of the convolutional layer, the size of the convolution kernel is selected to use the possibly largest convolution kernel and treat the feature map as a whole to extract the overall features rather than the local features. Therefore, the convolution kernel sizes of the second layer are set to be 86 $\times$ 86 and 41 $\times$ 41, which are the same sizes as the feature map to be processed with the size of the generated feature map as 1. When the number of convolution kernels is 512, two 512-dimensional features can eventually be learned for each DCTR8 feature and each DCTR4 feature.

The learning results of the convolutional layer are combined as the input of the fully connected layer, and the number of neurons in the three-layer fully connected layer is 1024, 64, and 16. The prediction probability of whether the image to be tested is the carrier or the stego-image is calculated by the softmax function.

In the neural network structure mentioned in this paper, the activation functions in the convolutional layer and the fully connected layer are the tanh and relu functions, respectively. The pooling layer typically used is removed because almost all pooling layers will lead to information loss to a certain extent. The network size is within the scope of ordinary computer processing, and hence, the training speed can be improved without affecting the accuracy.

## 4.2. *Parameter settings of the convolutional neural network*

The selection of network parameters and optimization algorithm will greatly affect the final classification performance during the use of neural networks. In the network structure proposed in this study, the initial value of the convolution kernel is randomly generated by a Gaussian distribution with a zero mean variance of 0.01, and the fully connected layer parameters are initialized by the method [19] proposed by Xavier et al. The optimization algorithm adopts the most commonly used gradient descent method in current neural networks, and many algorithms can be achieved based on this method. The gradient descent method adopted in this study is SGD (i.e., mini-batch gradient descent). The basic idea is to randomly extract a batch of samples from the training samples and update the neural network parameters according to these training samples. The traditional method requires calculation of all training samples before updating the parameters. Therefore, a large number of redundant calculations occur when the amount of data is large. SGD can avoid this redundancy by calculating the parameters of some of the sample updates, and the training speed is fast and stable. In this study, the SGD training process parameters are set as follows: number of samples per training (mini-batch) is 200, in which the number of carriers and stego samples is 100; the learning rate (learning rate) is 0.01; the decay rate of the learning rate decay rate is $5 \times 10^{-7}$; and the momentum value is 0.9. A large momentum value can speed up the convergence.

Techniques in training neural networks can be used to prevent training samples from overfitting and improve the performance of classification. The methods adopted are batch normalization [20] and dropout [21]. Batch normalization normalizes the calculation output of the corresponding activation function by mini-batch during each SGD calculation such that the average value of the output is 0 and the variance is 1, thereby speeding up the training and improving the accuracy. Experiments indicate that batch normalization can help speed up model convergence. The main idea of dropout is to randomly select the weights of some hidden layer nodes of the network to not work while only maintaining the weight value when training the model, thereby avoiding certain features from taking effect only under a fixed combination. Dropout is not adopted because the core of the proposed algorithm is the learning of the entire feature, and some nodes do not need to be removed to strengthen the network model.

## 5. Experimental results and performance analysis

### *5.1. Image data set and software platform*

In this experiment, the general experimental library Bossbase v1.01 is used in steganalysis to facilitate a comprehensive and fair evaluation of the proposed algorithm performance. The image gallery contains 10,000 spatial grayscale images of size $512 \times 512$, which are compressed and stored as JPEG images with a quality factor of 75. Given that rich model features are often used to detect adaptive steganography algorithms with strong undetectability, two new adaptive steganographic algorithms are adopted in this study, J-UNIWARD [12] and UED [22]. All convolutional neural networks are trained and tested on a Python-based deep learning framework named Theano [23], which speeds up operations by parallel processing on the GPU platform.

### *5.2. Filter selection for the convolution layer*

The detection performance of feature combinations under different DCT bases is discussed to study the influences of different DCT bases on the steganographic detection performance in the feature extraction process of rich models. The steganography algorithms are J-UNIWARD and UED, and the embedding rate is fixed to 0.4 bit per nonzero AC coefficient (bpnc). For the neural network in this paper, the numbers of training and test samples are 6,000 and 2,000, respectively, and the number of iterations is 800. The detection accuracy rate (acc) is used as the evaluation index, and the feature combination has one, two, and three feature combinations. The experimental results are shown in Table I. The combination mode shown in the first and fourth column means the combination of one, two or three types of features using DCT basis functions with different sizes. For example , "(4) " means that we extract features using DCT basis functions with size of 4, "(4,5)" means that we extract features using DCT basis functions with sizes of 4 and 5, and (4,5,6) means that we extract features using DCT basis functions with sizes of 4, 5 and 6.

In most cases, the performance of a combination of two features is better than that of a single feature. The combination of three types of features performs better than that of two types of features. Increasing DCT basis functions with different sizes can help improve the detection rate. However, the detection performance advantages of combining three types of features are no longer the best due to the considerable time consumption of the feature extraction and classification process of the rich model. The DCT base sizes of the final selected two types of features are 4 and 8.

**Table 1.** Correct detection rates (ACC) of feature combination under different DCT bases.

| combination | embedding algorithm | | combination | embedding algorithm | |
|---|---|---|---|---|---|
| mode | J-UNIWARD | UED | mode | J-UNIWARD | UED |
| (4) | 0.804 | 0.936 | (5) | 0.837 | 0.934 |
| (6) | 0.832 | 0.947 | (7) | 0.846 | 0.948 |
| (8) | 0.851 | 0.952 | | | |
| (4,5) | 0.815 | 0.924 | (4,6) | 0.822 | 0.942 |
| (4,7) | 0.837 | 0.938 | (4,8) | **0.866** | **0.969** |
| (5,6) | 0.845 | 0.956 | (5,7) | 0.839 | 0.948 |
| (5,8) | 0.854 | 0.961 | (6,7) | 0.849 | 0.957 |
| (6,8) | 0.861 | 0.966 | (7,8) | 0.850 | 0.964 |
| (4,5,6) | 0.843 | 0.978 | (4,5,7) | 0.848 | 0.957 |
| (4,5,8) | **0.868** | 0.970 | (5,6,8) | 0.864 | 0.966 |
| (5,7,8) | 0.860 | 0.964 | (6,7,8) | 0.863 | **0.971** |

## *5.3. Iteration time selection for neural network*

In the training process of a neural network, the selection of the iteration times poses a significant impact on the classification results and efficiency. Early stop [23] is an optimization method that effectively controls the number of iterations. The training data are In the training process of a neural network, the selection of the iteration times poses a significant impact on the classification results and efficiency. Early stop [23] is an optimization method that effectively controls the number of iterations. The training data are divided into a training set and a verification set, which are trained on the training set and tested on the verification set. As the number of iterations increases, the training is stopped if the test errors increase on the verification set, and the weight at this time is taken as the final parameter, which prevents overfitting. However, in the actual application process, the training process is not always stopped once the detection rate is lowered because the detection rate of the verification set may continue to increase as the training process increases.

The method used in this study is to select 6,000 images from the carrier and stego images as the training images. A total of 2,000 verification images and 2,000 test images are selected in the experimental process, and the training frequency is 1,000. The correct rates of each training sample and each verification sample are recorded. Two embedding algorithms with embedding rates of 0.1 and 0.4 bpnc are selected. The detection rate of training and verification samples varies with the number of iterations, as shown in Figures 3 and 4. In Figure 3 and Figure 4, "train_0.1" and "val_0.1" stands for the detection rates of training and verification samples with embedding rates of 0.1 bpnc, while "train_0.4" and "val_0.4" is the detection rates of training and verification samples with embedding rates of 0.4 bpnc.

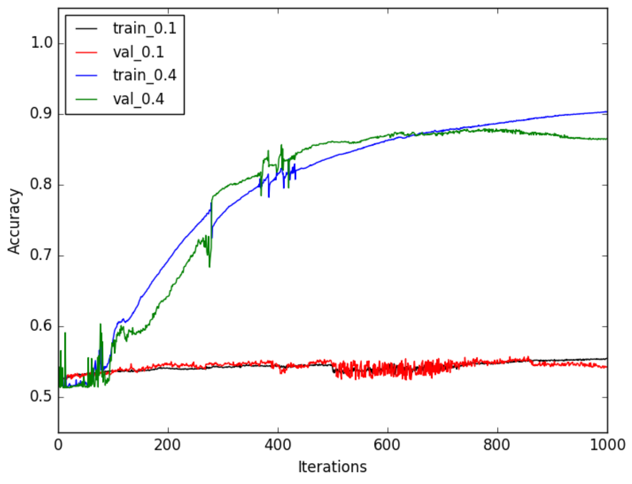

**Figure 3.** Recognition rate on J-UNIWARD training and verification sets.

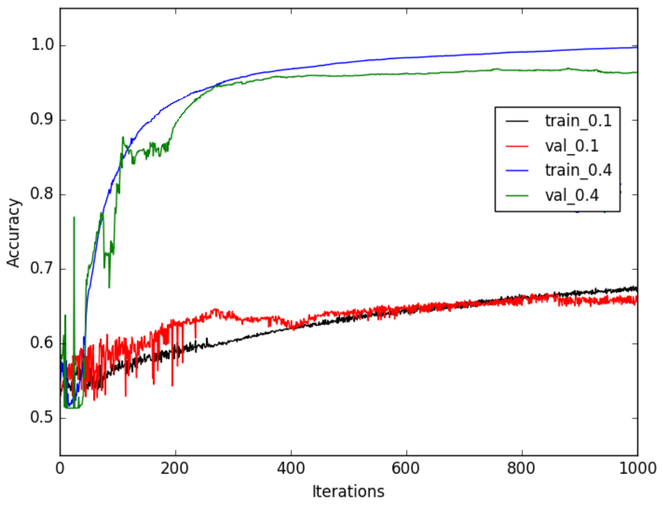

**Figure 4.** Recognition rates on UED training and verification sets.

As the iteration times increase, although the correct rate of the training set is still increasing, the correct rate of the verification set begins to decrease after reaching a certain level. When the number of training reaches 800 or so, the detection performance of the verification set is stable and optimal.

If the training continues, then overfitting will occur. The weight of the iteration times of 800 is selected as the final network parameter.

## *5.4. Comparison with ensemble classifier performance*

The performance of the proposed algorithm is verified by the following experiment. The model obtained in Section 4.3 is used as the test model, and 2,000 test images are used as test samples. This method is called CNN-DCTR. Traditional methods based on rich models and ensemble classifiers are considered as EC-DCTR, in which the numbers of training and test samples are also 6,000 and 2,000, respectively. The area under the receiver operating characteristic curve (AUC) and the detection accuracy rate are used as evaluation indexes, and the experimental results are shown in Figure 5. Experiments show that under various embedding rates, the detection rate of the proposed algorithm under different embedding algorithms and embedding rates is 2%–3% higher than that of traditional algorithms, which indicates that neural networks can be used for steganographic analysis based on rich models. The neural network-based training model often requires tens of thousands of images, whereas the proposed algorithm can converge through 6,000 images to obtain a stable classification effect.

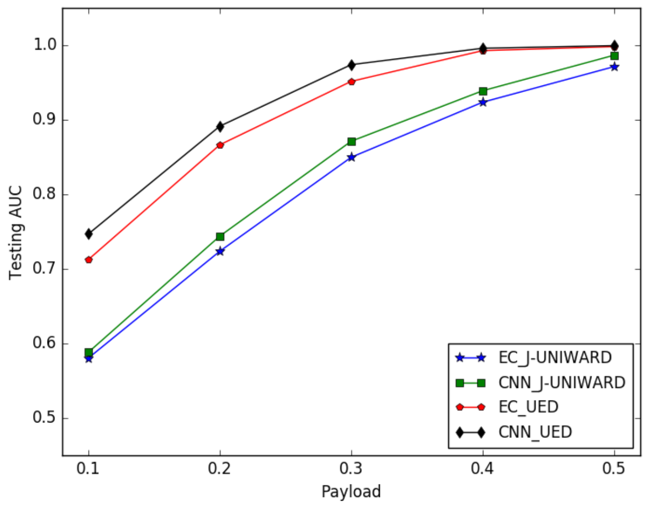

**Figure 5.** Performance comparison between the proposed algorithm and the ensemble classifier (AUC).

Although the proposed algorithm performs better than the traditional method, it is more computationally complex. The complexity of the neural-network-based classifier model depends on many factors, such as the complexity of the neural network structure, the number of training samples, and the number of training iterations. The most critical parameters are the weight and bias that must be optimized through continuous learning. In the network proposed in this study, the 512 86×86 and 41 × 41 convolution kernel parameter in convolution layer 2 is 512 × (86 × 86 + 41 × 41+ 2) = 4,648,448, which takes considerable time in the training process. Correspondingly, the numbers of parameters of convolutional layer 1 and the fully connected layer are 52 and 63,599 respectively. The computer used in the experiment is Dell XPS 8910 with Intel Core i76700 and NVIDIA graphics card GTX 1070. The iteration time for 6000 training images is about 11s. Approximately 2.5h are needed to complete the entire training process, and the ensemble classifier takes only 5 min to complete the entire training.

## 6. Conclusions

The steganographic algorithm proposed in this study utilizes the powerful parallel computing power of GPU and uses a convolutional neural network as a classifier instead of the traditional ensemble classifier. The problem of high feature dimension of the rich model is solved. A better detection performance than those of traditional methods can be obtained by combining different rich model features, and tens of thousands of training images can be used in the training process to converge and obtain better detection results. However, this method has certain problems and shortcomings while achieving excellent performance, mainly reflecting two aspects. First, the use of rich model features can improve the detection rate but also reduces the real-time performance. Second, the training time is longer than that for the traditional method, and the training requires GPU hardware support. For different rich model features, different scale inputs are required. In the future, we will discuss the application of different deep neural network structures in steganalysis of spatial and JPEG steganography.

**Acknowledgments**

This work was supported by the National Natural Science Foundation of China (No.61572518, U1636202，61602511, 61802430). The authors would like to thank the reviewers for their insightful comments and helpful suggestions.

## Conflict of interest

The authors declare no conflicts of interest.

## References

1. V. Holub and J. Fridrich, Low-complexity features for jpeg steganalysis using undecimated DCT, *IEEE Trans. Inf. Fore. Secu.*, **10**(2015), 219–228.
2. J. Fridrich and J. Kodovsky, Rich models for steganalysis of digital images, *IEEE Trans. Inf. Fore. Secu.*, **7**(2012), 868–882.
3. V. Holub and J. Fridrich, Phase-aware projection model for steganalysis of JPEG images, *Proc. SPIE*, **9409**(2015), 1–11.
4. J. Kodovsky, J. Fridrich and V. Holub, Ensemble classifiers for steganalysis of digital media, *IEEE Trans. Inf. Fore. Secu.*, **7**(2012), 432–444.
5. J. Fridrich, Steganalysis in high dimensions: fusing classifiers built on random subspaces, *Proc. SPIE*, **7880**(2011), 181–197.
6. V. Holub and J. Fridrich, Random projections of residuals for digital image steganalysis, *IEEE Trans. Inf. Fore. Secu.*, **8**(2013), 1996–2006.
7. J. Kodovskýand J. Fridrich, Steganalysis in high dimensions: fusing classifiers built on random subspaces, *Proc. SPIE*, **7880**(2011), 1–13.

8. S. Tan and B. Li, Stacked convolutional auto-encoders for steganalysis of digital images, *Proc. IEEE APSIPA*, Siem Reap, Cambodia, (2014), 1–4.
9. T. Pevný, T. Filler and P. Bas, Using high-dimensional image models to perform highly undetectable steganography, *Proc. IHW*, (2010), 161–177.
10. Y. Qian, J. Dong and W. Wang, Deep learning for steganalysis via convolutional neural networks, *Proc. SPIE*, **9409**(2015), 1–10.
11. V. Holub and J. Fridrich, Designing steganographic distortion using directional filters, *Proc. IEEE IFS*, (2012), 234–239.
12. V. Holub, J. Fridrich and T. Denemark, Universal distortion function for steganography in an arbitrary domain, *EURASIP J. Info. Sec.*, **2014**(2014), 1–13.
13. L. Pibre L, J. Pasquet and D. Ienco, Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch, *Proc. EI'2016*, (2016), 79–95.
14. Mo Chen, V. Sedighi, M. Boroumand, et al., JPEG-phase-aware convolutional neural network for steganalysis of JPEG images, *Proc. IH MMSec*, (2017), 1–10.
15. J. Zeng, S. Tan, B. Li, et al., Large-scale JPEG image steganalysis using hybrid deep-learning framework, *IEEE Trans. Info. Forens. Secu.*, **13**(2018), 1200–1214.
16. G. Xu, Deep convolutional neural network to detect JUNIWARD, *Proc. IH MMSec*, (2017), 1–6.
17. M. Boroumand, M. Chen and J. Fridrich, Deep residual network for steganalysis of digital images. *IEEE Trans. Info. Fore. Secu.*, **14**(2019), 1181–1193.
18. L. Chang, X. Den and M. Zhou, Convolution neural networks in image understanding, *Acta Automat. Sinica*, **42**(2016), 1300–1312.
19. X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, *J. Mach. Lear. Rese.*, **9**(2010), 249–256.
20. S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *Proc. ICML*, (2015), 448–456.
21. N. Srivastava, G. Hinton and A. Krizhevsky, Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Lear. Rese.*, **15**(2014), 1929–1958.
22. L. Guo, J. Ni and Y. Shi, Uniform embedding for efficient JPEG steganography, *IEEE Trans. Info. Fore. Secu.*, **9**(2014), 814–825.
23. J. Bergstra, O. Breuleux and F. Bastien, Theano: A CPU and GPU math compiler in Python, *Proc. PSC*, (2010), 1–7.