Computing and Informatics, Vol. 43, 2024, 295–316, doi: 10.31577/cai_2024_2_295

# SASRNET: SLIMMING-ASSISTED DEEP RESIDUAL NETWORK FOR IMAGE STEGANALYSIS

Siyuan Huang, Minqing Zhang, Yan Ke, Fuqiang Di, Yongjun Kong

*College of Cryptographic Engineering*
*Engineering University of PAP*
*710086 Xi'an, China*
*e-mail:* huangsiyuan828@163.com

**Abstract.** Existing deep-learning-based image steganalysis networks have problems such as large model sizes, significant runtime memory usage, and extensive computational operations, which hinder their deployment in many practical applications. To address these challenges, we applied model compression techniques to image steganalysis and designed a model called SASRNet, a slimming-assisted steganalysis residual network. We observed that the trainable scale factor of BN (batch normalization) layer in steganalysis network can be used as channel scaling factor for pruning. The channel-level sparsity of convolutional layers is enhanced by imposing L1 regularization on channel scaling factors and pruning less informative feature channels. With the goal of balancing performance and efficiency, the iterative algorithm is used to further compress the network to obtain a slimming-steganalysis detector. In contrast to many existing methods, our proposed method can be directly applied to steganalysis network architectures by introducing a minimal overhead to the training process. We have conducted extensive experiments on BOSSBase + BOWS2 dataset. Experiments show that, compared to the original steganalysis model, this method can achieve comparable performance with less than 5 % of the parameters, validating the feasibility and practicality of the new model.

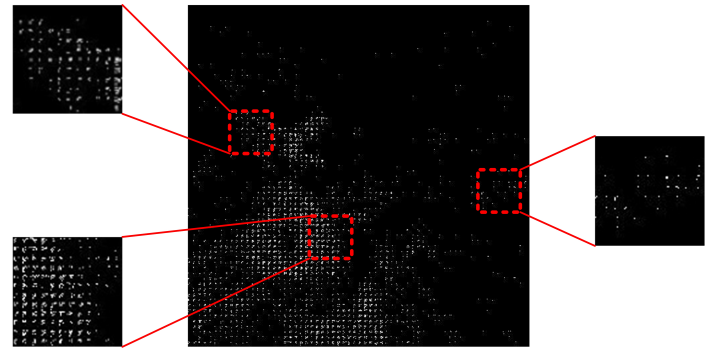
**Keywords:** Deep learning, image steganalysis, network slimming, model compression, channel pruning

## 1 INTRODUCTION

Image steganalysis and image steganography are technologies that act against each other and promote development of each other in gaming [1, 2, 3]. Steganalysis, the opposite of steganography, has been proposed to detect whether an image is embedded with secret information. The difficulty of image steganalysis is that the steganalysis model cannot fully and effectively model the slight differences that occur when the steganography operation embeds secret messages in an image. Figure 1 shows an example of image steganography, Figure 1 b) shows the cover image, and Figure 1 c) shows the stego image, and Figure 1 a) shows the pixel-by-pixel difference between the cover and stego images. It can be observed that Figures 1 b) and 1 c) are difficult to recognize with the human eye. In addition to the detection accuracy, running and training rate, the number of model parameters, and calculation cost, are all important factors to be considered in the real application of image steganalysis. At the same time, with the popularity of intelligent devices, image steganalysis is applied more and more widely. Therefore, the deployment of image steganalysis model has become a key problem worth studying.

Deep neural network (DNN) have recently become the primary method for computer vision tasks. They exhibit excellent performance in object detection [4], image classification [5], semantic segmentation [6], and other tasks. Image steganalysis based on DNN have also been proposed to detect whether secret information is embedded in an image. However, as the image steganalysis task complexity increases, the number of steganalysis model parameters increase. Further, the number of layers becomes deeper, resulting in a longer running time and more space overhead, which limits the use of steganalysis model in terminals with limited hardware conditions, such as applications on mobile phones and portable smart devices. For example, the basic DNN model ResNet [7], which is widely used in image steganalysis, the 152-layer ResNet has more than 60 million parameters, a significant resource burden for embedded devices. In recent years, the continuous development of model compression technology has solved some difficult problems in DNN deployment [8, 9, 10]. After model compression, the number of network parameters and amount of calculation are significantly reduced, resulting in low computational and storage costs. This significant reduction allows for performing complex imaging tasks with limited hardware conditions. Simultaneously, model compression can also efficiently compress and execute DNN without compromising accuracy. The current model compression methods can be divided into six categories: network pruning, parameter sharing, quantization, network decomposition, network distilling, and compact network design [11]. Pruning has become an important research direction for compressing and accelerating DNN because of its simple and effective characteristics in directly reducing redundancy in DNN models.

In this paper, we proposed SASRNet, using model compression for image steganalysis. The method introduces a pruning algorithm into the existing steganalysis model based on deep learning, thereby reducing the model parameters without significantly reducing the detection accuracy of steganalysis. Simultaneously, it reduces



a)

b)

c)

Figure 1. Cover image from BOSSbase1.01 and the corresponding stego image: a) the pixel-by-pixel differences between the cover image and stego image, b) cover image, and c) stego image; difficult to recognize with the human eye

the difficulty of model deployment and enables diversified practical applications of steganalysis research. Our method use a channel-pruning method to compress the trained model. The unimportant channels were automatically identified and then pruned to produce a compact model with considerable accuracy. Pruning unimportant channels may temporarily impair performance, but fine-tuning the pruned network can compensate for this effect. Currently, the application of model compression techniques to the field of image steganalysis is a challenging task. We have considered the applicability as well as the efficiency of various algorithms for steganalysis models in our design as well as in our experiments. We also optimize and improve the model compression scheme for the characteristics of image steganalysis. Multiple experimental results showed that our method could accelerate the model runtime and achieve the same or even higher accuracy while compressing the model.

The remainder of this paper is organized as follows. In Section 2, we review state-of-the-art research related to our proposed method. Section 3 elaborates on the proposed method in detail. The experimental results and analysis are presented in Section 4. Finally, Section 5 concludes the paper.

## 2 RELATED WORKS

### 2.1 Image Steganalysis Based on Deep Learning

Image steganalysis has experienced a development process from traditional to deep learning. The rich model proposed by Fridrich and Kodovsky [12] is a typical representative of traditional steganalysis, which combines various functions and uses an integrated classifier for training. In the JPEG domain, some modern schemes, such as DCTR [13] and SCA-GFR [14], extract features from the residuals of the decompressed JPEG images, achieving good results. However, these models also face many difficulties. First, the algorithm design is complicated. Second, adjusting characteristic parameters require a considerable amount of time and energy, leading to low experimental efficiency.

In recent years, the development of deep learning has unified and automated the two steps of feature extraction and classification in traditional image steganalysis, enabling end-to-end methods and achieving satisfactory results. Qian et al. [15] proposed a network structure with five convolutional layers, using the KV kernel as the preprocessing layer to preprocess the images, allowing the model to directly learn the residual images and reduce the interference of the image content in training. XuNet, proposed by Xu et al. [16], used the KV kernel as a high-pass filter layer for image preprocessing. The network used five convolutional layers, and the convolution kernel size of the first two layers was $5 \times 5$. Subsequently, Xu [17] proposed a model specifically for detecting JPEG steganography, referred to as J-XuNet herein. This architecture relies on the fixed preprocessing of the DCT kernels in the first convolutional layer and thresholding its feature maps. Ye et al. [18] proposed YeNet, which uses a deeper ten-layer convolutional network structure and 30 SRM convolution kernels as preprocessing layers to allow the model to learn more features. Chen et al. [19] proposed PNetVnet, which modified XuNet for the steganalysis of JPEG images by splitting the feature maps into 64 parallel channels to make the architecture aware of the JPEG phase the underlying grid of $8 \times 8$ pixels. Boroumand et al. [20] proposed the SRNet which comprises four convolutional layer modules with different functions, effectively using the BN (batch normalization) layer [21] and residual network. Further, it adds channel selection to improve the detection accuracy of the steganography algorithms. Notably, this model provides state-of-the-art detection accuracy for both the spatial domain and JPEG steganography. Zeng et al. [22] proposed WISERNet for steganalysis of color images, which preserves strong correlation patterns while damaging uncorrelated noise, effectively reducing the complexity of model detection performance gains. Zhang et al. [23] proposed a new CNN network structure that uses separable convolution to utilize channel correlation of residuals

to compress image content and improve signal-to-noise ratio, while using data enhancement technology to improve detection accuracy of spatial domain steganalysis. Fu et al. [24] considered that existing steganalysis models lack attention to regional features with complex textures, which affects the detection accuracy of steganalysis. They designed an image steganalysis model based on channel attention mechanism, and guided the model to focus on useful features. In this paper, we focus on improving the SRNet model because it is a classic, relatively pure end-to-end network and is suitable for both spatial and JPEG domains. The structure of the SRNet is shown in Figure 2.

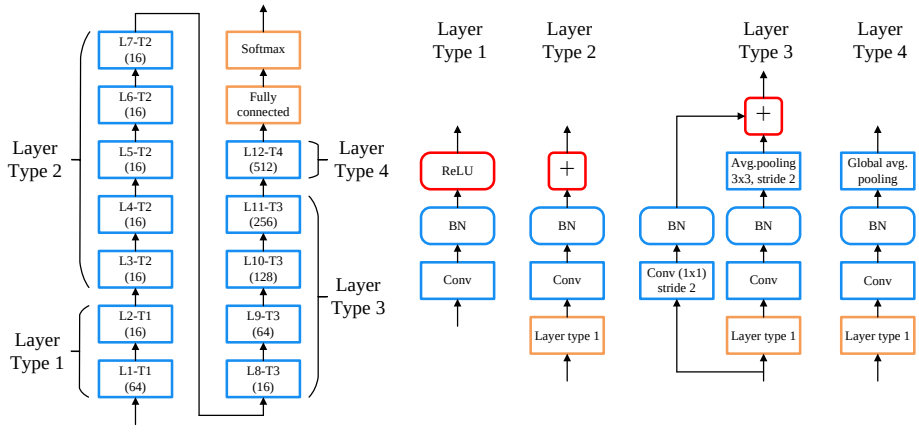



Figure 2. Structure of SRNet. The first type is to replace the traditional pre-processing stage with convolution. The second type is responsible for extracting noise residual in the image. The third type is mainly to reduce the dimension of the feature map. The fourth type mainly uses a standard full connection layer and a softmax node to classify the results.

SRNet is mainly composed of four parts. The first part (Layer Type 1) is to replace the traditional pre-processing stage with convolution. The second part (Layer Type 2) is responsible for extracting noise residual in the image. The third part (Layer Type 3) is mainly to reduce the dimension of the feature map, and the last part (Layer Type 4) mainly uses a standard full connection layer and a softmax node to classify the results. SRNet is the first steganalysis network that does not contain externally introduced modules. Although the pre-processing operations such as constrained kernel, heuristic kernel initialization and quantization can effectively extract favorable features in previous steganalysis studies, the application conditions of steganalysis model are limited to some extent. SRNet has excellent performance in both spatial and JPEG domains due to its pure end-to-end training mode. It also provides an independent and instructive environment for the application of model compression, which is why we use SRNet as the basic model to verify the availability of model compression in this paper.

### 2.2 Model Compression

DNN has become the most powerful tools in machine learning and have a wide range of applications in artificial intelligence tasks. The high performance of DNNs is achieved at the cost of high memory consumption and computational complexity. The existing deep learning frameworks based on DNN, including the image steganalysis model, are parameterized and have high computational cost and storage overhead. As a result, they are more likely to require large amounts of training data to achieve good performance, which significantly affects their deployment in embedded systems. Therefore, model compression has become an important research topic in deep learning. It also helps with the deployment of DNN-based deep models, which enhances the usefulness of the models. Therefore, model compression is proposed and widely used to save memory and speed up computation. Over the past few years, numerous model compression techniques have been developed to optimize and balance the relationship between memory and performance. Because network pruning in model compression is most relevant to the research content of this paper, this section focuses on network pruning.

Increasing network depth has become more complex recently, rendering network pruning a research focus. To solve the challenge of deploying a large central neural network with limited resources, Han et al. [25] discarded weights with small magnitudes. Small weights were set to zero and masked out during retraining. Louizos et al. [26] used hierarchy before pruning nodes nodes and posterior uncertainties to determine fixed-point precision. Zhuang et al. [27] used additional classification and reconstruction losses on intermediate layers to increase the intermediate discriminative power and select channels. Lin et al. [28] utilized generative adversarial learning (GAN) to derive a pruning generator. Liu et al. [8] proposed an approach called network slimming, which accepts wide and large networks as input models. However, during training, insignificant channels are automatically identified and pruned afterward, yielding thin and compact models with comparable accuracy. Considering the redundancy of convolution kernel and the influence of kernel shape on the performance of CNN model, Liu et al. [29] designed a framework for automatically searching the optimal kernel shape and performing strip trimming, which achieved excellent performance in terms of compression ratio and operation efficiency. In order to solve the problem of model underfitting, Jiang et al. [30] proposed a new pruning method, MaskSparsity, which applied fine-grained sparse regularization to a specific filter selected by pruning mask, and achieved better computational accuracy. Unlike many existing approaches, the approach proposed by Liu et al. [8] called network slimming imposes L1 regularization on the scaling factors in BN layers, which makes it is easy to implement without introducing any changes to the existing steganalysis model architectures. It is worth noting that this approach does not require a special software/hardware accelerator for the generated model. These characteristics are applicable to the running and computing scenarios of image steganalysis model and can be pruned at a lower cost.

## 3 OUR PROPOSED SASRNET

### 3.1 Rationale of Our Proposed SASRNet

Considering the large network and complex training problems of image steganalysis, our goal is to design a lightweight and simple method that can slim down the steganalysis model without affecting its detection accuracy. SASRNet learning an efficient steganalysis detector by performing channel-pruning on the convolutional layers. Searching for a more compact and effective channel configuration of convolutional layers can help reduce the trainable parameters and FLOPs. The whole idea is to adopt training-pruning-finetuning scheme. Firstly, sparse training is carried out on SRNet, a pre-training model with huge parameters, and then SASRNet is obtained by channel pruning on the network. Finally, the network is fine-tuned. After doing this once, we get a narrower network, which is not optimal. Therefore, we iterated through the process to get the most compact and effective model possible. Figure 3 shows how this scheme works.

When the depth of steganalysis network is large, the fine-grained sparse weight is more flexible, but it may cause higher compression rate and longer running time and requires special hardware accelerators to assist. Coarse-grained hierarchical sparsity is easy to operate, but it does not achieve high performance. Therefore, in the proposed scheme, we consider using channel sparsity to balance the relationship between performance and efficiency, so as to achieve the purpose of "slimming". However, the application of steganalysis model has some challenges. Channel sparsity requires the input and output associated with it to prune, while the weight value of ports is generally not close to zero, so it is difficult to prune on the pre-training model. So we introduced a simpler way to solve this problem. We introduced a scaling factor for each channel and multiplied it by the output of that channel. Subsequently, we jointly trained the network weights and these scaling factors and performed sparsity regularization. Finally, we pruned those channels with small factors and fine-tuned the pruned network, then obtained a model that can be deployed on the application side. In the following section, we introduce specific implementation details of this method.

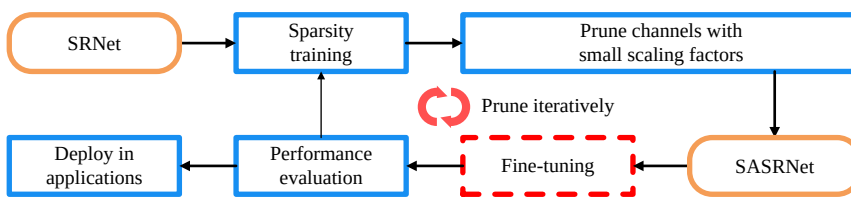



Figure 3. Iterative procedure for learning an efficient steganalysis detector for SASRNet through sparsity training and channel-pruning

### 3.2 Detailed Algorithm of Our Proposed SASRNet

**1) Sparsity Training.** The sparsity training of deep models is helpful for channel pruning and describes the number of less critical channels that can potentially be removed afterward. We want to find a scaling factor for each channel to describe it to facilitate channel pruning, where the absolute values of the scaling factors denote the channel importance. As the convolutional network is designed to be increasingly deeper, the network becomes increasingly difficult to train and converge. This is because the influence of the parameters in the shallow network is amplified in the deep network, changing the input distribution characteristics of each layer. The BN layer solved this problem, and was used as a standard method to achieve fast convergence and better generalization performance in most modern CNNs. The BN layer follows each convolutional layer in the SRNet to accelerate convergence and improve generalization. We adopted trainable scale factors in the BN layers as indicators of channel importance. The BN layer calculation formula is as follows:

$$y = \gamma \times \frac{x - \bar{x}}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \tag{1}$$

where $\bar{x}$ and $\sigma^2$ are the mean and variance of the input features in a mini-batch, respectively, $\gamma$ and $\beta$ denote the trainable scale factor and bias, respectively. We performed channel-wise sparsity training to effectively discriminate important channels from unimportant channels by imposing L1 regularization on $\gamma$. The formula for sparsity training is as follows:

$$L = loss_{SRNet} + \alpha \sum_{\gamma \in \Gamma} f(\gamma), \tag{2}$$

where $f(\gamma) = |\gamma|$ denotes the L1-norm and $\alpha$ denotes the penalty factor that balances the two loss terms. We used the subgradient method to optimize the nonsmooth L1 penalty term. Instead of resorting to group sparsity on convolutional weights, our approach imposes simple L1 sparsity on channel-wise scaling factors; thus, the optimization objective is considerably simpler.

**2) Channel Pruning.** After sparse training, many scaling factors in the SRNet model were approximately zero, as shown in Figure 4. Subsequently, we pruned the channels with a scaling factor close to zero by removing all incoming and outgoing connections and their corresponding weights. We used a global threshold across all the layers to prune the channel. The global threshold is defined as the percentage of all scaling factor values. For example, the threshold was set to 50 % to trim the 50 % channel with a lower scaling factor. Consequently, SRNet has fewer parameters and fewer calculation operations, thereby reducing the memory required at runtime.

In the SRNet, special processing is required for the shortcut layer. In this network structure, the output of a layer may be considered as the input of

multiple subsequent layers, wherein a BN layer is placed before the convolutional layer. In this case, sparsity is achieved at the incoming end of the layer. The layer selectively uses a subset of channels that it receives. To harvest the parameter and computation savings at the test time, we placed a channel selection layer to mask the insignificant channels. After pruning, the resulting narrower network was substantially more compact with respect to model size, runtime memory, and computing operations than the initial wide network.

Pruning a channel essentially corresponds to removing all channel incoming and outgoing connections. Thus, we can directly obtain a narrow network without resorting to any special sparse computing package, as shown in Figure 4. The scaling factors act as agents for channel selection. Because network weights jointly optimize them, the network can automatically identify insignificant channels and safely remove them without affecting the generalization performance significantly.

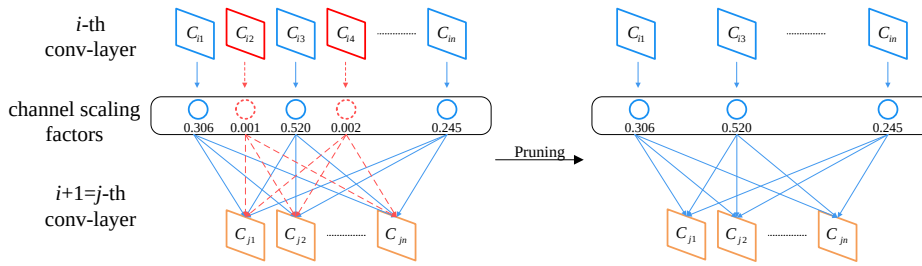



Figure 4. Schematic of channel pruning. Blue realizations represent important channels that cannot be removed and red dotted lines represent insignificant channels that can be removed.

**3) Fine-Tuning.** When the pruning ratio is high, pruning may cause a loss of accuracy. However, this can be compensated by fine-tuning the pruned network. In steganalysis tasks, detection performance is generally sensitive to channel pruning. Therefore, fine-tuning is vital for recovering the pruned model from potential performance degradation. In fine-tuning, we used the same training hyperparameters as the standard training of SRNet to retrain SASRNet directly.

## 4 EXPERIMENTS

### 4.1 Experiment Setup

To assess the robustness of the detector, the datasets BOSSbase 1.01 [31] and BOWS2 [32] were used for the performance evaluation, each containing 10 000 grayscale spatial images with a size of $512 \times 512$. Their distributions were close. All images were resized to $256 \times 256$ pixels using the MATLAB function imresize. The corresponding JPEG images were generated with quality factors (QFs) of 75 and 95.

Four representative steganographic schemes, namely, HILL [33] and HUGO [34] for the spatial domain and UERD [35] and J-UNIWARD [36] for the JPEG domain, were attacking targets in the experiments. For spatial-domain steganographic algorithms, the embedding payloads were set to 0.1 to 0.4 bits per pixel (bpp). For JPEG steganographic algorithms, the embedding payloads were set to 0.1 to 0.4 bits per non-zero AC DCT coefficient (bpnzAC). In all the experiments, we initialized all channel scaling factors $\gamma = 0.5$, $\alpha = 10^{-4}$ to observe the benefits of network slimming. To achieve the best performance of the model, the pruning rate was set to 60 % in the comparative experiments. All tests were performed using Tesla V100 and GeForce RTX3090 GPU cards.

In the experiment, 4 000 images were randomly selected as the training set, 1 000 as the validation set, and 5 000 as the test set. The stochastic gradient descent optimizer Adamax was used with mini-batches of 16 cover-stego pairs. The training database was shuffled after each epoch. In our dataset, the training was run for 400 k iterations with an initial learning rate of 0.001; subsequently, the learning rate was decreased to 0.0001 for an additional 100 k iterations. During the comparison experiment, the method and comparison method proposed in this paper used the same training set, validation set, and test set, and the images of the three datasets were not repeated. The performance was evaluated by testing the accuracy and steganalysis error rates as follows:

$$P_{Accuracy} = 1 - P_E, \tag{3}$$

$$P_E = \frac{1}{2}\left(P_{FA} + P_{FN}\right), \tag{4}$$

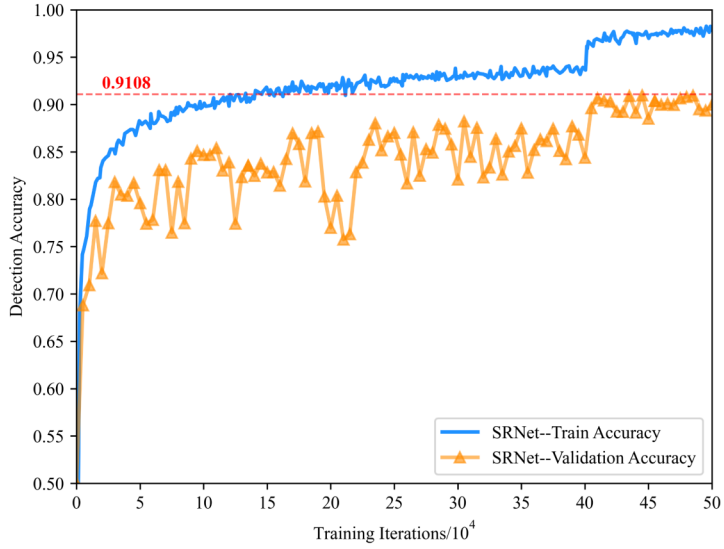
where $P_{FA}$ and $P_{FN}$ denote the probabilities of false positives and false negatives, respectively. The probability of the results was expressed as a percentage.

### 4.2 Detection Performance

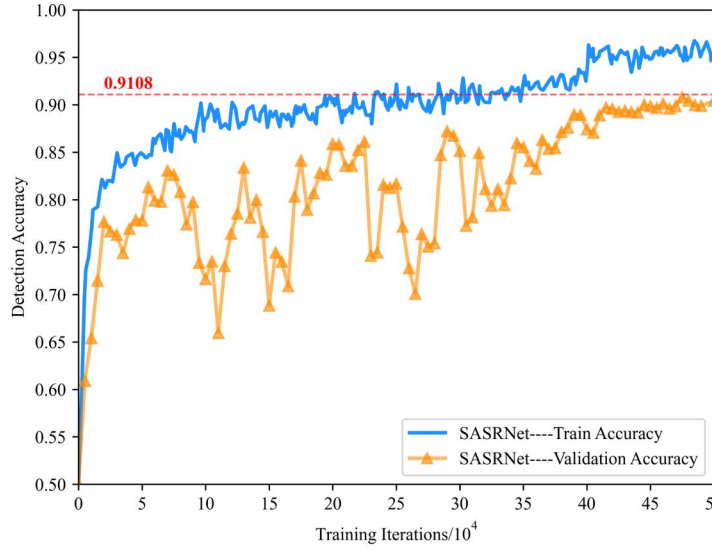
We compared the detection performance of the SASRNet and corresponding original SRNet to verify the performance of the SASRNet. To experiment, we uniformly used the J-UNIWARD steganographic image database; the quality factor was 75, and the embedding rate was 0.4 bpnzAC.

It can be observed from Figure 5 that as the number of parameters is reduced, the detection accuracy of SASRNet after model compression processing does not show excessive fluctuations compared to the original SRNet. In addition, we found in the experiment that the convergence effect of SASRNet in training was not as good as that of the original SRNet. Probably because model representation learning becomes more difficult after the parameters are reduced, leading to greater shocks. This problem provides a direction for improvement in future research.

All detectors were trained using BOSSbase. Testing was performed on the test set of BOSSbase and 5 000 randomly selected images of BOWS in JPEG and the spatial domain to evaluate the performance of SASRNet in every situation. From




a) SRNet




b) SASRNet

Figure 5. Comparison of SRNet and SASRNet training accuracy

the experimental results presented in Table 1 and Table 2, it can be seen that the detection accuracy of SASRNet is not significantly affected after model compression. It even surpasses SRNet in some cases, which we assume is because of pruning certain irrelevant factors.

| Dataset | Steganography | Detection Model | QF75 | | | | QF95 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0.1 | 0.2 | 0.3 | 0.4 | 0.1 | 0.2 | 0.3 | 0.4 |
| BOSSbase (Train & Test) | J-UNIWARD | SASRNet | 64.11 | 79.03 | 86.35 | 91.48 | 59.21 | 68.55 | 75.02 | 80.04 |
| | | SRNet | 64.49 | 79.28 | 86.67 | 91.75 | 59.66 | 68.83 | 75.44 | 80.46 |
| | UERD | SASRNet | 81.2 | 89.47 | 94.32 | 96.66 | 67.29 | 78.11 | 85.39 | 89.03 |
| | | SRNet | 81.61 | 89.85 | 94.33 | 96.58 | 67.6 | 78.41 | 85.73 | 89.38 |
| BOWS2 (Test) | J-UNIWARD | SASRNet | 63.74 | 76.77 | 85.39 | 90.02 | 56.62 | 67.49 | 73.96 | 79.89 |
| | | SRNet | 64.13 | 77.08 | 85.77 | 90.34 | 56.84 | 67.69 | 74.28 | 79.88 |
| | UERD | SASRNet | 80.33 | 89.03 | 93.25 | 95.52 | 67.12 | 77.81 | 85.01 | 88.29 |
| | | SRNet | 80.77 | 89.48 | 93.6 | 95.43 | 67.55 | 78.03 | 85.34 | 88.69 |

Table 1. Comparison of the performance of SRNet and SASRNet in the case of data source mismatch (JPEG domain) [%]

| Dataset | Steganography | Detection Model | 0.1 | 0.2 | 0.3 | 0.4 |
|---|---|---|---|---|---|---|
| BOSSbase (Train & Test) | HILL | SASRNet | 65.17 | 73.22 | 79.74 | 84.34 |
| | | SRNet | 65.53 | 73.41 | 79.72 | 84.25 |
| | HUGO | SASRNet | 71.02 | 80.11 | 86.09 | 90.6 |
| | | SRNet | 71.44 | 80.52 | 86.33 | 90.64 |
| BOWS2 (Test) | HILL | SASRNet | 62.29 | 71.56 | 78.41 | 82.3 |
| | | SRNet | 62.87 | 71.82 | 78.44 | 82.23 |
| | HUGO | SASRNet | 70.39 | 79.31 | 84.92 | 89.85 |
| | | SRNet | 70.53 | 79.66 | 85.01 | 89.88 |

Table 2. Comparison of the performance of SRNet and SASRNet in the case of data source mismatch (spatial domain) [%]

### 4.3 ROC Curve

The receiver operating characteristics (ROC) curve is an essential indicator of the steganalysis model. It reflects the model through the relationship between the true positive rate (TPR) of the vertical axis and false positive rate of the horizontal axis. The discriminative ability in the case of an uneven distribution of positive and negative samples in the dataset indicates the robustness of the model. The area under curve (AUC) is the area covered by the ROC curve. The larger the AUC value, the better is the model detection effect. Figure 6 shows the ROC curve, corresponding AUC value of the SASRNet model proposed in this study, and comparison model

SRNet with different embedding rates and adaptive steganography algorithms, with an image quality factor of 75. As can be observed from the figure, the performance of SASRNet is better than that of SRNet.

In Figure 6, we compare the detection performance of SASRNet and the corresponding original SRNet using the ROC curve. From Figure 6, we can observe that SASRNet achieves comparable detection performance. When used to detect a 0.2 bpp HILL spatial domain stego image dataset, SASRNet can achieve better detection performance. Notably, all high-performance implementations of the SASRNet rely on fewer parameters and require less computation time than the original SRNet.

### 4.4 Mismatch Test

We conducted mismatch tests to investigate the transferability of the obtained SASRNet architecture. The training and test sets often have distribution deviations in an application environment, resulting in mismatch problems. The reason is as follows:

1. The cover data derived from different sources and different imaging devices cause inconsistent noise distribution after cover quantization, which causes mismatch problems in the classification process.
2. In the process of steganographic image generation, different steganographic methods and embedding rates also cause mismatches.

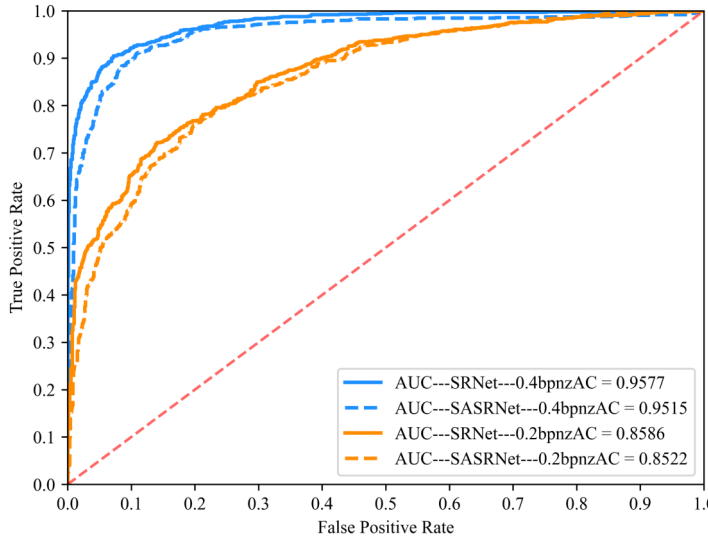
A training model is generated from a training image library with a high embedding rate. The detection result is mismatched when a steganalysis test is performed on a test image library with a low embedding rate.

| SRNet | | J-UNIWARD (Tested) | UERD (Tested) |
|---|---|---|---|
| | J-UNIWARD (Trained) | 91.75 | 93.89 |
| | UERD (Trained) | 78.59 | 96.58 |

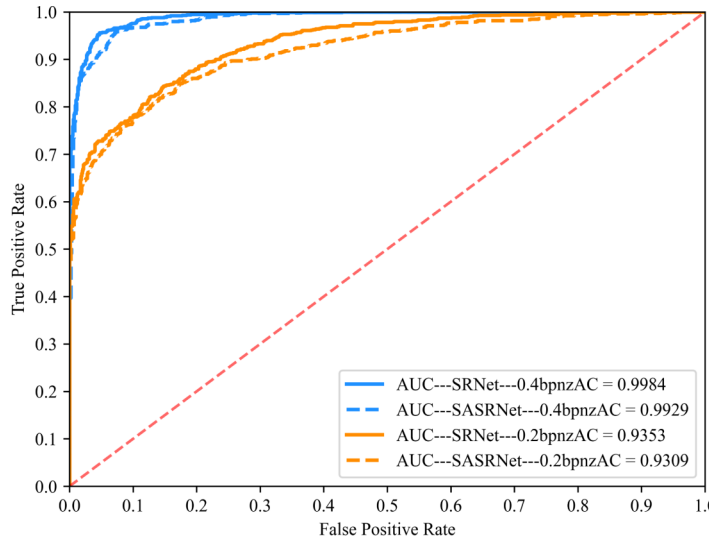
| SASRNet | | J-UNIWARD (Tested) | UERD (Tested) |
|---|---|---|---|
| | J-UNIWARD (Trained) | 91.48 | 93.51 |
| | UERD (Trained) | 79.63 | 96.66 |

Table 3. Comparison of the performance of SRNet and SASRNet in the case of steganography algorithm mismatch (JPEG domain) [%]

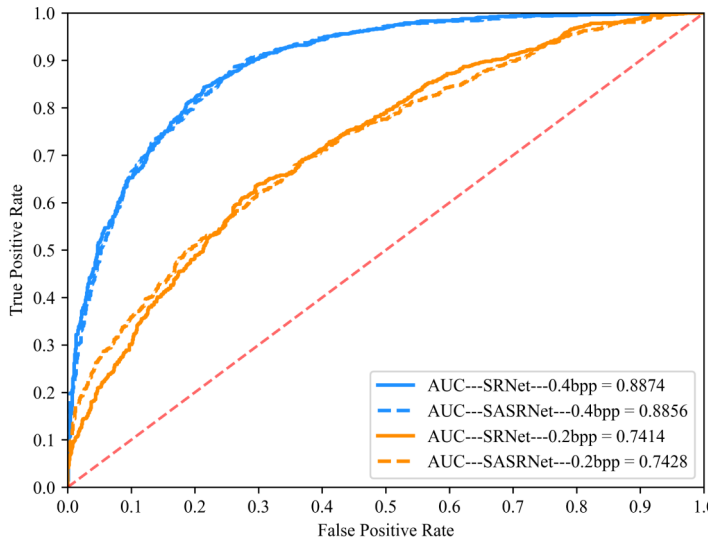
Table 3 and Table 4 compare the detection performances of SASRNet and original SRNet trained on one steganographic algorithm and tested on another. It can be observed from the experimental results that when the target payload is the same (0.4 bpp/bpnzAC), SASRNet can achieve a similar performance as that of the original SRNet.



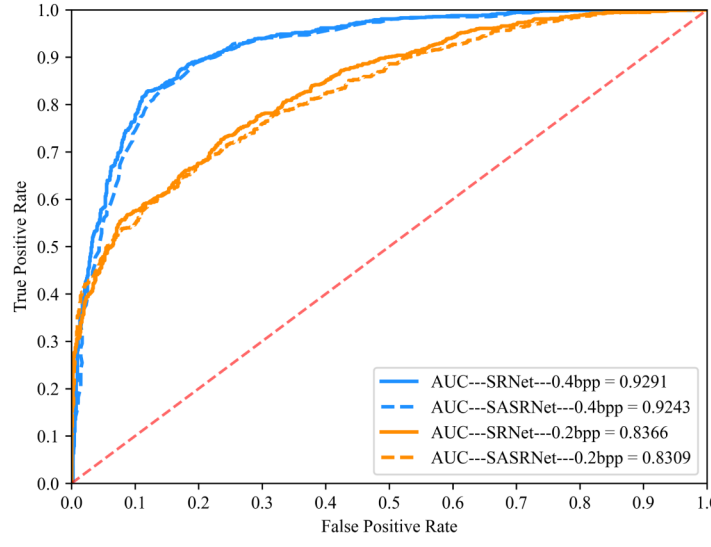

a) J-UNIWARD




b) UERD




c) HILL




d) HUGO

Figure 6. Comparison of the ROC curves for different embedding rates and steganography algorithms

| SRNet | | HILL (Tested) | HUGO (Tested) |
|---|---|---|---|
| | HILL (Trained) | 84.25 | 80.95 |
| | HUGO (Trained) | 75.1 | 90.64 |

| SASRNet | | HILL (Tested) | HUGO (Tested) |
|---|---|---|---|
| | HILL (Trained) | 84.34 | 80.47 |
| | HUGO (Trained) | 74.88 | 90.6 |

Table 4. Comparison of the performance of SRNet and SASRNet in the case of steganography algorithm mismatch (spatial domain) [%]

### 4.5 Slimming Analysis

After obtaining a model trained with sparse regularization, we must determine the percentage of channels that needed to be cut from the model. If we delete an extremely small number of channels, limited resources are saved. However, running many channels may damage the model, and we may not be able to restore accuracy through fine-tuning. We trained an SRNet model on the J-UNIWARD steganographic image dataset with a payload of 0.4 bpnzAC and a QF of 75 to show the effect of pruning different percentages of channels. In Figure 7, we show the variation in test accuracy with respect to the pruning rate.

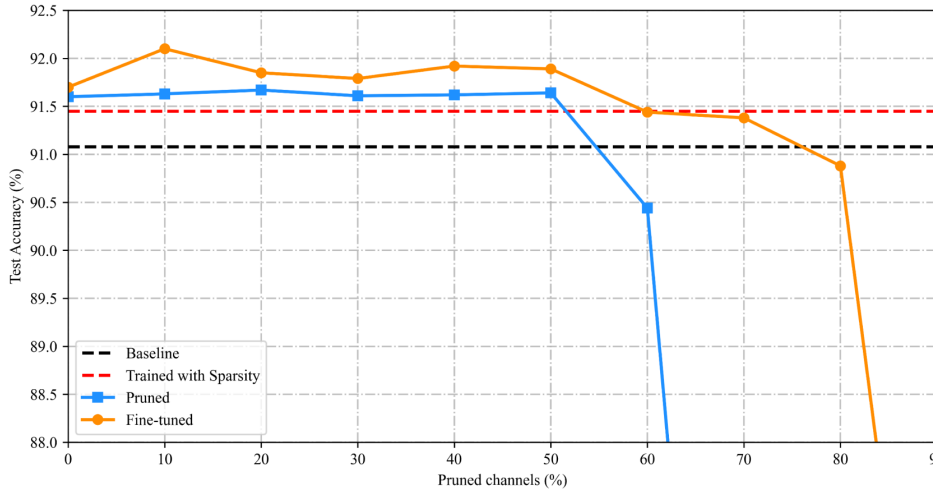



Figure 7. Relationship between detection accuracy and growth pruning rate

Figure 7 shows that the classification performance of the pruning or fine-tuning model decreases only when the pruning rate exceeds the threshold. The fine-tuning process can usually compensate for the possible loss of accuracy owing to trimming. However, when the threshold exceeds 75 %, the test error of the fine-tuned model

begins to drop below the baseline model. In the experiment, we also found that after sparsity training, even without fine-tuning, the performance of the proposed model was better than that of the original model. This may be because of the regularization effect of L1 sparsity on the channel scaling factors.

| Model | Pruned | Accuracy | Parameters | FLOPs | Time |
|---|---|---|---|---|---|
| SRNet | – | 96.58 % | $477.61 \times 10^4$ | $5.94 \times 10^9$ | 41 s |
| SASRNet | 40 % | 96.67 % | $25.97 \times 10^4$ | $3.45 \times 10^9$ | 38 s |
| | 60 % | 96.66 % | $8.66 \times 10^4$ | $2.97 \times 10^9$ | 25 s |
| | 80 % | 93.34 % | $2.39 \times 10^4$ | $0.83 \times 10^9$ | 20 s |

Table 5. Comparison of the impact of different pruned rates on the model

Finally, we compared the detection accuracies, parameter rate FLOPs, and test times at different pruning rates on a UERD with a load of 0.4 bpnzAC and a QF of 75 to reflect the optimal performance of the model, as shown in Table 5. Note that in Table 5 we list one of our most successful experiments. As can be seen from the table, as the pruning rate increases, the detection accuracy of the model increases, and it can achieve better performance with less than 5 % of the parameters, while the running time is shortened. This is an exciting result, and we analyze that it may be due to the fact that the model is more focused and efficient in training after removing the redundant factors. However, we have to admit that the performance of the model starts to decrease after the pruning rate reaches 80 %. Therefore, the appropriate pruning rate is also very important for different task requirements.

| Steganography Algorithm | Model | Accuracy (40 %) | Accuracy (80 %) |
|---|---|---|---|
| J-UNIWARD | XuNet | 86.23 % | |
| | SAXuNet | 86.47 % | 85.93 % |
| UERD | XuNet | 91.17 % | |
| | SAXuNet | 91.15 % | 90.87 % |

Table 6. Results of using slimming methods on XuNet

In this paper, we mainly improve on the SRNet model because of its advancement and universality in the field of steganalysis. In order to further illustrate the generalization of our proposed method on steganalysis models, we also try to deploy it on the classical model, XuNet, the model SAXuNet was constructed. As shown in Table 6, the feasibility and generalization of the scheme is verified with different steganography algorithms and different pruning rates (40 %, 80 %).

## 5 CONCLUSIONS

With the increasingly complex structure of image steganalysis model, its huge number of parameters leads to more and more difficult deployment on mobile terminals.

In this paper we propose SASRNet, a method using network slimming to the assisted steganalysis residual model. The major contributions of this work are as follows:

- We have observed that the trainable scale factor of BN layer in steganalysis network can be used as channel scaling factor for network slimming, which can be easily implemented without changing the existing steganalysis network architecture.
- We have proposed a slimming-assisted deep residual network architecture, which uses a model compression technology to balance performance and efficiency of the network without using a special accelerator.
- We have conducted extensive experiments on datasets with different circumstances. Moreover, we carried out mismatch test to verify practicability of our method. The experimental results show the SASRNet can still achieve similar performance with a few percent of the model size.

Our future work will focus on two aspects:

1. To design an adaptive compression algorithm based on different steganalysis models so that the depth models can be easily deployed;
2. To futher explore the way how to improve the model detection accuracy and operating efficiency under compression.

**Acknowledgement**

This work was supported by the National Natural Science Foundation of China (Grants No. 61872384, 62272478, 62102450, 62102451).

## REFERENCES

[1] Deng, X. Q.—Chen, B. L.—Luo, W. Q.—Luo, D.: Universal Image Steganalysis Based on Convolutional Neural Network with Global Covariance Pooling. Journal of Computer Science and Technology, Vol. 37, 2022, No. 5, pp. 1134–1145, doi: 10.1007/S11390-021-0572-0.

[2] Zhou, H.—Zhang, W.—Chen, K.—Li, W.—Yu, N.: Three-Dimensional Mesh Steganography and Steganalysis: A Review. IEEE Transactions on Visualization and Computer Graphics, Vol. 28, 2022, No. 12, pp. 5006–5025, doi: 10.1109/TVCG.2021.3075136.

[3] Tan, S.—Wu, W.—Shao, Z.—Li, Q.—Li, B.—Huang, J.: CALPA-NET: Channel-Pruning-Assisted Deep Residual Network for Steganalysis of Digital Images. IEEE Transactions on Information Forensics and Security, Vol. 16, 2021, pp. 131–146, doi: 10.1109/TIFS.2020.3005304.

[4] Xu, S.—Zhang, M.—Song, W.—Mei, H.—He, Q.—Liotta, A.: A Systematic Review and Analysis of Deep Learning-Based Underwater Object Detection. Neurocomputing, Vol. 527, 2023, pp. 204–232, doi: 10.1016/j.neucom.2023.01.056.

[5] Li, X.—Yang, X.—Ma, Z.—Xue, J. H.: Deep Metric Learning for Few-Shot Image Classification: A Review of Recent Developments. Pattern Recognition, Vol. 138, 2023, Art. No. 109381, doi: 10.1016/j.patcog.2023.109381.

[6] Wang, Q.—Piao, Y.: Depth Estimation of Supervised Monocular Images Based on Semantic Segmentation. Journal of Visual Communication and Image Representation, Vol. 90, 2023, Art. No. 103753, doi: 10.1016/j.jvcir.2023.103753.

[7] He, K.—Zhang, X.—Ren, S.—Sun, J.: Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[8] Liu, Z.—Li, J.—Shen, Z.—Huang, G.—Yan, S.—Zhang, C.: Learning Efficient Convolutional Networks Through Network Slimming. 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2755–2763, doi: 10.1109/ICCV.2017.298.

[9] Liu, Y.—Cao, J.—Li, B.—Hu, W.—Maybank, S.: Learning to Explore Distillability and Sparsability: A Joint Framework for Model Compression. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 45, 2023, No. 3, pp. 3378–3395, doi: 10.1109/TPAMI.2022.3185317.

[10] Wang, Z.—Luo, T.—Goh, R. S. M.—Zhou, J. T.: EDCompress: Energy-Aware Model Compression for Dataflows. IEEE Transactions on Neural Networks and Learning Systems, Vol. 35, 2024, No. 1, pp. 208–220, doi: 10.1109/TNNLS.2022.3172941.

[11] Deng, L.—Li, G.—Han, S.—Shi, L.—Xie, Y.: Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. Proceedings of the IEEE, Vol. 108, 2020, No. 4, pp. 485–532, doi: 10.1109/JPROC.2020.2976475.

[12] Fridrich, J.—Kodovsky, J.: Rich Models for Steganalysis of Digital Images. IEEE Transactions on Information Forensics and Security, Vol. 7, 2012, No. 3, pp. 868–882, doi: 10.1109/TIFS.2012.2190402.

[13] Holub, V.—Fridrich, J.: Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT. IEEE Transactions on Information Forensics and Security, Vol. 10, 2015, No. 2, pp. 219–228, doi: 10.1109/TIFS.2014.2364918.

[14] Denemark, T.—Sedighi, V.—Holub, V.—Cogranne, R.—Fridrich, J.: Selection-Channel-Aware Rich Model for Steganalysis of Digital Images. 2014 IEEE International Workshop on Information Forensics and Security (WIFS), 2014, pp. 48–53, doi: 10.1109/WIFS.2014.7084302.

[15] Qian, Y.—Dong, J.—Wang, W.—Tan, T.: Deep Learning for Steganalysis via Convolutional Neural Networks. Media Watermarking, Security, and Forensics 2015, Proceedings of SPIE, Vol. 9409, 2015, pp. 171–180, doi: 10.1117/12.2083479.

[16] Xu, G.—Wu, H. Z.—Shi, Y. Q.: Structural Design of Convolutional Neural Networks for Steganalysis. IEEE Signal Processing Letters, Vol. 23, 2016, No. 5, pp. 708–712, doi: 10.1109/LSP.2016.2548421.

[17] Xu, G.: Deep Convolutional Neural Network to Detect J-UNIWARD. Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security (IH & MMSec '17), 2017, pp. 67–73, doi: 10.1145/3082031.3083236.

[18] Ye, J.—Ni, J.—Yi, Y.: Deep Learning Hierarchical Representations for Image Steganalysis. IEEE Transactions on Information Forensics and Security, Vol. 12, 2017,

No. 11, pp. 2545–2557, doi: 10.1109/TIFS.2017.2710946.

[19] Chen, M.—Sedighi, V.—Boroumand, M.—Fridrich, J.: JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images. Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security (IH & MMSec '17), 2017, pp. 75–84, doi: 10.1145/3082031.3083248.

[20] Boroumand, M.—Chen, M.—Fridrich, J.: Deep Residual Network for Steganalysis of Digital Images. IEEE Transactions on Information Forensics and Security, Vol. 14, 2019, No. 5, pp. 1181–1193, doi: 10.1109/TIFS.2018.2871749.

[21] Ioffe, S.—Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Bach, F., Blei, D. (Eds.): Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research (PMLR), Vol. 37, 2015, pp. 448–456, http://proceedings.mlr.press/v37/ioffe15.html.

[22] Zeng, J.—Tan, S.—Liu, G.—Li, B.—Huang, J.: WISERNet: Wider Separate-Then-Reunion Network for Steganalysis of Color Images. IEEE Transactions on Information Forensics and Security, Vol. 14, 2019, No. 10, pp. 2735–2748, doi: 10.1109/TIFS.2019.2904413.

[23] Zhang, R.—Zhu, F.—Liu, J.—Liu, G.: Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis. IEEE Transactions on Information Forensics and Security, Vol. 15, 2020, pp. 1138–1150, doi: 10.1109/TIFS.2019.2936913.

[24] Fu, T.—Chen, L.—Fu, Z.—Yu, K.—Wang, Y.: CCNet: CNN Model with Channel Attention and Convolutional Pooling Mechanism for Spatial Image Steganalysis. Journal of Visual Communication and Image Representation, Vol. 88, 2022, Art. No. 103633, doi: 10.1016/j.jvcir.2022.103633.

[25] Han, S.—Pool, J.—Tran, J.—Dally, W.: Learning Both Weights and Connections for Efficient Neural Network. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 28 (NIPS 2015). Curran Associates, Inc., 2015, pp. 1135–1143, https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

[26] Louizos, C.—Ullrich, K.—Welling, M.: Bayesian Compression for Deep Learning. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran Associates, Inc., 2017, pp. 3288–3298, https://proceedings.neurips.cc/paper_files/paper/2017/file/69d1fc78dbda242c43ad6590368912d4-Paper.pdf.

[27] Zhuang, Z.—Tan, M.—Zhuang, B.—Liu, J.—Guo, Y.—Wu, Q.—Huang, J.—Zhu, J.: Discrimination-Aware Channel Pruning for Deep Neural Networks. Advances in Neural Information Processing Systems 31 (NeurIPS 2018), 2018, pp. 875–886, https://proceedings.neurips.cc/paper_files/paper/2018/file/55a7cf9c71f1c9c495413f934dd1a158-Paper.pdf.

[28] Lin, S.—Ji, R.—Yan, C.—Zhang, B.—Cao, L.—Ye, Q.—Huang, F.—Doermann, D.: Towards Optimal Structured CNN Pruning via Generative Ad-

versarial Learning. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2785–2794, doi: 10.1109/CVPR.2019.00290.

[29] Liu, G.—Zhang, K.—Lv, M.: SOKS: Automatic Searching of the Optimal Kernel Shapes for Stripe-Wise Network Pruning. IEEE Transactions on Neural Networks and Learning Systems, Vol. 34, 2023, No. 12, pp. 9912–9924, doi: 10.1109/TNNLS.2022.3162067.

[30] Jiang, N. F.—Zhao, X.—Zhao, C. Y.—An, Y. Q.—Tang, M.—Wang, J. Q.: Pruning-Aware Sparse Regularization for Network Pruning. Machine Intelligence Research, Vol. 20, 2023, No. 1, pp. 109–120, doi: 10.1007/s11633-022-1353-0.

[31] Bas, P.—Filler, T.—Pevný, T.: "Break Our Steganographic System": The Ins and Outs of Organizing BOSS. In: Filler, T., Pevný, T., Craver, S., Ker, A. (Eds.): Information Hiding (IH 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6958, 2011, pp. 59–70, doi: 10.1007/978-3-642-24178-9_5.

[32] Bas, P.—Furon, T.: Bows-2. 2019, `http://bows2.ec-lille.fr`.

[33] Li, B.—Wang, M.—Huang, J.—Li, X.: A New Cost Function for Spatial Image Steganography. 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 4206–4210, doi: 10.1109/ICIP.2014.7025854.

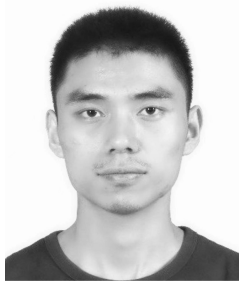
[34] Pevný, T.—Filler, T.—Bas, P.: Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. Information Hiding (IH 2010).

[35] Guo, L.—Ni, J.—Su, W.—Tang, C.—Shi, Y. Q.: Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited. IEEE Transactions on Information Forensics and Security, Vol. 10, 2015, No. 12, pp. 2669–2680, doi: 10.1109/TIFS.2015.2473815.

[36] Holub, V.—Fridrich, J.—Denemark, T.: Universal Distortion Function for Steganography in an Arbitrary Domain. EURASIP Journal on Information Security, Vol. 2014, 2014, No. 1, Art. No. 1, doi: 10.1186/1687-417X-2014-1.

**Siyuan Huang** received his M.Sc. degree in cryptography from the Engineering University of PAP, Xi'an, China in 2021, where he is working toward his Ph.D. degree. His research interests include information hiding, image steganalysis and deep learning.

**Minqing Zhang** received her M.Sc. degree in computer science and application from the Northwestern Polytechnical University, Xi'an, China, in 2001, and Ph.D. degree in network and information security from the Northwestern Polytechnical University, Xi'an, China, in 2016. Currently, she serves as Professor with the Key Laboratory of Network and Information Security under the Chinese People's Armed Police Force in the School of Cryptography Engineering in the Engineering University of PAP. Her research interests include cryptography, and trusted computation.



**Yan Ke** received his M.Sc. degree in cryptography from the Engineering University of PAP, Xi'an, China in 2016 and the Ph.D. degree in cryptography at the Engineering University of PAP. He is also with the Key Laboratory of Network and Information Security under the Chinese People's Armed Police Force in the School of Cryptography Engineering in the Engineering University of PAP. His research interest includes information hiding, lattice based cryptography, multi-party computing security.

**Fuqiang Di** received his M.Sc. degree and Ph.D. degree in information security from the Chinese People's Armed Police Force, Engineering University, China, in 2015 and 2019, respectively. He is currently Assistant Professor with the Engineering University of PAP, Xi'an, China. His research interests include multimedia security, image processing, and information hiding.

**Yongjun Kong** received his M.Sc. degree in cryptography from the Engineering University of PAP, Xi'an, China in 2019, where he is working toward his Ph.D. degree. His research interests include information security, reversible data hiding, and image application based on neural network image generators.