Self-attention enhanced deep residual network for spatial image steganalysis.

XIE, G., REN, J., MARSHALL, S., ZHAO, H., LI, R. and CHEN, R.

2023

Self-attention enhanced deep residual network for spatial image steganalysis

Guoliang Xie [a,b], Jinchang Ren [a,c,*], Stephen Marshall [b], Huimin Zhao [a], Rui Li [a,d], Rongjun Chen [a]

[a] School of Computing Sciences, Guangdong Polytechnic Normal University, Guangzhou, 510640, China
[b] Dept. of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, G1 1XQ, UK
[c] National Subsea Center, Robert Gordon University, Aberdeen, AB21 0BH, UK
[d] School of Art and Design, Guangzhou College of Commerce, Guangzhou, 511363, China

* Corresponding author at: National Subsea Center, Robert Gordon University, Aberdeen, AB21 0BH, UK.
E-mail address: jinchang.ren@ieee.org (J. Ren).

ABSTRACT

As a specially designed tool and technique for the detection of image steganography, image steganalysis conceals information under the carriers for covert communications. Being developed on the BOSSbase dataset and released a decade ago, most of the Convolutional Neural Network (CNN) architectures for spatial image steganalysis fail to achieve satisfactory performance on new challenging datasets, i.e. ALASKA#2, which was released recently and is more complex yet consistent with the real scenarios. In this paper, we propose an enhanced residual network (ERANet) with self-attention ability, which utilizes a more complex residual method and a global self-attention technique, to alleviate the problem. Compared to the residual network that was widely used in the state-of-the-art, the enhanced residual network mathematically employed a more sophisticated way to extract more effective features in the images and hence it is suitable for more complex situations in the new dataset. Our proposed Enhanced Low-Level Feature Representation Module can be easily mounted on other CNNs in selecting the most representative features. Although it comes with a slightly extra computational cost, comprehensive experiments on the BOSSbase and ALASKA#2 datasets at various sizes have demonstrated the effectiveness of the proposed methodology. In short, ERANet provides an improvement of about 3.77% on average, compared to a few state-of-the-art CNNs.

Keywords:
Image steganography; Spatial image steganalysis; Res2Net; BoTNet; Convolutional Neural Network (CNN)

1. Introduction

Steganography is a state-of-the-art technique of hiding secure messages inside multimedia files for encrypted communications, which include images and audio [1–4]. In this paper, image steganalysis techniques with Convolutional Neural Networks (CNN) are investigated. Although CNNs are usually computationally expensive, they are currently the most effective way to detect image steganography and have the potential to significantly advance image steganalysis [5].

In 2014, a convolutional auto-encoder was first introduced to image steganalysis [6], in which only one steganographic algorithm, the Highly Undetectable steGo (HUGO) [7], was investigated.

However, the CNN has demonstrated the potential in image steganalysis by including a high-pass kernel from the Spatial Rich Model (SRM) [8]. In [9], GNCNN was proposed, which shows better performance than the hand-crafted feature extractor, the Subtractive Pixel Adjacency Matrix (SPAM) method [10] in three different payloads, i.e. 0.3 bpp (bit-per-pixel), 0.4bpp and 0.5 bpp. Aiming to go deeper, in 2016, Xu et al. proposed their classic Xu-Net, the performance of which was superior to SRM in detecting the "High-pass, Low-pass, and Low-pass" steganography (HILL) [11] and the Spatial-UNIversal wavelet relative distortion (SUNI) [12] under the payload of 0.4 bpp [13].

Since training a CNN in image steganalysis is usually time-consuming, Qian et al. hoped to resort to the transfer learning [14] in image steganalysis, which showed that the parameters in the pre-trained CNN could be utilized in fine-tuning a CNN to detect the low-payload stego images. In most cases, ensemble learning could boost the performance of machine learning algorithms, Xu et al. also investigated this technique in image steganalysis in [15], and found that "learning from intermediate representation" of the CNNs, rather than output probabilities, could help to improve the performance.

Previous methods only utilized a single high-pass filter in the pre-processing layer, which might hinder the extraction of details. As a result, researchers tried to extract more different features by utilizing multiple high-pass filters. Ye et al. proposed their Thresholded Liner Unit (TLU)-CNN [16], which outperformed the best steganalysis method implemented with the hand-crafted feature extractor, maxSRM and an ensemble classifier [17]. In contrast to the previous CNN models, 30 high-pass filters from the SRM were used to extract different kinds of stego noise.

Although the dataset they used seems to be heterogeneous, it was believed that the size of the dataset might prevent the CNN from learning more features from them. To solve this problem, in [18], the Yedrouj-Net was proposed in 2018 with better results than the TLU-CNN, which suggested that the CNNs could always benefit from the "virtual augmentation" of the dataset, requiring label-preserving flips and rotations only. With this insight, Boroumand et al. proposed a powerful network architecture called SRNet [19], which was designed with little externally introduced knowledge, such as fixed kernels, thresholding and quantization. SRNet could outperform the SCA-TLU-CNN, i.e. the TLU-CNN with the selection-channel, and the SRNet could also benefit from the selection-channel information.

Previous CNNs consist of one CNN only, and researchers wondered whether using multiple CNNs could improve the performance or not. In the ReST-Net [20], a parallel subnet architec-ture was introduced, where the SRM filters, Gabor filters and the nonlinearity after SRM filtering in the pre-processing layer were analyzed. Different architectures were also considered to improve the detection accuracy, Zhu-Net [21] managed to achieve better results than the SRNet by using spatial pyramid pooling and depth-wise separable convolutions and shortcuts. Recently, an efficient yet powerful network named SiaStegNet was proposed [22]. Un-like ReST-Net, the two subnets in the SiaStegNet were identical and shared the same parameters for efficiency. On the BOSSbase dataset [23], the SiaStegNet achieved comparable performance to the SRNet, though it had only 0.7M trainable parameters compared to 4.7M for SRNet.

To fullying utilize the hidden information, Lai et al. investigated the residual networks in a generative algorithm for the extraction of convert information [24]. Aiming to explore more complex regional features, Fu et al. employed the Squeeze-and-Excitation module in the residual block in their module, which showed a better generalization ability than SRNet and Zhu-Net [25]. Different from the CNN methods, Arivazhagan et al. resorted to the Empirical Mode Decomposition technique to extract the stego noise, which was then enhanced for classifications [26]. More detail on the development of deep learning in image steganalysis can be found in [27–29].

Due to the limitation of the diversity of the images, the existing CNNs that were trained on the limited datasets can not provide a satisfactory performance on the more challenging datasets and the unseen scenarios. Moreover, one may ask if a more complex residual network is used, i.e. a deeper or more effective feature representation technique, will it help to move the image steganalysis further to real-life scenarios? To tackle these issues, a residual network with an enhanced low-level feature representation module is proposed for the effective detection of stego noise in the images. We aim to deliver highly discriminative features from a well-structured residual network while keeping the parameters on a controllable scale. To achieve this, the Res2Net [30] is taken as the backbone to build our feature extractor. Meanwhile, the self-attention mechanism [31] is used to construct our Enhanced Low-level Feature Representation Module (ELLFRM).

The main contributions of this paper can be summarized as follows:

- We constructed an enhanced low-level feature representation module, which can greatly improve the feature receptive field without significantly increasing the parameters. The proposed ELLFRM can effectively capture the pattern of the stego noise, which can even improve the performance of other CNNs, validating its effectiveness and versatility.
- We proposed an effective residual network with the self-attention capability. The network has been confirmed to be fast converging while providing state-of-the-art performance without introducing too many parameters or requiring too much memory of GPUs.

Currently, on the ALASKA#2 dataset, most new CNN architectures are developed for JPEG images, yet the proposed method is extendable for spatial images. Our experiments have validated the efficacy of current architectures for spatial image steganalysis, i.e. keeping the input image size basically unchanged during the feature *processing* stage while increasing the mappings rapidly in the feature *selection* stage.

The remaining paper is organized as follows. Section 2 introduces the related work and explains why these models are used. In Section 3, the details of the proposed architecture and the experimental settings are presented. The discussion of the features and an ablation study are provided in Section 4. The comparison with the state-of-the-art is given in Section 5, including transfer learning results. Finally, some concluding remarks are drawn in Section 6.

2. Theoretic background from the ResNet to self-attention module

In this section, we first briefly introduce the related work and also explain why these CNN architectures can be applied in image steganalysis. We will show that these CNN networks are theoretically feasible for the image steganalysis tasks.

2.1. Image steganalysis with the ResNet

Currently, many spatial steganalysis works rely on the residual architecture proposed in [32], for example, [33], [22] and [19]. Let C denote a cover image with a size of $n_1 \times n_2$, and M denote the secret message in the form of a matrix with the same size. $C = (C_{ij}) \in \{0, \ldots, 255\}^{n_1 \times n_2}$, $M = (M_{ij}) \in \{-1, 0, +1\}^{n_1 \times n_2}$. Let X denote the input image of a CNN, which can be either the cover image or the stego image. The objective of a CNN model is to tell if X is a cover or a stego, just as Eq. (1) shows.

$$X = \begin{cases} C + 0, & cover \\ C + M, & stego \end{cases} \tag{1}$$

To better illustrate the principle of the ResNet, the basic build-ing blocks from [32] are shown in Fig. 1 (a), in which the convolutional operation $W(\cdot)$ maps the input to the output $H(X)$, i.e. $H(X) = W(X)$. In a residual network, the residual information $F(X)$ is calculated as shown in Eq. (2) [33].

The extremely weak stego signal M can be effectively captured by the residual mapping network [33], which will then be "preserved and emphasized through the whole network". In a typical residual network, i.e. Fig. 1 (b), the residual information is usually computed as shown in Eq. (4), where $W_1(\cdot)$ is the convolutional operation with a kernel size of 1 and $W_3(\cdot)$ denote a kernel size of 3. The batch normalizations are not shown for simplicity.

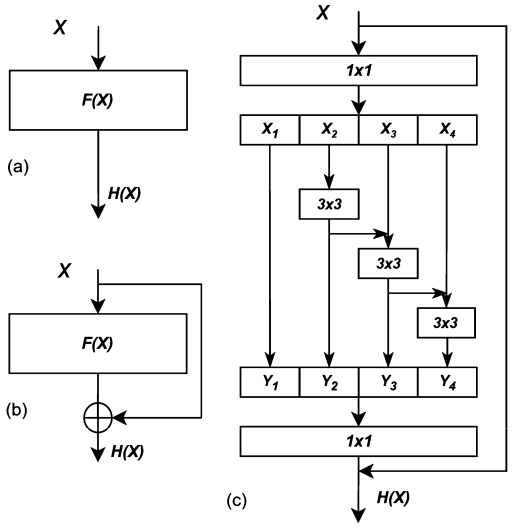

Fig. 1. Different basic blocks in CNNs: (a) a basic convolutional block in a typical CNN model; (b) a residual block; (c) an improved residual block from the Res2Net.

We denote $Y_{Res}(X) = W_3(W_1(X))$ as the tensor before entering the last convolutional layer in a residual block, thus $H(X)$ can be written into Eq. (4) for clarity.

$$F(X) := H(X) - X \tag{2}$$

$$Y_{Res}(X) = W_3(W_1(X)) \tag{3}$$

$$H_{Res}(X) = W_1(Y_{Res}(X)) + X \tag{4}$$

2.2. Theoretical consistency of the Res2Net

Aiming to provide a more effective method in representing features at multiple scales, the Res2Net was proposed [30]. In [32], Gao et al. realized that the original 3×3 filters in ResNet [32], may not provide enough receptive fields for CNNs, thus they replaced those filters with a set of small filter groups. The key to increasing the receptive field is these small filter groups, as typically illustrated in Fig. 1 (c).

In these sets of small filters, each group of filters will extract the corresponding features from the input features maps. The number of the feature groups is called "scales", S. The output fea-tures from the previous group are sent to the next group of filters. Feature maps from all groups are concatenated for the last 1×1 filters to fuse the information before all the input feature maps are processed [30]. Let Y_{Res2} denote the tensor before entering the last convolutional layer in a Res2Block, we can rewrite the residual mapping Eq. (4) as Eqs. (5) to (7), where $x_i = W_1(X)$, and $\frown$ is concatenation operation.

$$H_{Res2}(X) = W_1(Y_{Res2}(X)) + X \tag{5}$$

$$Y_{Res2}(X) = y_1 \frown y_2 \frown \ldots y_i \ldots \frown y_S \tag{6}$$

$$y_i = \begin{cases} x_i, & i = 1; \\ W_3(x_i), & i = 2; \\ W_3(x_i + y_{i-1}), & 2 < i \leq S \end{cases} \tag{7}$$

2.3. The self-attention mechanism

Recently, self-attention mechanism is investigated for computer vision tasks [31]. We are considering the self-attention mechanism as we would like a building block in the CNN model to provide good global attention ability. This is because in deep layers of CNN, the features are way more complex and not easy to understand, where a block with self-attention mechanism ability may help. As the convolutional operator is limited by its locality and lack of understanding of global contexts [34], the global attention mechanism is preferred for extraction of a statistical summary of the whole scene [35]. This ability is particularly helpful in extracting the extremely weak stego signal.

In [36], Vaswani et al. proposed an attention function, which describes a mapping between a query vector and a set of key-value pair vectors as the output. Let W_q, W_k, W_v denote the weights of the query, key and value, respectively, we can calculate the matrices of queries, Q, keys, K, and values, V in Eq. (8). With Q, K and V, we can determine the attention function for a single head y_1 as given in Eq. (9), where d_k is the dimension of keys and $f_s(\cdot)$ is a Softmax function. It is also suggested that projecting the queries, keys and values N times is helpful and hence multiple heads are used.

$$Q = W_q X, K = W_k X, V = W_v X \tag{8}$$

$$y_1 = f_s(\frac{1}{\sqrt{d_k}} Q K^T) V \tag{9}$$

By introducing the encoding of positional information in the attention mechanism, Ramachandran et al. [37] incorporated this kind of self-attention mechanism into their multi-head projection (MHSA) architecture. In [38], the content-content interaction and content-position interaction are used. Among these architectures, in [31], the Bottleneck Transformer block (BoTBlock) architecture is proposed. Following their idea, without changing the residual model given in Eq. (10), the output tensor Y_{BOT} can be calculated in Eq. (11). The sub-tensors y_i in Eq. (7) is now rewritten as Eq. (12), where d_i is the dimension of the keys in the ith head, $i = 1, \ldots, N$. The calculations of the matrices Q_i, K_i, V_i are shown in Eq. (13), where $W_q, W_k, W_v \in \mathbf{R}^{d_{in} \times d_{out}}$. d_{in} and d_{out} represent the dimension of the input and output, respectively.

$$H_{BoT}(X) = W_1(Y_{BoT}(X)) + X \tag{10}$$

$$Y_{BoT}(X) = y_1 \frown y_2 \frown \ldots y_i \ldots \frown y_N \tag{11}$$

$$y_i = f_s(\frac{1}{\sqrt{d_i}} Q_i K_i^T + Q_i R_h^T + Q_i R_w^T) V_i \tag{12}$$

$$Q_i = W_q X, K_i = W_k X, V_i = W_v X \tag{13}$$

As indicated in [39], "the memory and computation for self-attention scale quadratically with the spatial dimension", we will show how many BoTBlocks should be used in our architecture to reach the best balance between the computation complexity and performance in Section 4.3.4.

In short, compared to the original residual information of a residual block in ResNet, Eq. (4), although the definition of Res2Net, Eq. (5), and BotNet, Eq. (10), share the same formula, i.e. $H_{Res}(X)$, $H_{Res2}(X)$, $H_{BoT}(X)$, they are calculated in totally different ways. One can easily find that the latter two kinds of residual information are more sophisticated and hence can yield more complicated information for the CNN.

3. The proposed method

In this section, we will discuss our proposed model and its modules in detail. We will explain why these modules are used and how we process the features, especially explaining how the low-level features are enhanced via our ELLFRM module. Lastly, the implementation details are provided.

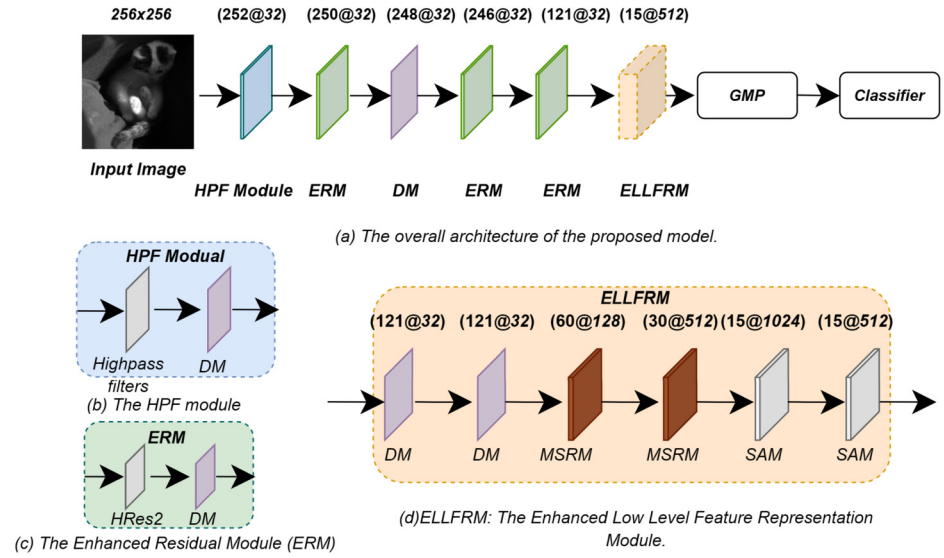

Fig. 2. The overall architecture of the proposed model, ERANet. DM is short for Downsampling Module, MSRM is the Multi-Stage Residual Module and SAM is short for the Self-attention module. GMP represents the Global Maximum Pooling. Inside each bracket, the first number is the width of the image and the second is the number of output channels.

3.1. The network architecture

The overall architecture of the proposed ERANet is shown in Fig. 2, which is composed of four different modules, i.e. the Highpass-filter module (HPF Module), the Enhanced Residual Mod-ule (ERM), the Downsampling Module and the Enhanced Low-level Feature Representation Module (ELLFRM). A Global Maximum Pooling layer follows after the ELLFRM module.

The ERANet can be roughly divided into two stages. The first is the feature processing stage, which runs from the HPF Module to the second ERM module. The second is the feature selection stage or our ELLFRM module. In the first stage, the number of feature maps is kept unchanged while the image size is slightly changed, which will be explained below. As inspired by [9] [40] and [21], keeping the number of feature maps unchanged during the image processing part allows the convolutional kernels to learn the edge patterns accurately. To ensure the 3×3 kernels can extract as much information as possible, we endeavour to avoid any down-sampling to the images during the feature processing stage.

In the second stage, i.e. the ELLFRM module, the number of feature maps and the image sizes are adjusted for better feature selection. We first increase the feature maps to have more channels and local receptive fields for picking up different features. Then we decrease the feature maps to keep the most effective features learnt from the previous stage for image classification.

3.1.1. Feature processing stage

In the feature processing stage, we deploy an HPF module in the beginning. This module consists of one convolutional layer and a Downsampling Module (DM). The convolutional layer is in accor-dance with the works [16,22], which has 30 high-pass filters, each with a size of 5×5. This layer has been proven to help the network converge in the early stage of the training. As "avoid pooling in the first layer" can provide better performance [5,41], we set the padding (p) to 0 and slide (s) to 1. This architecture utilizes multiple DM modules, and the reasons are mainly threefold, i.e. increasing the convergence speed, downsampling the input images and adjusting the size of the feature maps. The modules usually contain a convolutional layer and a batch normalization layer. In the DM of the HPF module, the convolutional layer has 32 channels with a kernel size of 3×3, and we set $p = s = 1$, which is kept the same as in [40]. A batch normalization layer is used for improving the convergence speed and stability of the CNN. The ReLU function is used for selecting the initial features. In our architecture, only 3×3 kernels are used in the convolutional layers, which can reduce the training parameters whilst providing better performance, according to [42].

Then, we use an ERM, i.e. $Y_{ERM}(X) = \widehat{y}_2 \widehat{y}_3 y_4$, to extract features as it provides more receptive fields compared to Eq. (3). The $H_{Res2}(X)$ functions share the same structure as shown in Fig. 1 (c). After that, two DM modules are used, which will slightly decrease the image size while ensuring the magnitude of features is normalized again. After that, two ERM modules are employed to extract the complex features. Those settings are inspired by [40]. However, the optimal number of such modules in our architecture is experimentally validated, as detailed in Section 4.3.2. After the feature processing part, a Maxpooling layer is used for feature selection and dimension reduction with $k = 3$, $s = 2$, and $p = 1$.

3.1.2. Feature selection with the proposed enhanced low-level feature representation module

In the feature selection stage, we propose to use our Enhanced Low-level Feature Representation Module (ELLFRM) to effectively select the features. In this module, firstly, we force the output images to go through two DM modules to shrink the image, for achieving a good balance between the performance and GPU requirement. These DMs are used for extracting features and selecting the effective ones, which have the same settings, with the same number of input and output channels, i.e. 32 3×3 kernels with $s = 1$, $p = 1$. Then, an average pooling layer is implemented to average each patch of the feature map. Unlike the MaxPooling layer which removes some features, it is used to shrink the size of the feature maps. We set $s = 2$, $p = 0$, which halves the image size and reduces the parameters.

We then expand the feature maps for further feature selection by using the two proposed Multi-Stage Residual Modules (MSRM). These MSRMs are designed to greatly increase the receptive fields so that the complex features in different shapes can be effectively captured, meanwhile, these MSRMs should reuse their features during the processing with the introduced residual mechanism. Lastly, these MSRMs should be deep enough to fit different sizes of images. Hence, our MSRMs have the same shape as shown in Fig. 3. They can be written as in Eq. (14), where Y_{MSRM} is defined in Eq. (15) and X1, X2 are created by splitting X along the dimension of channels. Note Eq. (14)shows the case of two stages, yet it can be easily extended to multiple stages.

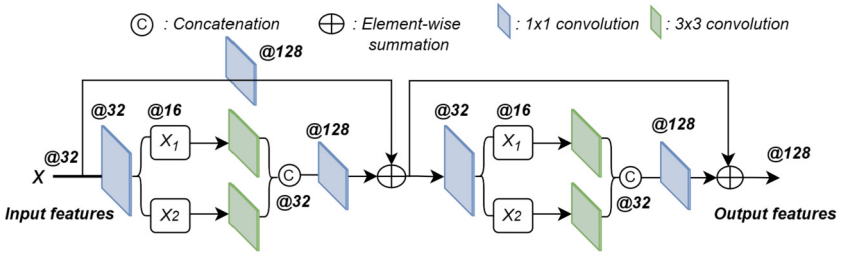

Fig. 3. Proposed Multi-Step Residual Module (MSRM). The numbers after symbol @ represent the number of the output channels in our first MSRM module.

$$H_{MSRM}(X) = W_1(X) + Y_{MSRM}(X) \; + Y_{MSRM}(W_1(X) + Y_{MSRM}(X)) \tag{14}$$

$$Y_{MSRM}(X) = W_1(W_3(W_1(X_1)) \frown W_3(W_1(X_2))) \tag{15}$$

As the CNN goes deeper, it will have smaller receptive fields and less specific features. These features are more complex than the ones in the shallower layers, hence more channels are needed to capture these features. And hence the MSRM is used for feature selection. The first and the second MSRM modules will expand the feature maps to 128 and 512 dimensions, respectively. It is experimentally validated to provide improved performance while keeping the number of the parameters low compared to the basic residual block in Eq. (3) and the Res2Block in Eq. (6).

To enhance the capability of capturing those complex features, we introduce the Self-Attention Modules (SAM) into our model. According to the experiments in [43], the self-attention module will help the CNN to focus more on the embedding areas than the CNN without it. In our architecture, the self-attention modules are used in the very deep layers, where the features are not human-understandable ones. Hence we could not show the difference through visual explanations. A SAM module is an optimized BoTBlock with N set to 4 and the activation is set to ReLU, i.e. $Y_{SAM}(X) = y_1 \frown y_2 \frown y_3 \frown y_4$.

We also found that these SAMs can further improve the per-formance of CNN because of their global self-attention mechanism. The first SAM takes the input features with a size of 512 and will expand them to 1024, while the second SAM will then shrink them back to 512. This move is inspired by the CNN architecture in [44], where they have also increased the number of the output channels and decreased it before classification for an effective feature selection. We have experimentally validated that two SAMs will provide the best performance, as detailed in Section 4.3.4. Next, these feature maps will be processed by a Global Max Pooling layer and output a 512×1 feature. Before going through the last fully connected layer, the feature will be processed by the dropout method to prevent overfitting with the possibility set to 0.5 [45]. Ultimately, the output feature will be used for classification.

3.2. Implementation details

An ERM has two parameters, i.e. the Basewidth and the scale. The Basewidth is set to 36 and the scale is set to 4, which are used to make sure we get 4 mappings in Eq. (6). The output of this layer keeps the same dimension as the input, and hence $p = s = 1$. For SAMs, we set the N to 4 and the activation function is ReLU as recommended in [31]. The Adamax optimizer is employed with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 0.001$ [46]. The initial learning rate is set to 0.001. L2 regularization is used to prevent overfitting. The weight decay is set to 0.1. The classic cross-entropy loss function is employed in our method. The network is trained for 500 epochs, and the learning rate is divided by 10 whenever the error plateaus. The network is implemented using Pytorch, 1.7.0, and the whole model requires about 12GB of memory. The experiments are carried out on a Tesla V100 Card, and it takes about 28 hours to train on the BOSSbase 1.01 dataset [23] and 156 hours to train on the ALASKA#2 dataset [5].

4. Feature analysis and ablation study

In this section, we will first introduce the datasets that we used in our experiments and the evaluation metrics. Next, we provide a detailed analysis of the feature maps and the numerical results in subsection 4.2.1. Then, we will explain why the architecture in Section 3 is selected by an ablation study in subsection 4.3.

4.1. Dataset introduction

We have evaluated our proposed method on two datasets, BOSSBase 1.01 [23] and ALASKA#2 [47][5]. The BOSSBase 1.01 dataset contains 10,000 grayscale images with a ".pgm" format. Ac-cording to [48], these images were initially taken by seven cameras in the RAW format, and transformed into 8-bit grayscale images, before being cropped into the size of 512×512. We used the MAT-LAB function *imresize*() to resize the images to 256 × 256 as in [19]. In the following experiments on the BOSSbase dataset, if not specified, 6,000 images were randomly selected for training, 1,000 images for validating and the rest 3,000 for testing. Note that this dataset is **different** from the one used in the SiaStegNet [22], as the images are pre-processed in different ways. According to Table 2 in [40], different pre-processing ways have a large impact on the detection results. Specifically, they first cropped the images into squares based on the shorter side and then resized them to 256×256 using the "imresize" function with the bilinear interpolation algorithm in Matlab R2017a. Therefore, one should not compare the results of this paper and the results reported in [22] directly.

The ALASKA#2 dataset is a new dataset proposed in 2020, which was created to provide "a large and heterogeneous dataset" for steganalysis [47]. It contains 80,005 grayscale images from more than 40 cameras with different sensors, and the images were processed in a highly heterogeneous way. In our experiments, the images have a ".tif" format and each with a size of 256 × 256. We will keep the same setting as in the SiaStegNet [22], i.e. the ratios of the training set to the validation set and the testing set are 6:1:3. There exists no overlap among them. For a fair comparison, we used the *Image* function from the Python Image Library[1] to read those images in the input of all detectors. One of the cover images, its stego image (0.4bpp), and their difference are shown in Fig. 4.

We employ two different evaluation metrics in this paper. The first is the detection accuracy as a percentage, and the second is the area under the curve (AUC) within [0,1].

Fig. 4. Example of a cover image (left), its stego image (middle), and their difference image (right).

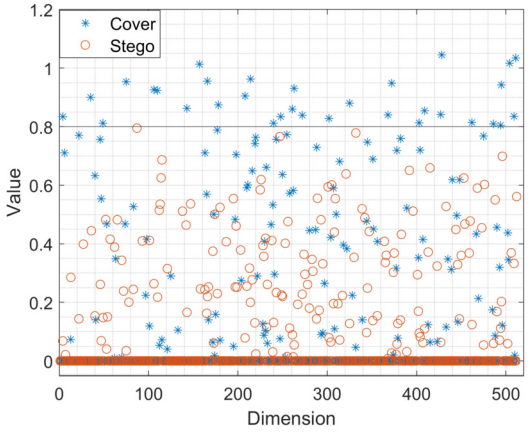

Fig. 5. The output features of the ERANet for the cover and stego image.

[1] https://pillow.readthedocs.io/en/stable/reference/Image.html.

4.2. Visualization of the features

4.2.1. The visualization of the output features

We show the output features in the Global Max Pooling layer in Fig. 5. For the cover image, 59.76% of features are effective or non-zero while for the stego image, the number is 64.45%. For a better understanding, we draw an auxiliary horizontal line at 0.8, which shows that nearly all the stego features (in circles) are below 0.8 while there are 32 cover features (in Asterisks) above 0.8. We believe the ERANet keeps the effective features above zero while keeping the silent features as zeros, and thus differentiates the cover and the stego images.

4.2.2. The comparisons between the cover and stego features

We first show the feature maps of the cover and stego images in Fig. 6, and they are extracted from the last DM module of the ERANet. The rectangle areas indicate highlighted parts of the differences. The StegoChannels 1 to 4 show more "bubbles" or black dots in the top and bottom areas, which indicate the embedding areas. However, these patterns are not seen in the CoverChannels, and we believe those patterns can help CNN to differentiate them. Note that although the majority part of the embedding area is not shown in the feature maps, they are not what this CNN is trained for. To highlight the difference, or to find the suspicious area, however, is the real purpose.

4.3. Ablation analysis

In this subsection, we provide an ablation study to explain how the parameters and the architecture are determined and also show how they may affect the results.

4.3.1. Trainable SRM kernels

We first investigate whether the fixed kernels in the HPF module can help to improve the detection accuracy. As explained before, the introduction of the high-pass filters in our CNN architecture was to help the network to converge fast. The ERANet will converge much slower if this layer is removed. In our experiment, these fixed kernels can help to converge fast, but with a slight performance loss. The fixed-SRM-kernel version of the ERANet has an accuracy of 82.07% in detecting the HILL at 0.4 bpp, i.e. 0.42% less than the trainable one.

4.3.2. The analysis of the use of the ERM

Firstly, we investigate the number of layers in the feature pro-cessing part, and the results are shown in the upper part of Table 1. In the first model, ERANet#A, the last ERM module was removed. This move has slightly reduced the need for GPU memory from 12GB to 11GB but the accuracy is degraded by about 1.5%, thus not a good solution as the current one used. Similarly, in the second mode, ERANet #B, we added one more ERM before the ELLFRM, with the same setting as its previous module. At this time, the requested memory is increased by 1.5GB, and hence the model requires about 13.5GB to run. However, the accuracy is not increased further. This suggests that the current setting is the best to achieve good classification accuracy and modest GPU memory.

4.3.3. The analysis of the use of the MSRM

Next, we analyse the influence of changing the MSRMs in the feature selection part. As mentioned before, we use the proposed MSRM in the feature selection part because the MSRM can retain more effective features during the previous feature processing part compared to the ERM module.

Experiments including removing and increasing one MSRM, changing one MSRM into ERM, and changing all ERMs to MSRMs were carried out, and the results are compared in Table 1. In model ERANet #C, we change the first MSRM to the ERM and keep the rest untouched. This move causes a 1.25% performance loss. In the model ERANet #D, the second MSRM is changed in the same way. However, the size of the output image was halved in ERANet, hence we need to append a DM module to enlarge the feature maps to 512 and to reduce the image size. In this way, we can keep the rest layers unchanged. However, this move reduces the accuracy by about 1%. The situation is similar to the cases when we remove the first MSRM in ERANet #E. In ERANet #X, where all the ERMs are changed to MSRMs, the overfitting happens and the test accuracy has decreased by 20%. This step has proven the rationality of the ERM in the feature processing stage and indicated the rationality of the current settings.

4.3.4. Effect of the SAM modules

We investigate the introduction of the SAMs and show how these modules may affect the accuracy and the trainable parameters.

We compare four models in the current architecture, namely ERANet #F, ERANet #G, ERANet #H and ERANet. The first model here, ERANet #F, does not use any SAM modules, where the first SAM in Fig. 2 is replaced by ERM and the second SAM is replaced by the simple DM module to maintain the same number of layers and the same dimension of the feature maps. In the second model, ERANet #G, only the first SAM is replaced by ERM and the remaining layers are unchanged. Using the SAM in the last convolutional layer is also the way suggested in [31]. In the ERANet, we have only two SAMs, and we cannot use three of them without changing the current architecture because of the limited GPU memory. In addition, to verify the effectiveness of introducing the relative-position-information (RPI) into the SAMs, we removed the R_h and R_w in Eq. (12) from the ERANet and hence y_i in Eq. (12) becomes Eq. (9). We named this model ERANet #H.

The comparing results are shown in Table 2, in which the model with two SAMs using relative-position-information achieves the highest accuracy and the least number of parameters. Removing both SAMs in the ERANet results in the largest performance drop and removing the relative-position-information (RPI) in these

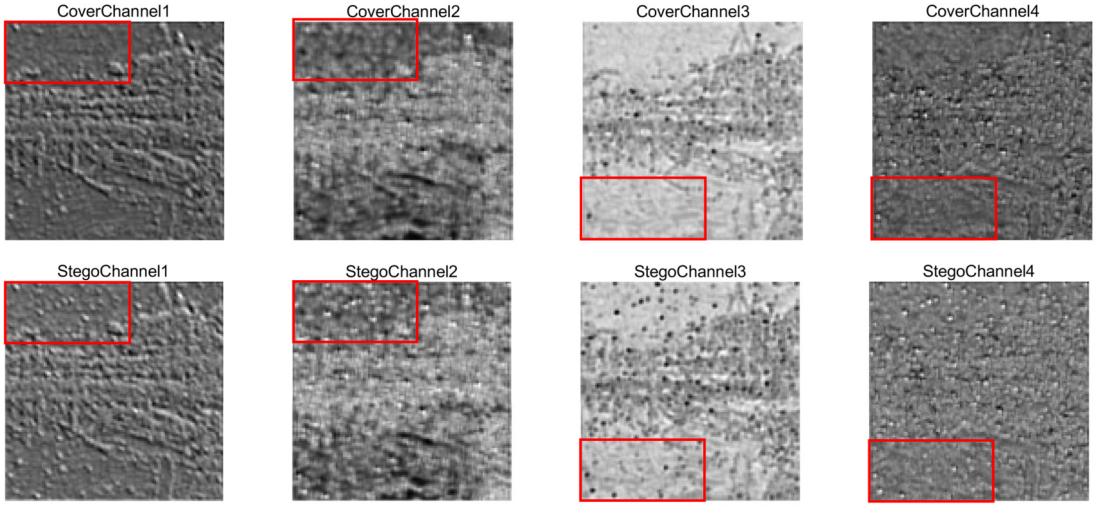

Fig. 6. Comparison of the feature maps between the cover image and stego image in the **last** DM module of the ERANet.

Table 1
The analysis of the use of ERM and MSRM.

Model	Action	Memory Requirement	Accuracy (%)
ERANet#A	remove the last ERM	11GB	81.08
ERANet#B	increase one EMR before the feature selection part	13GB	82.19
ERANet#C	change the **first** MSRM to ERM	12GB	81.25
ERANet#D	change both MSRMs to ERMs	12GB	81.53
ERANet#E	change the **last** MSRM to ERM	12GB	81.63
ERANet#X	change all ERMs to MSRMs	11GB	62.09
ERANet	-	12GB	**82.49**

Table 2
The analysis of the SAM modules.

Model	Action	Number of Param	Accuracy (%)
ERANet#F	remove both SAMs	2.71M	80.92
ERANet#G	change the first SAM	2.97M	81.48
ERANet#H	remove RPI	2.35M	81.20
ERANet	-	2.35M	**82.49**

Table 3
The influence of the batch size in the proposed model.

Batch size	24	32	40	48
Accuracy (%)	81.87	**82.49**	81.70	81.27
Memory Requirement (GB)	9	12	15	18

Table 4
The effect of the ELLFRM on other CNN models against different steganographic algorithms at 0.4 bpp on the sizing 512×512 BOSSbase dataset.

Model	WOW	SUNI	HILL
Xu-Net	80.44	78.93	79.03
Xu-ELLRM	**82.10**	**79.33**	**80.19**
SID	79.08	74.85	73.89
SID-ELLRM	**89.11**	**83.16**	**83.82**
SiaStegNet	89.56	87.30	84.53

SAMs decreases the accuracy by about 1.2%. Although in ERANet #H, using only one SAM provides a comparable performance, the model is not as effective as the proposed ERANet with two SAMs.

4.3.5. Batch size

We have also investigated the influences of the batch size, and the results are shown in Table 3. As seen, the performance of the network is not significantly influenced by the batch size, though the best batch size is found at 32. Increasing the batch size from 32 can no longer provide any improvement but will require more memory.

4.3.6. The effect of applying the proposed ELLFRM module in other CNNs To investigate the versatility of the proposed ELLFRM, we add our ELLFRM Module to two classic CNN models, i.e. the Xu-Net [13] and SID [40]. We enhance the corresponding architectures by changing the layers whose size of the input features is 128×128 and the deeper layers to our ELLFRM. In this way, we created Xu-ELLFRM and SID-ELLFRM.

For evaluation, we test on the BOSSbase 1.01 dataset for these CNNs, yet the images are not resized, hence each image has a size of 512×512. The ratios of the training set to the validation set and the testing set are 4:1:5. The hyperparameters for these models are unchanged except that the optimizer for SID-ELLFRM was changed to Adamax, otherwise it can not converge. All the models are trained for 500 epochs, and the payload is set to 0.4 bpp. The results are shown in Table 4.

As seen in Table 4, our ELLFRM can improve both models without changing their pre-processing layers and the hyperparameters. For the Xu-Net, our modified model provides an improvement of up to 1.5%. However, for SID, the improvement is up to 9%. Note that SID-ELLFRM has achieved SOTA performance in detecting WOW and HILL.

These results can be explained twofold. Firstly, the single high-pass filter in the Xu-Net fails to provide sufficient complex features for its deeper layers yet those features are captured by the 30 high-pass filters from the SID. Secondly, the SID is deeper than Xu-Net, hence providing more receptive fields and more effective features for image classification.

Table 5
Performance comparisons (Accuracy) of different detectors for four payloads in bpp and different steganographic methods on the BOSSbase 1.01 dataset (in %).

	WOW(bpp)				SUNI(bpp)				HILL(bpp)			
CNN Scheme	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
SRNet	71.57	80.58	85.84	89.18	68.55	77.48	85.06	89.08	65.16	74.01	79.64	84.07
SiaStegNet	**72.93**	**80.72**	85.54	88.29	**68.80**	77.24	83.72	87.93	64.30	72.49	78.53	81.55
ERANet	70.17	80.02	85.56	88.80	66.00	76.03	82.89	87.18	63.34	71.16	78.43	82.49

Table 6
AUC comparisons of different detectors for four payloads in bpp and different steganographic methods on the BOSSbase 1.01 dataset (in %).

	WOW(bpp)				SUNI(bpp)				HILL(bpp)			
CNN Scheme	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
SRNet	80.38	89.92	94.20	96.75	76.48	87.81	94.09	96.98	71.84	82.37	88.73	93.18
SiaStegNet	**80.93**	**90.91**	94.11	96.38	**77.35**	87.33	93.29	95.45	71.77	81.86	88.10	91.51
ERANet	79.85	90.36	**94.91**	**96.89**	74.85	86.73	93.02	95.94	71.45	81.67	88.44	92.79

Table 7
Performance comparisons (Accuracy) of different detectors for four payloads in bpp and different steganographic methods on the ALASKA#2 dataset (in %).

	WOW(bpp)				SUNI(bpp)				HILL(bpp)			
CNN Scheme	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
SRNet	56.80	61.63	65.70	67.83	53.47	56.60	59.18	62.60	57.58	63.49	67.78	71.10
SiaStegNet	57.72	63.94	67.88	70.55	55.08	60.93	64.41	67.55	58.26	63.20	67.43	71.01
ERANet	**60.79**	**65.95**	**69.99**	**72.71**	**59.11**	**65.21**	**69.86**	**72.48**	**61.81**	**68.17**	**72.14**	**74.95**

Table 8
AUC comparisons of different detectors and different steganographic methods on the ALASKA#2 dataset (in %).

	WOW(bpp)				SUNI(bpp)				HILL(bpp)			
CNN Scheme	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
SRNet	58.58	64.14	67.61	70.36	54.76	58.79	61.50	64.40	59.64	66.50	72.87	75.36
SiaStegNet	63.30	72.95	78.09	81.48	59.19	68.27	73.57	78.12	64.08	71.54	77.18	81.69
ERANet	**67.89**	**75.74**	**80.88**	**84.10**	**65.74**	**74.70**	**80.70**	**83.87**	**69.63**	**78.50**	**83.25**	**86.57**

5. Comparing with the state-of-the-art models

We compare our work with the SRNet [19] and SiaStegNet [22], using the aforementioned two datasets, i.e. the BOSSBase and ALASKA datasets. The SRNet is selected for its superior performance on the BOSSBase dataset while the SiaStegNet is selected for the ALASKA dataset. Note all CNNs were trained on the datasets from scratch at 0.4 bpp, and we selected the best model during the validation for testing. For other payloads, the CNNs were all trained with the curriculum training strategy as suggested in [19].

5.1. Results on the BOSSbase dataset

The first group of the data will show the detection performance on the BOSSbase 1.01. However, according to [47], "neural networks and deep learning require a big enough dataset". The BOSSbase 1.01 itself seems not to be large enough even with 60% of the data used for training. To see how different networks perform with these data, we did the experiments and show the results from different detectors in Table 5 and Table 6.

In Table 5, we can see that SRNet shows the best performance for most cases in this dataset, and the SiaStegNet surpasses it when the payload is 0.1 for WOW and SUNI. Our model only shows comparable performance to them and it suffers the most from insufficient training data.

The situation is similar for the AUC results in Table 6. The results were calculated by using the "sklearn" tool offline [49]. However, ERANet shows the best results when the payloads are 0.3 and 0.4 bpps for WOW.

5.2. Results on the ALASKA dataset

For the ALASKA#2 dataset, we also show how the detectors perform when they are faced with a more complicated scenario and report the results in Table 7 and Table 8. Note that we have to slightly lower the learning rate of SiaStegNet from 1e-3 to 8e-4 when trained for the WOW algorithm, due to a different version of Pytorch, otherwise the network would not converge in our experiments. When trained for the other two algorithms, their learning rates were unchanged, i.e. 1e-3.

From Table 7, one can easily find that the least detectable alWgorithm, HILL, has become the most detectable one in this dataset. All detectors can achieve an accuracy higher than 70% in detecting HILL at 0.4 bpp, and the accuracy of our proposed model is about 4% higher than the other two CNNs.

The steganographic algorithm, SUNI, has become the most undetectable where both the SRNet and SiaStegNet achieve an accuracy of less than 70% when the payload is 0.4 bpp. However, the ERANet is about 5% better than SiaStegNet and about 10% better than SRNet. In the case of WOW, the situation is similar to the SUNI, where the proposed model is about 2% better than the SiaStegNet and about 5% better than the SRNet when the payload is 0.4 bpp.

In Table 7, the results at the low payloads are also worth noting, especially the cases of 0.1 bpp. First of all, all three methods can only provide an accuracy of about 60%. For HILL and WOW, ERANet is about 4% better than the other two CNNs. For SUNI, it is about 4% better than the SiaStegNet and 6% better than the SRNet. In short, can still significantly outperform the other two state-of-the-art CNNs.

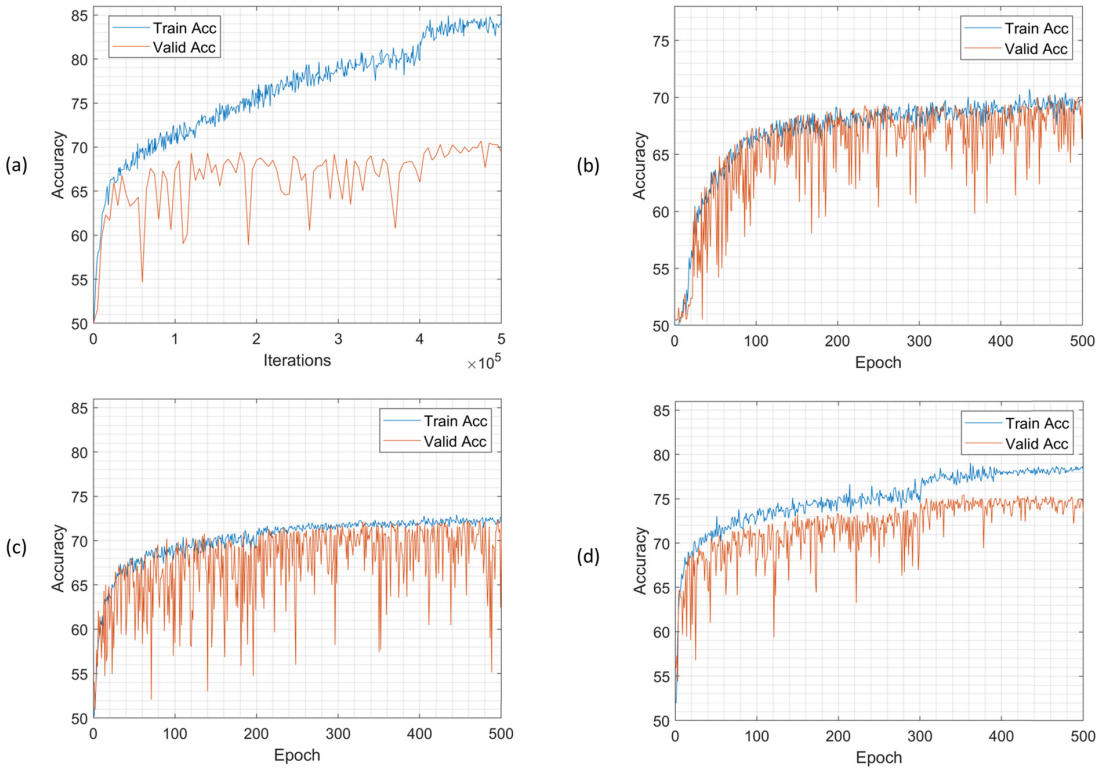

Fig. 7. The training processes of different CNNs for steganography HILL at 0.4 bpp on the ALASKA#2 dataset; (a) SRNet (b) SiaStegNet (c) ERANet without SAMs (d) ERANet.

Table 9
Performance comparisons (Accuracy) of different detectors and different stegano-graphic methods on the sizing 128 × 128 ALASKA#2 dataset (in %) at 0.4bpp.

Model	WOW	SUNI	HILL
SRNet	59.20	56.06	84.42
SiaStegNet	67.31	65.63	77.35
ERANet	**69.74**	**67.63**	78.31

For further comparisons between the SiaStegNet and the proposed ERANet, we show the AUC results in Table 8. All the AUC results confirm again that our ERANet can provide overall much-improved results. To explain the results, we show the training processes of different CNNs for HILL at 0.4 bpp in Fig. 7. Note that the curves in Fig. 7 are unsmoothed. From Fig. 7 (a), we can see that there is a large gap between the training and validating accuracy, which indicates potential overfitting. In Fig. 7 (c), the model has no SAMs. We see that even though Fig. 7 (c) and Fig. 7 (d) are using the same training strategy, the validation accuracy does not improve when the learning rate is divided by 10 after 300 epochs. Also, the validation curve in Fig. 7 (c) shows large perturbations after the 400th epoch. However, after introducing the self-attention mechanism, the validation curve in Fig. 7 (d) becomes more stable along with an improved validation accuracy.

The performance of the ERANet on the 128 × 128 dataset is also investigated in Table 9. Our ERANet achieves the best results for WOW and SUNI and the second-best for HILL at the payload 0.4 bpp.

5.3. Transfer learning results

One advantage of the CNN-based steganalyzers is transferability, which means the trained network can be used for steganalysis even when the cover or stego source mismatch with each other.

Table 10
Transferability (Accuracy) of SRNet for different steganographic methods on the two datasets at 0.4 bpp (in %).

BOSSbase 1.01 Train / Test	WOW	SUNI	HILL
WOW	89.18	82.85	66.63
SUNI	90.74	89.08	74.16
HILL	83.93	79.72	84.07
ALASKA#2 **Train / Test**	**WOW**	**SUNI**	**HILL**
WOW	67.83	58.34	60.02
SUNI	60.68	62.60	59.08
HILL	54.85	54.85	71.10

Those scenarios are discussed below to further validate the superiority of the proposed model.

5.3.1. Mismatched stego sources

Mismatched stego sources mean the network was trained on one steganographic method and then tested on a different one at the same payload [19]. We perform the experiments on both the BOSSbase 1.01 and ALASKA#2 datasets, and the results are shown in Tables 10 to 12.

From the upper parts in these tables, the CNNs trained on the least detectable algorithm (HILL) transfer the best, which is consistent with the results in SRNet [19]. We also found that the CNN that performed the best in this dataset usually transfers the best. In the bottom parts, although the SiaStegNet achieved the second-best in detecting WOW, it transfers the best. In other cases, the proposed ERANet transfers the best.

Table 11
Transferability (Accuracy) of SiaStegNet for different steganographic methods on the two datasets at 0.4 bpp (in %).

BOSSbase 1.01 Train / Test	WOW	SUNI	HILL
WOW	88.29	81.60	64.51
SUNI	88.39	87.93	71.98
HILL	83.52	78.63	81.55
ALASKA#2 **Train / Test**	**WOW**	**SUNI**	**HILL**
WOW	70.55	63.83	64.11
SUNI	64.04	67.55	63.62
HILL	62.28	57.19	71.01

Table 12
Transferability (Accuracy) of ERANet for different steganographic methods on the two datasets at 0.4 bpp (in %).

BOSSbase 1.01 Train / Test	WOW	SUNI	HILL
WOW	88.80	74.92	61.97
SUNI	85.96	87.18	67.30
HILL	83.37	76.64	82.49
ALASKA#2 **Train / Test**	**WOW**	**SUNI**	**HILL**
WOW	72.71	62.26	63.26
SUNI	67.53	72.48	65.53
HILL	66.99	59.63	74.95

Table 13
Performance comparisons (Accuracy) of transferability of different detectors for different steganographic methods at 0.4 bpp (in %).

Train on ALASKA#2 Test on BOSSbase 1.01	WOW	SUNI	HILL
SRNet SiaStegNet	75.61	62.95	76.38
	78.23	73.78	73.78
ERANet without SAMs ERANet	**84.56**	77.87	76.99
	82.60	**81.63**	**81.38**
Train on BOSSbase 1.01 Test on ALASKA#2	WOW	SUNI	HILL
SRNet SiaStegNet	58.48	57.40	**62.85**
	57.34	58.14	58.26
ERANet without SAMs ERANet	58.73	56.34	62.15
	59.20	**58.84**	62.53

5.3.2. Mismatched cover sources

Mismatched cover sources mean the network was trained on one dataset but tested on another dataset [50]. In the following experiments, we will compare in Table 13 the transferability of different CNNs in terms of different datasets. In the upper part of Table 13, it shows the case when training on the ALASKA#2 dataset but testing on the BOSSbase dataset, where the training dataset has 48,000 training samples and the testing dataset has 3,000 samples. The bottom part of the table shows the results when training on the BOSSbase dataset (with 6,000 samples) and testing on the other (with 24,000 samples). These experiments simulate those situations, i.e. training on a large dataset and testing on a small one versus training on a small dataset and testing on a larger one.

From the upper part of Table 13, we see that under most circumstances, the CNNs produce good results in the unseen dataset. To be specific, ERANet provides the best results when detecting SUNI and HILL, about 4% better than the ERANet with the SAMs removed. Although removing the SAMs would achieve the best result in detecting WOW, it has a performance close to SRNet in detecting HILL.

From the bottom part, we can see that the CNNs are showing a result better than random guessing even with 6,000 training samples, yet they have a much larger testing set. In this table, the SRNet shows the best result in detecting HILL, which is in accordance with its performance in the BOSSbase dataset. However, the proposed ERANet shows the best results in detecting WOW and SUNI, also a comparable performance to the SRNet in detecting HILL, again, the best in the group.

6. Conclusions

In this paper, to tackle the much more complex scenarios of realistic images in the ALASKA#2 dataset, we proposed a new enhanced residual network with the self-attention capability for spatial image steganalysis.

We employed a sophisticated way to extract more effective features in the images in the form of residuals, which is theoretically and experimentally validated that this provides improved performance in detecting the stego noise. Moreover, our proposed Enhanced Low-Level Feature Representation Module can also help the classic models to achieve better results without modifying their pre-processing layers and the hyperparameters. Extensive experiments on both the BOSSbase 1.01 and ALASKA#2 datasets at various sizes have fully validated the effectiveness of the proposed model.

We aim to provide a new way of exploring the residual information to obtain the extremely weak stego signal and this CNN architecture can be used to train a Generative Adversarial Network and for more effective steganography. Further improvement may be implemented in the future by applying new techniques such as multiscale noise-robust feature optimization [51], attention network feature enhancement [52] and feature refinement for edge computing e.g. using the MobileNet [53].

CRediT authorship contribution statement

Concept design (J Ren, S Marshall and H Zhao); coding and implementation (G Xie); data collection (G Xie, R Li and R Chen); results analysis (all); paper drafting (G Xie, J Ren); paper revision (all). All authors contribute to the final version of the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] X. Wu, C.-N. Yang, Partial reversible ambtc-based secret image sharing with steganography, Digit. Signal Process. 93 (2019) 22–33.
[2] K. Manjunath, G.K. Ramaiah, M. GiriPrasad, Backward movement oriented shark smell optimization-based audio steganography using encryption and compression strategies, Digit. Signal Process. 122 (2022) 103335.
[3] H. Ghasemzadeh, M.T. Khass, M.K. Arjmandi, Audio steganalysis based on reversed psychoacoustic model of human hearing, Digit. Signal Process. 51 (2016) 133–141.
[4] A. Nissar, A.H. Mir, Classification of steganalysis techniques: a study, Digit. Signal Process. 20 (6) (2010) 1758–1770.
[5] R. Cogranne, Q. Giboulot, P. Bas, Alaska# 2: challenging academic research on steganalysis with realistic images, in: International Workshop on Information Forensics and Security (WIFS), IEEE, 2020, pp. 1–5.

[6] S. Tan, B. Li, Stacked convolutional auto-encoders for steganalysis of digital im-ages, in: Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, 2014, pp. 1–4.
[7] T. Pevnỳ, T. Filler, P. Bas, Using high-dimensional image models to perform highly undetectable steganography, in: International Workshop on Information Hiding, Springer, 2010, pp. 161–177.
[8] J. Fridrich, J. Kodovsky, Rich models for steganalysis of digital images, IEEE Trans. Inf. Forensics Secur. 7 (3) (2012) 868–882.
[9] Y. Qian, J. Dong, W. Wang, T. Tan, Deep Learning for Steganalysis via Con-volutional Neural Networks, Media Watermarking, Security, and Forensics, vol. 9409, International Society for Optics and Photonics, 2015.
[10] T. Pevny, P. Bas, J. Fridrich, Steganalysis by subtractive pixel adjacency matrix, IEEE Trans. Inf. Forensics Secur. 5 (2) (2010) 215–224, https://doi.org/10.1109/TIFS.2010.2045842.
[11] B. Li, M. Wang, J. Huang, X. Li, A new cost function for spatial image steganog-raphy, in: 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 4206–4210.
[12] V. Holub, J. Fridrich, T. Denemark, Universal distortion function for steganogra-phy in an arbitrary domain, EURASIP J. Inf. Secur. 1 (2014) 1–13.
[13] G. Xu, H.-Z. Wu, Y.-Q. Shi, Structural design of convolutional neural networks for steganalysis, IEEE Signal Process. Lett. 23 (2016) 708–712.
[14] Y. Qian, J. Dong, W. Wang, T. Tan, Learning and transferring representations for image steganalysis using convolutional neural network, in: IEEE International Conference on Image Processing, 2016, pp. 2752–2756.
[15] G. Xu, H.-Z. Wu, Y.Q. Shi, Ensemble of cnns for steganalysis: an empirical study, in: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, 2016, pp. 103–107.
[16] J. Ye, J. Ni, Y. Yi, Deep learning hierarchical representations for image steganalysis, IEEE Trans. Inf. Forensics Secur. 12 (11) (2017) 2545–2557, https://doi.org/10.1109/TIFS.2017.2710946.
[17] J. Kodovsky, J. Fridrich, V. Holub, Ensemble classifiers for steganalysis of digital media, IEEE Trans. Inf. Forensics Secur. 7 (2) (2011) 432–444.
[18] M. Yedroudj, F. Comby, M. Chaumont, Yedroudjnet: an efficient cnn for spatial steganalysis, in: International Conference on Acoustics, Speech and Signal Processing, IEEE, 2018, pp. 2092–2096.
[19] M. Boroumand, M. Chen, J. Fridrich, Deep residual network for steganalysis of digital images, IEEE Trans. Inf. Forensics Secur. 14 (5) (2018) 1181–1193.
[20] B. Li, W. Wei, A. Ferreira, S. Tan, Rest-net: diverse activation modules and par-allel subnets-based cnn for spatial image steganalysis, IEEE Signal Process. Lett. 25 (5) (2018) 650–654.
[21] R. Zhang, F. Zhu, J. Liu, G. Liu, Depth-wise separable convolutions and multi-level pooling for an efficient spatial cnn-based steganalysis, IEEE Trans. Inf. Forensics Secur. 15 (2019) 1138–1150.
[22] W. You, H. Zhang, X. Zhao, A siamese cnn for image steganalysis, IEEE Trans. Inf. Forensics Secur. 16 (2020) 291–306.
[23] P. Bas, T. Filler, T. Pevnỳ, Break our steganographic system: the ins and outs of organizing boss, in: International Workshop on Information Hiding, Springer, 2011, pp. 59–70.
[24] Z. Lai, X. Zhu, J. Wu, Generative focused feedback residual networks for image steganalysis and hidden information reconstruction, Appl. Soft Comput. 129 (2022) 109550.
[25] T. Fu, L. Chen, Z. Fu, K. Yu, Y. Wang, Ccnet: Cnn model with channel attention and convolutional pooling mechanism for spatial image steganalysis, J. Vis. Commun. Image Represent. 88 (2022) 103633.
[26] S. Arivazhagan, E. Amrutha, W.S.L. Jebarani, Universal steganalysis of spatial content-independent and content-adaptive steganographic algorithms using normalized feature derived from empirical mode decomposed components, Signal Process. Image Commun. 101 (2022) 116567.
[27] W.M. Eid, S.S. Alotaibi, H.M. Alqahtani, S.Q. Saleh, Digital image steganalysis: current methodologies and future challenges, IEEE Access 10 (2022) 92321–92336.
[28] M. Bouzegza, A. Belatreche, A. Bouridane, M. Tounsi, A comprehensive review of video steganalysis, IET Image Process. 16 (13) (2022) 3407–3425.
[29] T. Muralidharan, A. Cohen, A. Cohen, N. Nissim, The infinite race between steganography and steganalysis in images, Signal Process. 201 (2022) 108711.
[30] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, P. Torr, Res2net: a new multi-scale backbone architecture, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2) (2021) 652–662.
[31] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, A. Vaswani, Bottleneck transformers for visual recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16519–16529.
[32] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[33] S. Wu, S. Zhong, Y. Liu, Deep residual learning for image steganalysis, Mul-timed. Tools Appl. 77 (9) (2018) 10437–10453.
[34] I. Bello, B. Zoph, A. Vaswani, J. Shlens, Q.V. Le, Attention augmented convolutional networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3286–3295.
[35] L. Itti, C. Koch, Computational modelling of visual attention, Nat. Rev. Neurosci. 2 (3) (2001) 194–203.
[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[37] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, J. Shlens, Stand-alone self-attention in vision models, Adv. Neural Inf. Process. Syst. 32 (2019).
[38] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position repre-sentations, in: NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2018.
[39] Y. Tay, M. Dehghani, D. Bahri, D. Metzler, Efficient transformers: a survey, preprint, arXiv:2009.06732, 2020.
[40] C.F. Tsang, J. Fridrich, Steganalyzing images of arbitrary size with cnns, J. Elec-tron. Imaging 2018 (7) (2018) 121–1.
[41] Y. Yousfi, J. Butora, E. Khvedchenya, J. Fridrich, Imagenet pre-trained cnns for jpeg steganalysis, in: Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS, 2020.
[42] R. Zhang, F. Zhu, J. Liu, G. Liu, Depth-wise separable convolutions and multi-level pooling for an efficient spatial cnn-based steganalysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 1138–1150.
[43] B. Singh, A. Sur, P. Mitra, Multi-contextual design of convolutional neural network for steganalysis, preprint, arXiv:2106.10430, 2021.
[44] J. Yang, B. Lu, L. Xiao, X. Kang, Y.-Q. Shi, Reinforcement learning aided network architecture generation for jpeg image steganalysis, in: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security, 2020, pp. 23–32.
[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[46] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, preprint, arXiv: 1412.6980, 2014.
[47] R. Cogranne, Q. Giboulot, P. Bas, The alaska steganalysis challenge: a first step towards steganalysis, in: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH& MMSec'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 125–137.
[48] G. Xu, Deep convolutional neural network to detect juniward, in: Proceed-ings of the 5th ACM Workshop on Information Hiding and Multimedia Security, 2017, pp. 67–73.
[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[50] J. Kodovskỳ, J. Fridrich, Steganalysis in High Dimensions: Fusing Classifiers Built on Random Subspaces, Media Watermarking, Security, and Forensics III, vol. 7880, International Society for Optics and Photonics, 2011.
[51] P. Ma, J. Ren, G. Sun, H. Zhao, X. Jia, Y. Yan, J. Zabalza, Multiscale superpixelwise prophet model for noise-robust feature extraction in hyperspectral images, IEEE Trans. Geosci. Remote Sens. 61 (2023) 1–12, https://doi.org/10.1109/TGRS.2023.3260634.
[52] Y. Li, J. Ren, Y. Yan, Q. Liu, P. Ma, A. Petrovski, H. Sun, Cbanet: an end-to-end cross band 2-d attention network for hyperspectral change detection in remote sensing, IEEE Trans. Geosci. Remote Sens. (2023), in press.
[53] R. Chen, H. Huang, Y. Yu, J. Ren, P. Wang, H. Zhao, X. Lu, Rapid detection of multi-qr codes based on multistage stepwise discrimination and a compressed mobilenet, IEEE Int. Things J. (2023), in press.