Distribution-preserving-based Automatic Data Augmentation for Deep Image Steganalysis

Jiansong Zhang, Kejiang Chen, Chuan Qin, Weiming Zhang, Nenghai Yu

***Abstract*—In recent years, deep learning-based steganalyzers far outperformed handcrafted feature-based steganalyzers. However, a large amount of data is needed to train deep learning networks. For steganalysis tasks, the steganographic traces are subtle and the steganographic signals are difficult to be captured when the number of cover/stego pairs in the training set is insufficient. Data augmentation has been proved to be effective in improving accuracy and generalization for deep learning models. Yet not all data augmentation methods are universal for all tasks. When performing data augmentation, we argue that data distribution under the target tasks should be maintained. Since the steganalysis task is more concerned with the high-frequency signals of the images, if the high-frequency signals are unchanged, the data distribution from the perspective of steganalysis will remain largely unchanged. Based on this principle, we designed a neural network called cover augmentation network, which enriches the dataset by intelligently adding noise to the original cover to generate the augmented cover. Further, we designed a whole process of data augmentation based on the cover augmentation network. Experimental results show that the proposed data augmentation method can effectively improve the performance of steganalysis networks, and the advantage is significant at low payloads.**

***Index Terms*—image, steganography, steganalysis, data augmentation.**

I. Introduction

Steganography is a technique used to create covert communication channels that hide secret information in multimedia, such as text, image, 3D meshes, etc. In the past decades, digital image steganography [1]–[3] has been well developed. Nowadays, the most popular and effective steganographic methods are based on the minimization distortion model. In the framework of the minimization distortion model, steganography is divided into two tasks: 1) defining modification costs of modifying the elements of the cover using heuristic-based approaches or neural network-based approaches; 2) designing practical embedding methods that minimize the total modification cost defined before. Since syndrome-trellis codes (STCs) [4] have a performance close to the theoretical bound in the second task, now the researches of steganography mainly focus on the design cost functions, such as WOW [5], UNIWARD [6], HILL [7], MiPOD [8], UERD [9], ASDL-GAN [10], UT-GAN [11], SPAR-RL [12], MCTSteg [13], etc.

Steganalysis is a defense and analysis technique for steganography. Image steganalysis plays an important role in many information security systems. The current researches on image steganalysis mainly focus on secret message detection, so steganalysis is usually simplified to a binary classification problem. Nowadays, various handcrafted feature-based methods have been proposed for steganalysis tasks. Like other image classification tasks, traditional handcrafted feature-based steganalysis [14]–[19] consists of two tasks: 1) a high-dimensional feature extractor that can capture the subtle modifications made by steganography; 2) a binary classifier trained on the high-dimensional features. The most successful feature extractors in the spatial domain are the Spatial Rich Model (SRM) [20] and its variants [21], [22]. For the classification task, ensemble classifiers [23]–[25] are widely used.

In recent years, inspired by the success of convolutional neural networks (CNNs) in various fields [26]–[28], CNN-based steganalyzers are also proposed, such as XuNet [29], YeNet [30], Yedroudj-Net [31], SRNet [32], CovPool-Net [33], CALPA-Net [34], etc. Currently, many steganalysis networks outperform the handcrafted feature-based steganalyzers a lot. However, since steganographic modifications are subtle, a large amount of data is needed to capture the steganographic signals when training the steganalysis network. Especially at low steganographic payloads, it is difficult for the steganalysis networks to capture the steganographic signals so that the steganalysis networks fail to converge when the cover/stego pairs in the training set are scarce. In steganalysis, the so-called irreducible error region [35] probably requires many more images than those normally used today [36], [37]. In the case of [38], it even takes one million images for the training phase. Therefore, when the quantity of images in the original dataset is insufficient, it is necessary to expand the dataset.

Adding additional images is an intuitive way to expand a dataset. However, Yedroudj et al. [39] show that expanding the dataset can have a negative effect when expanding images from: 1) other cameras, 2) strongly dissimilar sources and unbalance proportions, 3) the same RAW images but with a different development or 4) re-development of the learning set. Thus, cautions have to be taken when increasing the database size if one desire to improve the classification accuracy of steganalysis.

We argue that the key to dataset expansion is to maintain the data distribution from the perspective of the target task. The steganalysis task is more concerned with the high-frequency

The authors are with CAS Key Laboratory of Electro-Magnetic Space Information, University of Science and Technology of China, Hefei 230026, China (e-mail: zjs0014@mail.ustc.edu.cn; chenkj@ustc.edu.cn; qc94@mail.ustc.edu.cn; zhangwm@ustc.edu.cn; ynh@ustc.edu.cn.)

Corresponding authors: Kejiang Chen and Weiming Zhang.

This work was supported in part by the Natural Science Foundation of China under Grant 62102386, 62002334 and 62072421, and by China Postdoctoral Science Foundation under Grant 2021M693091, and by Anhui Science Foundation of China under Grant 2008085QF296 .

signals of the image, so if the high-frequency signals of the expanded image are consistent with that of the original image, the invariance of the image distribution under the steganalysis task is ensured to a large extent. Based on this principle, we proposed a data augmentation method that intelligently adds noise to the original cover while maintaining data distribution under steganalysis tasks as much as possible. Experimental results show that our data augmentation method works well on several networks and outperforms the previous method [36]. At low payloads, our method even drives the network to converge more easily. Our contributions are as follows:

- We argue that two basic requirements should be met when conducting data augmentation: 1) ensuring that the distribution of the expanded dataset is consistent with that of the original dataset from the perspective of the target task, and 2) assuring that the expanded dataset differs from the original dataset in content to enrich the dataset.
- We propose that the key to maintaining the data distribution under the steganalysis task is to keep the high-frequency signals constant. Based on this principle, we design a cover augmentation network, which can intelligently add noise to the original cover to generate augmented cover.
- Experimental results show that the proposed data augmentation method can improve the performance of multiple CNN-based steganalyzers and outperform easy-low-complexity augmentations [40]–[43] and pixels-off [36].

The rest of this paper is organized as follows. In Section II, notations, the steganalysis networks used in this paper, and the previous data augmentation methods are introduced. In Section III, the proposed data augmentation framework for steganalysis is demonstrated. Section IV gives the experimental setup and experimental results. Finally, in Section V, we summarize our work and provide an outlook for future work.

II. Preliminaries

A. Notations

Throughout the paper, matrices and vectors are written in capital orthographic typeface and elements are written in lowercase letters. The original cover (of size of $n_1 \times n_2$) is denoted by $\mathrm{X} = (x_{ij})^{n_1 \times n_2}$. $\mathrm{X_{Aug}} = (x_{ij}^{\mathrm{Aug}})^{n_1 \times n_2}$ denotes the augmented cover. $\mathrm{Y} = (y_{ij})^{n_1 \times n_2}$ denotes the stego corresponding to the original cover. $\mathrm{Y_{Aug}} = (y_{ij}^{\mathrm{Aug}})^{n_1 \times n_2}$ denotes the stego corresponding to the augmented cover. $\mathrm{P} = (p_{ij})^{n_1 \times n_2}$ denotes the probability map and $\mathrm{N} = (n_{ij})^{n_1 \times n_2}$ denotes the noise map. x_{ij}, x_{ij}^{Aug}, y_{ij}, y_{ij}^{Aug} and n_{ij} are integers, x_{ij}, x_{ij}^{Aug}, y_{ij}, $y_{ij}^{\mathrm{Aug}} \in \{0, ..., 255\}$ and $n_{ij} \in \{-1, 0, 1\}$. p_{ij} is a real number and $p_{ij} \in [0, 0.5]$.

B. CNN networks for spatial steganalysis

Steganalysis can be seen as a binary classification problem, and nowadays CNN-based steganalysis has been well developed. Herein, we review three typical steganalysis networks: Yedroudj-NET (YedNet) [31], SRNet [32] and CovPool-Net (CovNet) [33], on which we evaluated the effectiveness of our method.

1) YedNet: YedNet [31] is a shallow neural network that can converge relatively quickly, and it has good performance in the spatial domain steganalysis task. The network consists of three modules: a pre-processing module, a convolutional module, and a fully connected module. The pre-processing module includes one convolutional layer whose weights are initialized with high-pass filter kernels derived from the SRM linear filters [20], and this module serves to amplify the steganographic signals. The convolutional module is composed of five convolutional blocks to extract features. As for the classification module, it consists of three fully connected layers and a softmax activation function to realize binary classification.

2) SRNet: SRNet is a widely used steganalysis network that minimizes the use of heuristics and external forcing elements [32]. There are three modules in the network: a pre-processing module, a feature extraction module, and a classification module. The pre-processing module in SRNet includes two convolutional blocks, where the parameters are not predetermined and learned during the training process. The feature extraction module consists of ten convolutional blocks and the classification module is composed of a fully connected layer, which plays the role of binary classification. The advantage of SRNet is that it does not need to artificially set heuristic parameters for the pre-processing module, and it has good performance in the spatial and JPEG domains, but its large number of parameters makes training time-consuming. At low payloads, the network has difficulty capturing the steganographic signals at first and therefore converges slowly. To speed up the training of SRNet at low payloads, we first trained a pre-trained model on the original training set at high payloads, and SRNet was trained based on this pre-trained model for lower payloads.

3) CovNet: Global Covariance Pooling Network, referred to as Covpool-Net (CovNet) [33], is a deep network that performs well in the spatial domain. And it takes much less time to train than SRNet. Similar to the above two networks, CovNet can be divided into three modules, a pre-processing module, a feature extraction module, and a classification module. Its pre-processing module is the same as YedNet. The feature extraction module consists of four groups of convolutional blocks. The first three convolutional blocks are all followed by average pooling layers, and the fourth convolutional block is followed by a global covariance pooling layer. The classification module is composed of a fully connected layer that serves as a binary classification. This network first introduces global covariance pooling into steganalysis to exploit the second-order statistic of high-level features for further improving the performance.

C. Dataset enrichment for steganalysis

Adding additional datasets [30] is a common method to be used to enrich the initial training set, e.g., merging the BOSSBase [44] and the BOWS2 [45] in the steganalysis task. As stated in section I, caution should be taken when merging two datasets. Yedroudj et al. in [39] propose that the steganalyst needs to have an access to knowledge about:

1) raw cameras used for generating the target database or 2) original RAW images and their development when augmenting the learning database. But the two possible ways to enrich the database are very restrictive.

Data augmentation has been proved to be an effective method to improve accuracy for deep learning tasks, and it has been applied in various fields, such as natural language processing [46], computer vision [47] and speech recognition [48]. With the development of CNN-based steganalysis, data augmentation techniques have also been applied to the steganalysis field. Some data augmentation methods that are effective for computer vision tasks are still effective for steganalysis, such as image mirror flipping and image rotation. However, not all methods can be directly migrated due to the different focus of steganalysis and computer vision tasks. For example, image scaling does not affect the semantic features of images, so it is valid for computer vision tasks. But image scaling does not work in the steganalysis task because it can lead to a loss or increase in image detail, and the detailed information is the concern of the steganalysis task.

In [39], the author generated a new cover from the original cover for dataset expansion. The author proposed two methods. The first method applies a sub-pixel image translation of 0.5 pixel followed by a cropping operation to obtain a new 256 × 256-pixel image. And the second is to upsample with a Lanczos3 filter to obtain a 512 × 512-pixel image and then downsample it with the same interpolation kernel to obtain a 256 × 256-pixel image. Both of the above processes change the details of cover images, which is what the steganalysis task is concerned with. Naturally, these processes will instead reduce the accuracy of the steganalysis network.

Yedroudj et al. [36] proposed a data augmentation method called pixels-off, which randomly selects a small number of pixels in the original cover and sets their values to 0 to obtain the augmented cover. The method adds a small amount of noise to the original image, and it can improve the performance of CNN-based steganalysis. However, since the positions of noise are randomly selected and the noise is added in such a way that all of the selected pixels are set to 0, so this will inevitably change the original cover distribution from the perspective of steganalysis. Further, Yedroudj et al. [36] also proposed an improved version of pixels-off (denoted as adaptive-pixels-off): some pixels are randomly turned off at the 10% of pixels with the highest embedding probability. Adaptive-pixels-off can further improve the performance of the steganalysis network. However, it is empirically designed and relies on a specific steganography algorithm to perform data augmentation manually, and we think it still has room for improvement.

III. The Proposed Method

A. Motivation

Currently, steganalysis models based on deep learning have advanced performance. When training deep learning models, we need a large amount of data to improve the accuracy and generalization ability of the model. Specific to steganalysis tasks, since the steganographic signals are very subtle at low payloads, the network needs a large amount of data to capture the steganographic signals. Therefore, it is necessary to enrich the dataset if the cover/stego pairs in the original dataset are insufficient.

Adding additional datasets is an intuitive way to enrich the dataset. As presented in Section II, the problem is that it is difficult to ensure that the data distribution of the added dataset is similar to that of the original dataset. This will have a negative effect if datasets with very different data distributions are added.

Data augmentation is a common way to enrich a dataset. In CNN-based steganalysis, pixels-off [31] has good performance. It achieves data augmentation by adding noise, which is added in a manual heuristic way. We believe it still has room for enhancement and propose ways to improve it.

Based on the requirements of data augmentation (which will be introduced in Section III-B) and the characteristics of steganalysis, we designed a data augmentation framework specifically for steganalysis, in which the augmented dataset is generated with a similar distribution to the original dataset from the perspective of steganalysis. The proposed method does not require access: 1) to images other than those of the initial learning database and 2) to the original cameras (or the original raw images) or any knowledge about the development. Moreover, the proposed method can expand the size of the original dataset by several times, which greatly enriches the content of the dataset. In this way, the networks have a greater possibility to capture the steganographic signals so that the networks converge more easily and perform better.

B. Framework

Before introducing our framework, we propose two basic requirements for data augmentation:

- The distribution of the augmented data is similar to that of the original data. Since the steganalysis task is more concerned with the high-frequency signals of images, keeping the high-frequency components invariant to a large extent keeps the image distribution invariant from the perspective of steganalysis.
- Enriching the content of the dataset. Ensure that the augmented image differs in content from the original image to enrich the dataset.

Based on the above two requirements, we designed a cover augmentation network as shown in Figure 1. The network can intelligently add noise to the original cover to generate the augmented cover.

Flow of the cover augmentation network.

(1) The original cover X goes through the UNet to generate the probability map of adding noise (P in Figure 1). The structure of the UNet is shown in Figure 2. The UNet consists of eight convolutional layers and eight deconvolutional layers. The feature maps of the same size are shortcut connected, and the output is obtained by $\mathrm{ReLU}(\mathrm{sigmoid}(\cdot) - 0.5)$ operation, so the modification probability is clamped in the interval $[0, 0.5]$.

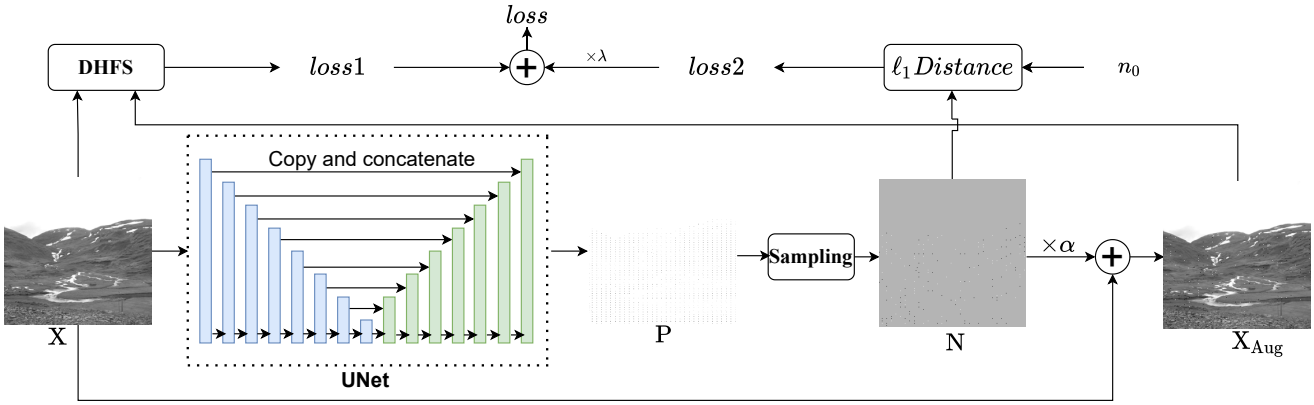


Fig. 1: The structure of the proposed cover augmentation network.

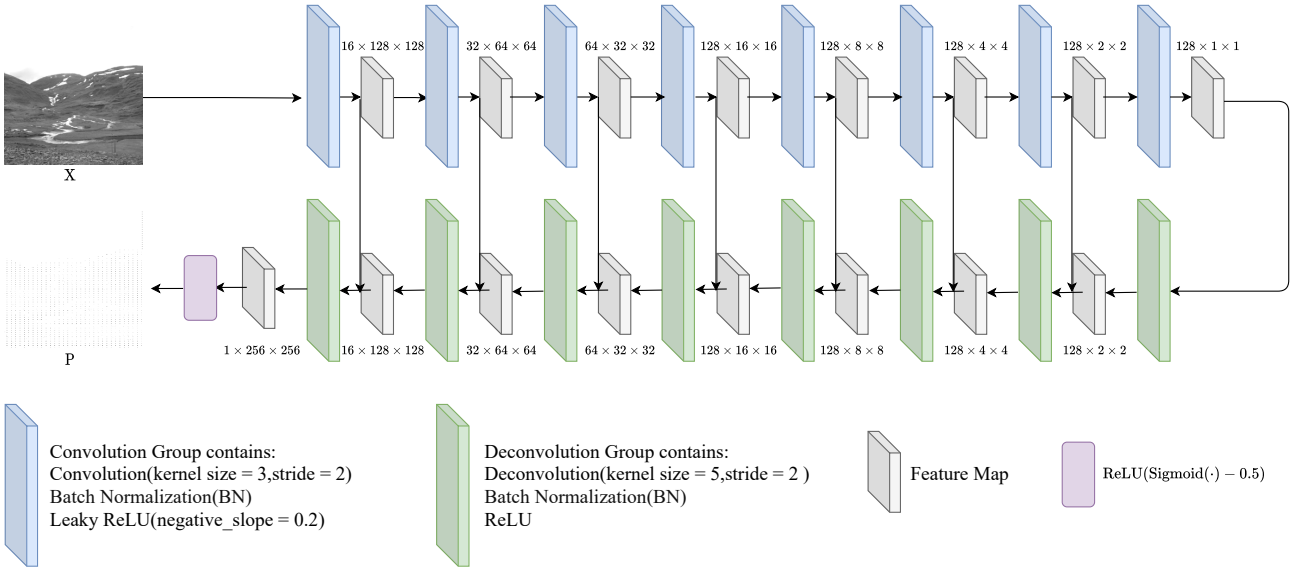


Fig. 2: The structure of UNet used in the cover augmentation network.

(2) Sample the probability map P (using Gumbel-softmax trick, which will be introduced later in the paper) to obtain the actual noise map N as shown in Figure 1. The elements in N have three kinds of values: $+1$, -1, and 0, indicating additive noise, subtractive noise, and no modification, respectively.

(3) Multiply the noise map N by the amplitude α and then add it to the original cover to get the augmented cover $\mathrm{X_{Aug}}$.

Gumbel-softmax trick [49]–[51]. The output of the UNet network is the probability map of adding noise, and sampling is needed to get the actual noise map. If we sample directly, we only get the sampled values, not the sampled expressions, so that the gradient of this process cannot be back-propagated. To get the expression of the sampling process, we use Gumbel-softmax trick.

To sample without destroying the gradient propagation of the computational graph, we introduce the Gumbel distribution, by which we can obtain the required randomness in the sampling process. The specific expression of the noise n_{ij} with its probability p_{ij} is as follows:

$$n_{ij} = \arg\max_a(\log(p_{ij}^a) + g_{ij}^a), a = -1, 0, +1, \tag{1}$$

where p_{ij}^{+1} and p_{ij}^{-1} denote the probabilities of additive and subtractive noise respectively and p_{ij}^0 denotes the probability of not adding noise. We set $p_{ij}^{+1} = p_{ij}^{-1} = p_{ij}/2$ so $p_{ij}^0 = 1 - p_{ij}$. g_{ij}^a is the gumbel noise and the specific expression of g_{ij}^a is as follows:

$$g_{ij}^a = -\log(-\log(u_{ij}^a)), u_{ij}^a \sim Uniform(0, 1). \tag{2}$$

However, the $argmax$ function is non-differentiable, thus we leverage the distilled $softmax$ to approximate it. In this way, Equation (1) is transformed into the following equation:

$$n_{ij} = \sum_{k=-1,0,+1} k \times \frac{\exp(\frac{\log(p_{ij}^k)+g_{ij}^k}{\tau})}{\sum_{a=-1,0,+1} \exp(\frac{\log(p_{ij}^a)+g_{ij}^a}{\tau})}, \tag{3}$$

where τ is the temperature parameter. The smaller the τ, the closer the $softmax$ is to $argmax$. But if τ is too small, this will lead to gradient explosion when conducting

backpropagation. (The value of τ will be discussed later in the experimental part.)

Using the Gumbel-softmax trick, we obtained the explicit expression (3) for the noise n_{ij} with its probability p_{ij}. In this way, we transferred the randomness required in the sampling process to g_{ij}^a so that the derivative of p_{ij} can be obtained when conducting backpropagation.

Loss function. According to the basic requirements of data augmentation, we design two loss functions. Based on the first requirement that the augmented data distribution should be similar to the original data distribution from the perspective of steganalysis, we define high-frequency loss named $loss1$ as follows:

$$loss1 = |\mathrm{F} * \mathrm{X} - \mathrm{F} * \mathrm{X}_{\mathrm{Aug}}|_{l_1}, \tag{4}$$

where F is a set of 30 high-pass filters from SRM [52], X is the original cover, $\mathrm{X}_{\mathrm{Aug}}$ is the augmented cover, and "$*$" is the convolution operation. $loss1$ ensures the consistency of the high-frequency signals of the original cover and augmented cover, and thus largely ensures the consistency of the distribution under the steganalysis task. The "DHFS" (abbreviation of "Difference of High-Frequency Signals") module in Figure 1 plays the role of calculating the loss function $loss1$.

If we only use $loss1$ to constrain the network, the learned noise probability map will tend to 0. To meet the second requirement, i.e., enriching the content of the dataset, we design the content variance loss function named $loss2$ as follows:

$$loss2 = \left| \sum_{n_{ij} \in \mathrm{N}} |n_{ij}| - n_0 \right|_{l_1}, \tag{5}$$

where N is the generated noise map and n_0 is the hyperparameter. The value of n_0 indicates the expected amount of noise points to be added. If n_0 is too large, too much noise will affect the distribution of the original image; if n_0 is too small, the difference between the augmented image and the original image is too subtle to achieve the effect of enriching the dataset. We determine the optimal n_0 on the validation set.

The total loss function of the cover augmentation network is as follows:

$$loss = loss1 + \lambda \cdot loss2, \tag{6}$$

where λ is the hyperparameter. We experimentally select the optimal λ in the next section.

The full process of the proposed data augmentation. Based on the cover augmentation network, we further designed the full process algorithm of data augmentation (as shown in **Algorithm 1**).

As shown in **Algorithm 1**, 2nd line, only the cover is retained out of the training set since it is the one fed into the cover augmentation network. The "Augmentation-Net" in the fourth step is the cover augmentation network mentioned above. After being well trained, the network can generate augmented cover. Steps 6 to 11 while loops traverse the entire cover list. For each cover X, feed the cover X into Augmentation-Net to get the augmented cover $\mathrm{X}_{\mathrm{Aug}}$.

Algorithm 1 Data augmentation for steganalysis.

Input: train-set-list, τ, n_0, α; ▷ τ, n_0 and α are hyperparameters of the Augmentation-Net.
Output: Augmentation-set-list
1: Augmentation-set-list = [ ];
2: cover-list = get-covers(train-set-list);
3: num = length(cover-list), $iter = num$;
4: Model = Augmentation-Net(α, n_0, τ); ▷ Augmentation-Net is the cover augmentation network.
5: Training the Model;
6: **while** $iter$>0 **do**
7: X = cover-list($iter$);
8: $\mathrm{X}_{\mathrm{Aug}}$ = Model(X);
9: $\mathrm{Y}_{\mathrm{Aug}}$ = embedding($\mathrm{X}_{\mathrm{Aug}}$); ▷ The embedding algorithm is the same as that used in the original dataset.
10: Augmentation-set-list.append([$\mathrm{X}_{\mathrm{Aug}}$,$\mathrm{Y}_{\mathrm{Aug}}$]);
11: $iter - -$;
12: **return** Augmentation-set-list

Subsequently using the same embedding algorithm to generate stego $\mathrm{Y}_{\mathrm{Aug}}$ (Step 9). These generated pairs (cover/stego) are subsequently appended to the Augmentation-set-list (Step 10) and then be added to the training set.

Data augmentation renderings. Our data augmentation renderings are illustrated in Figure 3. The pattern of augmented noise has the following characteristics:

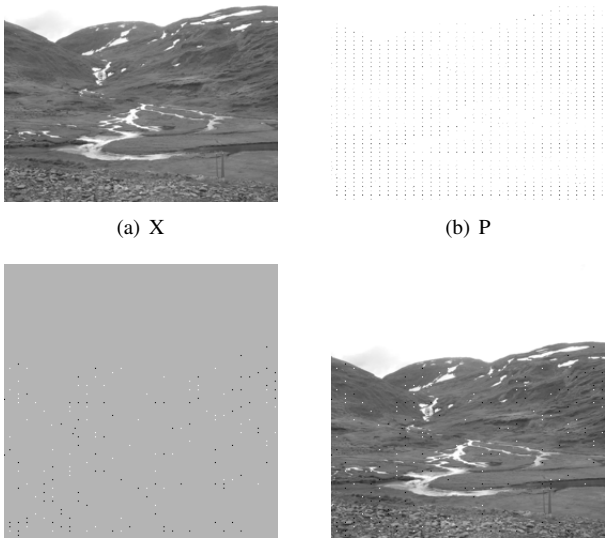
- The added noise is mainly distributed in the regions of complex texture (as shown in Figure 3(c)).
- The positions of the probability extremes of the added noise are regular and neatly arranged (as shown in Figure 3(b)).

IV. Experiments

A. Setups

1) Dataset: The experiment is carried out on the BOSSBase [44], which is composed of 10,000 pieces of spatial grayscale images of 512×512 pixels. In order to save computing time, we use MATLAB's "nearest" method to reduce the size of images to 256×256. When training the steganalysis network, 10,000 images are divided into three non-overlapping parts: 4,000 for the training set, 1,000 for the validation set, and 5,000 for the testing set. We determine the optimal hyperparameters on the validation set, and then evaluate the effect of the proposed data augmentation method on the testing set. All the experiments were run on an NVIDIA GEFORCE RTX 2080 Ti GPU card.

2) The method of steganography: The steganography method is based on a minimization distortion model. The distortion function is referenced to the S-UNIWARD method [53] and the HILL method [7]. The embedding is done using the simulated embedding method. The steganographic trace is subtle and the distribution of cover and stego is similar at low payloads, so steganalysis at low payloads is challenging. We explore the effect of our data augmenta-



(a) X (b) P (c) N (d) X_{Aug}

Fig. 3: Our cover augmentation renderings are illustrated on a 256×256 pixels image 1.pgm from BOSSBase [44]. (a) represents the original cover and (b) shows the probability map of added noise (the darker the color, the higher the probability). Gumbel sampling is performed on (b) to obtain (c). (c) shows the added noise (white, gray, and black represent -1, 0, and 1 respectively). (d) shows the final generated augmented cover. For demonstration purposes, we set the noise amplitude α = 255 (and clamped the pixel values to [0,255]) to highlight the added noise.

tion method at low payloads, and the payloads are set as $\{0.02, 0.04, 0.06, 0.08, 0.1, 0.2\}$ (bits per pixel, bpp).

3) Baseline: We call the method that performs only virtual augmentation (including mirror flip and image rotation) on the original dataset *Base*.

Pixels-off [36] selects n pixels in cover at random and sets their pixel values to 0 to achieve the data augmentation. According to pixels-off [36] and Figure 3(b) from our experiment, it is better to select the noise at the location with higher steganographic modification probability, therefore, we select n points with the maximum modification probability and set their pixel values to 0. According to pixels-off [36], the best results are obtained at $n = 400$, so in subsequent experiments we set $n = 400$. This control experiment is noted as *pixels-off_Single*. Moreover, pixels-off combines different augmentation datasets generated by different n together and this will also play a good effect. Combine the augmentation datasets obtained by taking n = 100, 256, and 400 together to get a triple expanded dataset, recorded as *pixels-off_Triple*.

4) Experimental setup for training the proposed cover augmentation network: The network structure and loss function are described in Section III. In the following, we will introduce the experimental setup during training the cover augmentation network.

The dataset for training the cover augmentation network consists of the original cover. We apply the Adam optimizer to train the cover augmentation network. Due to GPU memory limitation, the mini-batch size in the training is set to 16. All layers are initialized using the Xavier method [54]. Based on the above settings, the networks are then trained to minimize the loss function mentioned in section III. During the training, we adjust the learning rate as follows: The initial learning rate is set to 0.0001. When the training iteration equals one of the specified step values, the learning rate will be divided by 10. Concretely, we set the total epoch to 300 and the learning rate will be decreased at epochs 100 and 200, respectively. After the cover augmentation network is well trained, our data augmentation method can be achieved using **Algorithm 1**.

5) Evaluation metrics: The accuracy acc_t and Area Under the ROC Curve (AUC) [55] of the steganalysis network on the testing set are used as the evaluation metrics. The expression of acc_t is as follows:

$$acc_t = \frac{N_{correct}}{N_{test}}, \tag{7}$$

where $N_{correct}$ is the number of samples correctly predicted by the steganalysis network and N_{test} is the total number of samples in the testing set.

AUC [55] is defined as the area under the ROC curve (receiver operating characteristic curve). It is also a widely used metric to measure the performance of binary classifiers, with higher values indicating better performance.

B. Hyperparameters

1) τ in the Gumbel-softmax trick: As in Equation (3), the smaller the τ, the better the $softmax$ function matches the $argmax$ function and the more accurate the forward pass, but the gradient may explode in the backward, which will affect training.

In the process of training the cover augmentation network, we first take a large τ in order to facilitate the convergence of the network, and gradually reduce τ when the training iteration equals one of the specified step values. Concretely, the initial τ is set to 1, and τ will be divided by 10 at epochs 100 and 200 respectively. In this way, the impact of gradient explosion is avoided and forward pass is more precise.

TABLE I: Relationship between accuracy on the validation set (acc_v) and noise amplitude α.

α	1	2	4	8	16
acc	0.7665	0.7760	0.7780	0.7740	**0.7845**
α	32	64	128	256	
acc	**0.7785**	**0.7795**	0.7740	0.7700	

TABLE II: Relationship between accuracy on the validation set (acc_v) and expected number of noise points n_0.

n_0	100	256	400	1024
acc_v	0.7776	0.7770	**0.7845**	0.7815

2) Noise amplitude α and number of noise points n_0: We determine the optimal parameters α and n_0 on the validation set. Specifically, the hyperparameters corresponding to the highest accuracy of the network on the validation set are selected. According to the requirements of data augmentation: (1) The distribution of augmented data should be similar to the distribution of original data. To keep the distribution similar, the smaller the noise amplitude α and the number of points of noise n_0, the better. (2) Enriching the content of the dataset. In order to make the content of the augmented cover differ from the content of the original cover, the larger the noise amplitude α and the number of points of noise n_0, the better. So we should take a compromise under requirements (1)(2).

To explore the optimal noise amplitude α, we set $\alpha \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$, payload = 0.2 bpp, $n_0 = 400$. The results of CovNet on the validation set are shown in Table I; To explore the optimal n_0, we set $n_0 \in \{100, 256, 400, 1024\}$, payload = 0.2 bpp, $\alpha = 16$. The results of CovNet on the validation set are shown in Table II.

From Table I, the accuracy on the validation set is the top three high when α is set to 16, 64, and 32. Therefore, the noise amplitude α is set to 16 to generate a single expanded augmentation dataset, and the noise amplitude α is set to 16, 64, and 32 to generate a triple expanded augmentation dataset. In addition, as shown in Table II, the accuracy of the validation set is highest when the number of noise n_0 is 400, so the n_0 is fixed to 400 in the subsequent experiments.

TABLE III: Relationship between accuracy on the validation set (acc_v) and λ.

λ	0.01	0.1	1.0	10	100
acc_v	0.7801	0.7795	**0.7845**	0.7812	0.7815

3) The λ from Equation (6): To explore the optimal noise amplitude λ, we set $\lambda \in \{0.01, 0.1, 1.0, 10, 100\}$, payload = 0.2 bpp, $\alpha = 16$, $n_0 = 400$. The results of CovNet on the validation set are shown in Table III. As shown in Table III, when λ = 1.0, the model performs best on the validation set. Therefore, we choose λ = 1.0 to train our augmentation network in the subsequent experiments.

C. The impact on the cover statistics

As shown in Figure 4 and Figure 5, among pixels-off, adaptive-pixels-off and our method, pixels-off has the largest impact on the steganographic modification probability of cover, adaptive-pixels-off has the second largest impact, and our method has the smallest impact. Furthermore, we calculate the effect of the proposed augmentation method, pixels-off, and adaptive-pixels-off on several metrics.

Table IV presents the comparison of pixels-off, adaptive-pixels-off and our augmentation method on several metrics. APD [56] is the abbreviation of average pixel discrepancy and it is calculated as the L_1 norm of the difference between the original cover and the augmented cover; PSNR means peak signals to noise ratio; SSIM [57] is a metric to measure the similarity between two images and the larger the SSIM the more similar the two images are; DHFS indicates the difference of the high-frequency signals between two images. The high-frequency signals are obtained by feeding the images into SRM high-pass filters. The defining equation of DHFS is shown in Equation (4). The above metrics are calculated from 4,000 cover images of the original training sets and the corresponding augmentation dataset generated by the pixels-off, adaptive-pixels-off, and the proposed data augmentation method. The proposed method is better than both pixels-off and adaptive-pixels-off on all these four evaluation metrics, so it further verifies that the proposed method keeps the distribution of the original image better.

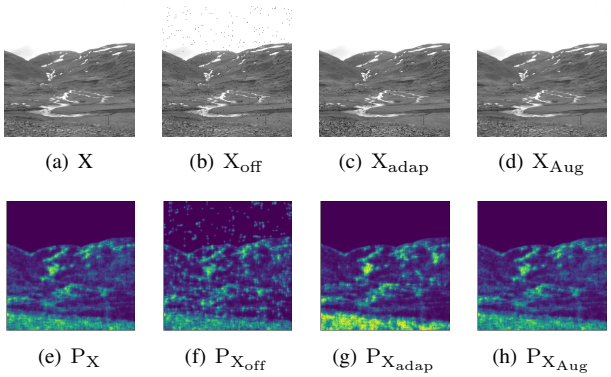


Fig. 4: The steganography modification probability map is illustrated on a 256×256 pixels image 1.pgm from BOSSBase [44], and we choose the pixels-off method as a comparison. (a) represents the original cover. (b) and (c) denote the augmented cover generated by pixel-off and adaptive-pixel-off, respectively. (d) shows the augmented cover generated by the proposed method ($\alpha = 16$, $n_0 = 400$). (e)-(h) shows the steganographic modification probability maps corresponding to (a)-(d), respectively (The probability maps are generated by the S-UNIWARD algorithm with a payload of 0.4 bpp. The more yellow area corresponds to the higher steganographic modification probability, and the darker area corresponds to the lower steganographic modification probability).

D. Evaluation of the proposed augmentation method

To verify the effectiveness of the proposed data augmentation method, we selected three networks (CovNet, SRNet, and YedNet) to evaluate. The networks are trained under three situations: *Base*, *adaptive-pixels-off* (including *pixels-off_Single* and *pixels-off_Triple* introduced in Section IV-A3) and our augmented dataset. We set the noise amplitude α to 16, 32, and 64 to generate three augmented datasets. The augmented dataset corresponding to the noise amplitude of 16 is denoted as *Ours_Single*. The three augmented datasets are combined to produce a larger augmented dataset, which is called *Ours_Triple*. We further explore the effects of combining *Ours_Triple* and *pixels-off_Triple* (denoted as *combine*). The accuracy (denoted as acc_t) and AUC on the testing set were used as the evaluation metric.

In Table V - Table X, the data on the left side and right side of the symbol " / " are the accuracy and AUC, respectively. As shown in Figure 6, Figure 7, and Table V - Table X, our

TABLE IV: The comparison of the proposed augmentation method, pixel-off, and adaptive-pixels-off on several metrics ("std" is the abbreviation for standard deviation).

Metrics	APD($mean \pm std$)	PSNR($mean \pm std$)	SSIM($mean \pm std$)	DHFR($mean \pm std$)
pixel-off	0.5823 ± 0.1954	29.92 ± 3.00	0.8994 ± 0.0422	1.7248 ± 0.5786
adaptive-pixel-off	0.5525 ± 0.2175	31.05 ± 4.21	0.9511 ± 0.0239	1.4655 ± 0.5757
Ours	0.0709 ± 0.0446	48.66 ± 3.09	0.9927 ± 0.0075	0.2087 ± 0.1336

TABLE V: Test results (acc_t/AUC) on CovNet (detecting S-UNIWARD).

Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5000/0.5007	0.5036/0.5070	0.5561/0.5936	0.6003/0.6679	0.6501/0.7332	0.7555/0.8623
pixels-off_Single	0.5011/0.5012	0.5000/0.5009	0.5183/0.5282	0.6412/0.7206	0.6698/0.7636	0.7814/0.8868
Ours_Single	0.4985/0.5009	0.5342/0.5504	0.6004/0.6662	0.6440/0.7275	0.6764/0.7694	0.7829/0.8858
pixels-off_Triple	0.4984/0.5000	0.5016/0.5009	0.5010/0.5079	0.6420/0.7241	0.6737/0.7686	0.7890/0.8925
Ours_Triple	**0.5220/0.5415**	0.5746/0.6293	0.6175/0.6911	0.6511/0.7406	0.6869/0.7832	0.7924/0.8973
combine	0.5017/0.5012	**0.5820/0.6344**	**0.6251/0.7048**	**0.6703/0.7655**	**0.6960/0.7958**	**0.8044/0.9069**

TABLE VI: Test results (acc_t/AUC) on SRNet (detecting S-UNIWARD).

Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5113/0.5226	0.5407/0.5815	0.5683/0.6183	0.6039/0.6690	0.6316/0.7110	0.6825/0.7752
pixels-off_Single	0.5001/0.5002	0.5008/0.5011	0.5032/0.5012	0.5674/0.5972	0.6053/0.6524	0.7403/0.8413
Ours_Single	0.5288/**0.5467**	0.5661/**0.6073**	0.5864/0.6413	0.6208/0.6925	0.6581/0.7387	0.7521/0.8359
pixels-off_Triple	0.4995/0.5003	0.5012/0.5002	0.5002/0.5000	0.5530/0.5344	0.5953/0.6017	0.7335/0.8263
Ours_Triple	**0.5296**/0.5436	**0.5669**/0.6017	**0.5997/0.6519**	**0.6313/0.6976**	0.6580/**0.7459**	**0.7706/0.8723**
combine	0.5002/0.5000	0.5006/0.5023	0.5035/0.5023	0.6239/0.5042	**0.6656**/0.7401	0.7628/0.8556

TABLE VII: Test results (acc_t/AUC) on YedNet (detecting S-UNIWARD).

Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5054/0.5074	0.5226/0.5380	0.5438/0.5780	0.5691/0.6164	0.5914/0.6501	0.6488/0.7328
pixels-off_Single	0.5087/0.5142	0.5000/0.5005	0.5089/0.5035	0.5486/0.5036	0.5975/0.6313	0.6872/0.7852
Ours_Single	0.5099/0.5139	0.5261/0.5447	0.5532/0.5854	0.5746/0.6245	0.6026/0.6644	0.6894/0.7852
pixels-off_Triple	0.4997/0.5002	0.5003/0.5016	0.5004/0.5037	0.5067/0.5028	0.6048/0.5619	0.7259/0.7458
Ours_Triple	0.5103/0.5147	**0.5347/0.5583**	**0.5594/0.6013**	0.5904/**0.6456**	0.6120/0.6797	0.7168/0.8176
combine	**0.5109/0.5174**	0.4997/0.5042	0.5401/0.5189	**0.5979**/0.6350	**0.6582/0.7134**	**0.7631/0.8544**

TABLE VIII: Test results (acc_t/AUC) on CovNet (detecting HILL).

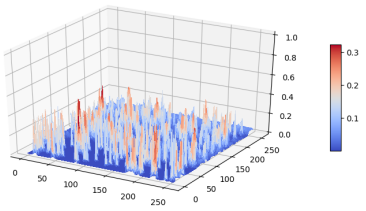
Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5012/0.5016	0.5191/0.5248	0.5760/0.6208	0.6035/0.6733	0.6376/0.7195	0.7280/0.8425
pixels-off_Single	0.5030/0.5097	0.5558/0.5875	0.5967/0.6595	0.6249/0.7155	0.6561/0.7533	0.7365/0.8614
Ours_Single	0.5026/0.5090	0.5612/0.6128	0.6023/0.6766	0.6267/0.7169	0.6562/0.7627	0.7383/0.8585
pixels-off_Triple	0.5031/0.5100	0.5100/0.5159	0.5974/0.6604	0.6307/0.7149	0.6568/0.7523	0.7518/0.8691
Ours_Triple	**0.5437/0.5559**	0.5827/0.6652	0.6199/0.7107	0.6461/0.7623	0.6706/0.7828	0.7567/0.8809
combine	0.5187/0.5242	**0.5919/0.6772**	**0.6297/0.7283**	**0.6573/0.7697**	**0.6875/0.7982**	**0.7699/0.8903**

TABLE IX: Test results (acc_t/AUC) on SRNet (detecting HILL).

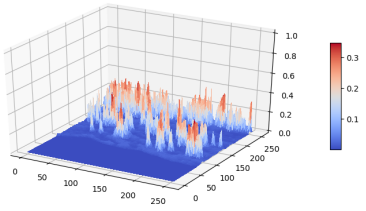
Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5228/0.5684	0.5499/0.5901	0.5833/0.6366	0.6057/0.6729	0.6289/0.7504	0.7029/0.8465
pixels-off_Single	0.5112/0.5239	0.5511/0.5970	0.5844/0.6387	0.6105/0.6814	0.6391/0.7590	0.7158/0.8646
Ours_Single	0.5291/0.5779	0.5632/0.6255	0.5880/0.6505	0.6194/0.6923	0.6457/0.7828	0.7175/0.8699
pixels-off_Triple	0.5085/0.5128	0.5532/0.6028	0.5819/0.6317	0.6142/0.6898	0.6385/0.7524	0.7186/0.8680
Ours_Triple	**0.5344/0.5803**	**0.5706/0.6351**	**0.6095/0.6739**	0.6213/0.7044	0.6496/0.7936	0.7291/0.8783
combine	0.5284/0.5761	0.5696/0.6285	0.6033/0.6595	**0.6290/0.7135**	**0.6598/0.8007**	**0.7393/0.8942**

TABLE X: Test results (acc_t/AUC) on YedNet (detecting HILL).

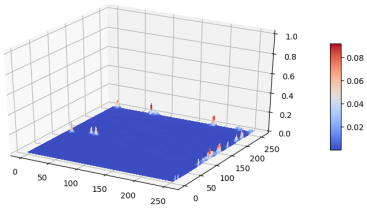
Setting	0.02 bpp	0.04 bpp	0.06 bpp	0.08 bpp	0.1 bpp	0.2 bpp
Base	0.5030/0.5067	0.5180/0.5341	0.5482/0.5605	0.5529/0.5744	0.5911/0.6534	0.6162/0.6848
pixels-off_Single	0.5053/0.5117	0.5291/0.5427	0.5640/0.5825	0.5698/0.5901	0.5937/0.6569	0.6574/0.7459
Ours_Single	0.5071/0.5152	0.5310/0.5499	0.5741/0.5953	0.5745/0.5960	0.6043/0.6688	0.6622/0.7520
pixels-off_Triple	0.5156/0.5239	0.5406/0.5707	0.5760/0.5958	0.5842/0.6315	0.6085/0.6788	0.6881/0.7807
Ours_Triple	**0.5200/0.5376**	0.5490/0.5827	0.5867/0.6336	0.5925/0.6409	0.6115/0.6833	0.6894/0.7836
combine	0.5194/0.5366	**0.5497/0.5836**	**0.5945/0.6437**	**0.5987/0.6582**	**0.6183/0.6906**	**0.7248/0.8333**



(a) Pixels-off

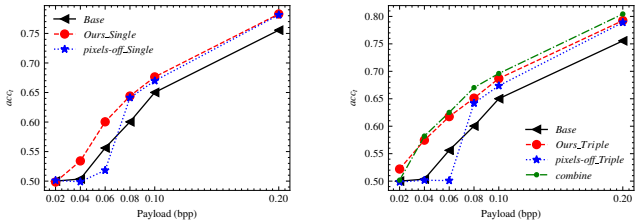


(b) Adaptive-pixels-off



(c) Ours

Fig. 5: Visualization by elevation for the differences between the steganographic modification probability. (a) shows the difference of steganographic modification probability between the original cover (Figure 4(e)) and its pixels-off version (Figure 4(f)), (b) shows the difference of steganographic modification probability between the original cover (Figure 4(e)) and its adaptive-pixel-off version (Figure 4(g), and (c) shows the difference of steganographic modification probability between the original cover (Figure 4(e)) and the proposed augmented cover (Figure 4(h))).



(a) CovNet-Single expansion (b) CovNet-Triple Expansion

(c) SRNet-Single expansion (d) SRNet-Triple Expansion

(e) YedNet-Single expansion (f) YedNet-Triple Expansion

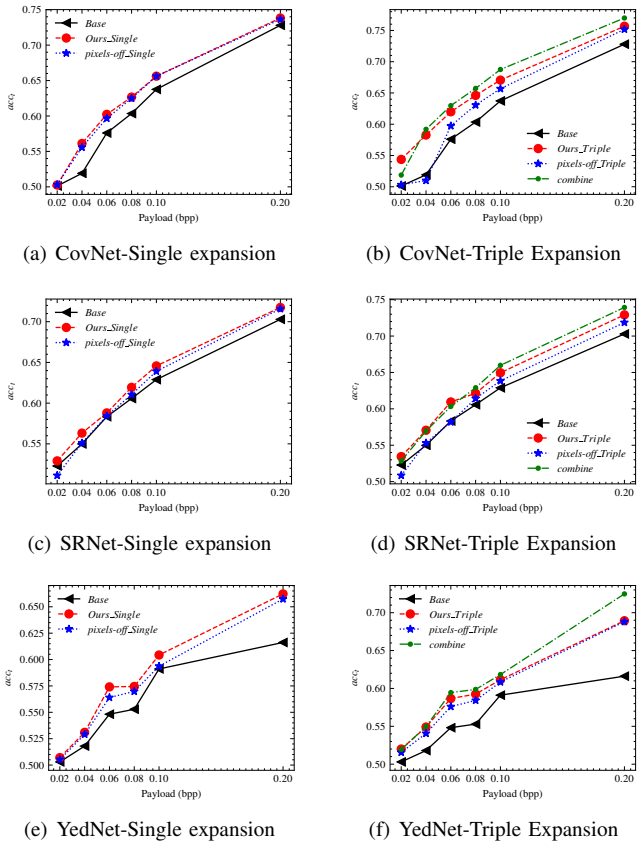
Fig. 6: Test results (acc_t) on CovNet, SRNet, and YedNet. S-UNIWARD is selected as the steganography algorithm. (a), (c), and (e) show the effect of our single-expansion dataset. (b), (d), and (f) show the effect of our triple-expansion dataset.

data augmentation method *Ours_Single* is effective and outperforms the *pixels-off_Single* method in the vast majority of cases. *Ours_Triple* further improves the network performance and outperforms *pixels-off_Triple*. Moreover, merging our augmented dataset *Ours_Triple* with pixels-off augmented dataset *pixels-off_Triple* (as the the method *combine* shows) further improves the performance of the network at relatively high payloads. The detailed analysis when detecting S-UNIWARD is as follows:

Performance on CovNet. CovNet fails to converge on the original training set of 4,000 cover/stego pairs with payloads below 0.04 bpp. At the payload of 0.02 bpp, CovNet can converge on our augmented dataset *Ours_Triple*, and at the payload of 0.04 bpp, CovNet can converge on our augmented dataset *Ours_Single* and *Ours_Triple*. However, the control data augmentation method *pixels-off_Single* and *pixels-off_Triple* even has a negative effect with payloads below 0.08 bpp. At higher payloads, our augmentation method is slightly better than the *pixels-off* method. And *combine* further improves the performance of CovNet with payloads above 0.02 bpp. The detailed experimental results are shown in Figure 6(a), Figure 6(b) and Table V.

Performance on SRNet. As can be seen from Figure 6(c), Figure 6(d) and Table VI, SRNet converges on the original training set at all payloads, and our method can further improve the performance of SRNet at all payloads. However, the control methods *pixels-off_Single* and *pixels-off_Triple* produce negative effects with payloads below 0.2 bpp. At higher payloads, the proposed method still prevails. On SRNet, the method *combine* does not work well and even produces negative effects with payloads below 0.08 bpp. At higher payloads the method *combine* is comparable to using our data augmentation method *Ours_Triple* alone.

Performance on YedNet. Our augmentation method improves the performance of YedNet at all payloads, but the control methods *pixels-off_Single* and *pixels-off_Triple* have a negative effect with payloads below 0.1 bpp. At higher payloads, the *pixels-off* method is comparable to our method.



(a) CovNet-Single expansion (b) CovNet-Triple Expansion

(c) SRNet-Single expansion (d) SRNet-Triple Expansion

(e) YedNet-Single expansion (f) YedNet-Triple Expansion

Fig. 7: Test results (acc_t) on CovNet, SRNet, and YedNet. HILL is selected as the steganography algorithm. (a), (c), and (e) show the effect of our single-expansion dataset. (b), (d), and (f) show the effect of our triple-expansion dataset.

The method *combine* improves the performance of YedNet significantly with payloads above 0.08 bpp. The detailed experimental results are shown in Figure 6(e), Figure 6(f) and Table VII.

As for detecting HILL, in most cases, the experimental results are similar to those of S-UNIWARD. The difference is that at low payloads *pixels-off_Single* and *pixels-off_Triple* detect HILL better than they detect S-UNIWARD, but still not as good as our method *Ours_Single* and *Ours_Triple*. The detailed experimental results are shown in Figure 7 and Table VIII - Table X.

As shown in Table VIII - Table X, the trends of AUC and accuracy of the steganalyzers are consistent in the vast majority of cases, which indicates that the improvement effect of our method on AUC is roughly consistent with the improvement in accuracy of the steganalyzers. Therefore, our approach is still effective and prevails from the AUC perspective.

E. Comparison with other data augmentation techniques

In this subsection, we test the performance of some easy-low-complexity augmentations on steganalysis tasks. These low-complexity methods include CutMix [40], BitMix [41], CutOut [42], and Mixup [43]. We also test a pixels-off with an embedding in high-frequency regions (randomly pick some pixels in the 10% pixels of the highest frequency power), and we denote this method as *pixels-off-HF*.

S-UNIWARD is adopted as the steganographic method in this experiment. Referring to the experimental results in the previous subsection, we know that at relatively high payloads our method has less advantage, so for more convincing we choose a relatively high payload (0.2 bpp). CovNet is selected for its high efficiency for training. To explore the performance of these augmentation methods with different sizes of training sets, there are two dataset partitioning schemes as follows: 1) BOSSBase is divided into three non-overlapping parts of sizes 4,000, 1,000, and 5,000, corresponding to the training set, validation set, and testing set. 2) All 10,000 images from BOSSBase are used as the training set, and the BOWS2 is divided into two non-overlapping parts of size 1,000 and 9,000, corresponding to the validation set and the testing set.

In Table XI and Table XII, the labels in the top row have the same meaning as in the previous subsection. The row where *Baseline* is located indicates that only our methods or pixels-off methods are used for data augmentation, without easy-low-complexity augmentations. The column where *Base* is located indicates that only the easy-low-complexity augmentations are used, without pixels-off methods or our methods. The rest of the data are the results of combining our methods or pixels-off methods with those easy-low-complexity augmentations.

From Table XI, Table XII, and Table XIII, we summarize as follows:

- Among these easy-low-complexity augmentations, BitMix, CutMix, and CutOut improve the performance of the steganalysis network, while Mixup has a negative impact. BitMix has the largest boost, with 1.41% and 0.72% for training set sizes of 4,000 and 10,000, respectively. The boosts of our method are 3.68% and 1.93% for training set sizes of 4,000 and 10,000, respectively, which is advantageous to these easy-low-complexity augmentations.
- As shown in the Table XIII, *pixels-off-HF* and the adaptive-pixels-off proposed in [36] have comparable performance, but are not as good as our method.
- The combination of our method and easy-low-complexity augmentations can further improve the performance of the network, with the best performance being "*BitMix + combine*", which improve by 6.05% and 3.17% at training set sizes of 4,000 and 10,000, respectively.
- The effect of data augmentations decreases as the size of training set increases. The performance gain obtained by data augmentation on a small training set is greater.

Computation complexity. When the size of the training set is 4,000 images of 256×256, CovNet takes about 62 seconds to train one epoch, and the easy-low-complexity augmentations consume less than 1 second, which is negligible compared to the time consumed by training the network. Tables XI and XII also give the total time consumption for training the CovNet. As shown in Tables XI and XII, those easy-low-complexity algorithms incur almost no additional time overhead. The time overheads of our approach and pixels-off are comparable. The method *combine*, while further improving network performance in most cases, also has a significantly higher time overhead.

TABLE XI: Test result (acc_t) and time consumption of various augmentations when the training set size is 4,000. The data in brackets indicate the change of acc_t compared to the data in the *Base* column, *Baseline* row.

Setting	*Base*	*pixels-off_Single*	*Ours_Single*	*pixels-off_Triple*	*Ours_Triple*	*combine*
Baseline	0.7555	0.7814(↑2.59%)	0.7829(↑2.74%)	0.7890(↑3.35%)	0.7923(↑3.68%)	0.8044(↑4.89%)
CutMix	0.7583(↑0.28%)	0.7795(↑2.40%)	0.7800(↑2.45%)	0.7886(↑3.31%)	0.7942(↑3.87%)	0.8103(↑5.48%)
BitMix	0.7696(↑1.41%)	0.7877(↑3.22%)	0.7970(↑4.15%)	0.7955(↑4.00%)	0.8000(↑4.45%)	**0.8160(↑6.05%)**
CutOut	0.7612(↑0.57%)	0.7815(↑2.60%)	0.7890(↑3.35%)	0.7852(↑2.97%)	0.7930(↑3.75%)	0.8020(↑4.65%)
Mixup	0.6911(↓6.44%)	0.7220(↓3.35%)	0.7360(↓1.95%)	0.7650(↑0.95%)	0.7707(↑1.52%)	0.7933(↑3.78%)
Time (hours)	3-4	6-7	6-7	12-13	12-13	24-25

TABLE XII: Test result (acc_t) and time consumption of various augmentations when the training set size is 10,000. The data in brackets indicate the change of acc_t compared to the data in the *Base* column, *Baseline* row.

Setting	*Base*	*pixels-off_Single*	*Ours_Single*	*pixels-off_Triple*	*Ours_Triple*	*combine*
Baseline	0.7059	0.7130(↑0.71%)	0.7171(↑1.12%)	0.7161(↑1.02%)	0.7252(↑1.93%)	0.7264(↑2.05%)
CutMix	0.7068(↑0.09%)	0.7203(↑1.44%)	0.7246(↑1.87%)	0.7191(↑1.32%)	0.7308(↑2.49%)	0.7309(↑2.50%)
BitMix	0.7131(↑0.72%)	0.7208(↑1.49%)	0.7258(↑1.99%)	0.7243(↑1.84%)	0.7344(↑2.85%)	**0.7376(↑3.17%)**
CutOut	0.7060(↑0.01%)	0.7120(↑0.61%)	0.7209(↑1.50%)	0.7192(↑1.33%)	0.7257(↑1.98%)	0.7250(↑1.91%)
Mixup	0.6981(↓0.78%)	0.6884(↓1.75%)	0.7004(↓0.55%)	0.6956(↓1.03%)	0.7056(↓0.03%)	0.7009(↓0.50%))
Time (hours)	8-9	16-17	16-17	28-29	28-29	53-54

TABLE XIII: Test result (acc_t) of *pixels-off-HF*. The data in brackets indicate the change of acc_t compared to *Base*.

Setting	*Base*	*pixels-off_Single*	*pixels-off-HF_Single*	*Ours_Single*	*pixels-off_Triple*	*pixels-off-HF_Triple*	*Ours_Triple*
size=4,000	0.7555	0.7814(↑2.59%)	0.7751(↑1.96%)	0.7829(↑2.74%)	0.7890(↑3.35%)	0.7907(↑3.52%)	**0.7923(↑3.68%)**
size=10,000	0.7059	0.7138(↑0.79%)	0.7130(↑0.71%)	0.7171(↑1.12%)	0.7161(↑1.02%)	0.7215(↑1.56%)	**0.7252(↑1.93%)**

TABLE XIV: Test results (acc_t) when the augmentation network is applied across datasets. The data in brackets indicate the change of acc_t compared to *Base*.

Steganalyzers	Method	*Ours_Single*	*Ours_Triple*	*combine*
CovNet (*Base*: 0.7555)	*Baseline*	0.7829(↑2.74%)	0.7924(↑3.69%)	0.8044(↑4.89%)
	Cross-dataset (BOW2)	0.7815(↑2.60%)	0.7940(↑3.85%)	0.8035(↑4.80%)
	Cross-dataset (SZUBase)	0.7810(↑2.55%)	0.7930(↑3.75%)	0.8007(↑4.52%)
YedNet (*Base*: 0.6488)	*Baseline*	0.6894(↑4.06%)	0.7168(↑6.80%)	0.7631(↑11.43%)
	Cross-dataset (BOW2)	0.6880(↑3.92%)	0.7150(↑6.62%)	0.7600(↑11.12%)
	Cross-dataset (SZUBase)	0.6905(↑4.17%)	0.7183(↑6.95%)	0.7560(↑10.72%)
SRNet (*Base*: 0.6825)	*Baseline*	0.7521(↑6.96%)	0.7706(↑8.81%)	0.7628(↑8.03%)
	Cross-dataset (BOW2)	0.7490(↑6.65%)	0.7627(↑8.02%)	0.7776(↑9.51%)
	Cross-dataset (SZUBase)	0.7459(↑6.34%)	0.7747(↑9.22%)	0.7780(↑9.55%)

F. Cross-dataset application of the augmentation network

In this sub-section, we explore the performance of the augmentation network in cross-dataset usage.

We retrain the augmentation networks on the BOWS2 [45] and SZUBase [10], and use them to perform augmentation on BOSSBase. S-UNIWARD at 0.2 bpp is selected as the steganography algorithm. CovNet, YedNet, and SRNet are selected as the steganalyzers.

The data in Table XIV show the accuracy of the steganalyzers. *Base*, *Ours_Single*, *Ours_Triple*, and *combine* have the same meaning as in Section IV-D. *Baseline*, *Cross-dataset (BOW2)*, and *Cross-dataset (SZUBase)* in Table XIV denote the augmentation networks trained on the BOSSBase, BOWS2, and SZUBase, respectively. For a fair comparison, only 4,000 images from each dataset are selected as the training set of the augmentation network.

As shown in Table XIV, the augmentation network trained on BOW2 and SZUBase is still effective for application on BOSSBase, and the performance is comparable to that of the augmentation network trained on BOSSBase itself. Therefore, we do not have to retrain the augmentation network when we need to apply the augmentation network on new datasets. In the actual implementation, we can directly use the pretrained augmentation network (without retraining) to generate augmented cover. This process is relatively low time-consuming (Generating 10,000 augmented images with an NVIDIA GEFORCE RTX 2080 Ti GPU card takes only 165 seconds).

G. Performance of the augmentation network for non-adaptive steganography

In this paper, we argue that steganalysis is more concerned with high-frequency information of images, which is based on the hypothesis of adaptive steganography. The previous experiments are all based on adaptive steganography (including S-UNIWARD and HILL), and in this subsection we explored the effect of our method on non-adaptive steganography. We chose LSBM [58] as the steganography algorithm and CovNet

TABLE XV: Test results (acc_t) of detecting non-adaptive steganography method LSBM. The data in brackets indicate the change of acc_t compared to *Base*.

Setting	*Base*	*pixels-off_Single*	*Ours_Single*	*pixels-off_Triple*	*Ours_Triple*	*combine*
0.01 bpp	0.7902	0.7938(↑0.36%)	0.7952(↑0.50%)	0.7991(↑0.89%)	0.8003(↑1.01%)	**0.8070(↑1.68%)**
0.1 bpp	0.9580	0.9603(↑0.23%)	0.9631(↑0.51%)	0.9630(↑0.50%)	**0.9649(↑0.69%)**	0.9643(↑0.63%)

as the steganalyzer. The dataset is set up the same as in the Section IV-A1.

As shown in Table XV, our augmentation method still has a boosting effect under LSBM steganography and outperforms the pixels-off method. However, compared to adaptive steganography, our approach has less boost for non-adaptive steganography. In general, adaptive steganography is more difficult to be detected, so our work mainly focuses on adaptive steganography detection.

V. Conclusion

In this paper, we design a cover augmentation network based on two basic requirements of data augmentation (not changing data distribution from the perspective of steganalysis and enriching content of the dataset) and characteristics of steganalysis (focusing on high-frequency signals), and we further propose a full-flow data augmentation algorithm based on the cover augmentation network. Experimental results show that the proposed method can effectively improve the network performance and outperform the previous methods. More importantly, our method enables the network to capture the steganographic signals more easily and induces the network to converge more easily at low payloads. However, there are still some defects in our method and we would like to improve them in future work. For example, the noise amplitude α and the expected number of noise points n_0 are determined on the validation set. In future work, we can explore ways for the network to learn these two parameters; in this paper, the method of keeping the distribution constant from the perspective of steganalysis is to maintain consistent high-frequency signals (generated by SRM high-pass filters). In the follow-up, we can further explore how to design better principles to keep the original data distribution constant under the steganalysis task.

References

[1] A. Cheddad, J. Condell, K. Curran, and P. McKevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Process.*, vol. 90, no. 3, pp. 727–752, 2010.

[2] B. Li, J. He, J. Huang, and Y. Q. Shi, "A survey on image steganography and steganalysis," *J. Inf. Hiding Multim. Signal Process.*, vol. 2, no. 2, pp. 142–172, 2011.

[3] M. Hussain, A. W. A. Wahab, M. Y. I. B. Idris, A. T. S. Ho, and K. Jung, "Image steganography in spatial domain: A survey," *Signal Process. Image Commun.*, vol. 65, pp. 46–66, 2018.

[4] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, 2011.

[5] V. Holub and J. J. Fridrich, "Designing steganographic distortion using directional filters," in *2012 IEEE International Workshop on Information Forensics and Security, WIFS 2012, Costa Adeje, Tenerife, Spain, December 2-5, 2012*. IEEE, 2012, pp. 234–239.

[6] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 2014, no. 1, p. 1, 2014.

[7] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 4206–4210.

[8] V. Sedighi, R. Cogranne, and J. J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 2, pp. 221–234, 2016.

[9] L. Guo, J. Ni, W. Su, C. Tang, and Y.-Q. Shi, "Using statistical image model for JPEG steganography: uniform embedding revisited," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2669–2680, 2015.

[10] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1547–1551, 2017.

[11] J. Yang, D. Ruan, J. Huang, X. Kang, and Y. Shi, "An embedding cost learning framework using GAN," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 839–851, 2020.

[12] W. Tang, B. Li, M. Barni, J. Li, and J. Huang, "An automatic cost learning framework for image steganography using deep reinforcement learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 952–967, 2021.

[13] X. Mo, S. Tan, B. Li, and J. Huang, "Mctsteg: A monte carlo tree search-based reinforcement learning framework for universal non-additive steganography," *CoRR*, vol. abs/2103.13689, 2021.

[14] W. Lie and G. Lin, "A feature-based classification technique for blind image steganalysis," *IEEE Trans. Multim.*, vol. 7, no. 6, pp. 1007–1020, 2005.

[15] B. Li, Z. Li, S. Zhou, S. Tan, and X. Zhang, "New steganalytic features for spatial image steganography based on derivative filters and threshold LBP operator," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1242–1257, 2018.

[16] L. Li, W. Zhang, K. Chen, and N. Yu, "Steganographic security analysis from side channel steganalysis and its complementary attacks," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2526–2536, 2019.

[17] Z. Jin, G. Feng, Y. Ren, and X. Zhang, "Feature extraction optimization of JPEG steganalysis based on residual images," *Signal Process.*, vol. 170, p. 107455, 2020.

[18] G. Feng, X. Zhang, Y. Ren, Z. Qian, and S. Li, "Diversity-based cascade filters for JPEG steganalysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 376–386, 2020.

[19] Y. Ma, X. Luo, X. Li, Z. Bao, and Y. Zhang, "Selection of rich model steganalysis features based on decision rough set α-positive region reduction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 2, pp. 336–350, 2019.

[20] J. J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, pp. 868–882, 2012.

[21] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," in *2014 IEEE International Workshop on Information Forensics and Security, WIFS 2014, Atlanta, GA, USA, December 3-5, 2014*. IEEE, 2014, pp. 48–53.

[22] V. Holub, J. J. Fridrich, and T. Denemark, "Random projections of residuals as an alternative to co-occurrences in steganalysis," in *Media Watermarking, Security, and Forensics 2013, Burlingame, CA, USA, February 5-7, 2013, Proceedings*, ser. SPIE Proceedings, vol. 8665. SPIE, 2013, p. 86650L.

[23] J. Kodovský and J. J. Fridrich, "Steganalysis in high dimensions: fusing classifiers built on random subspaces," in *Media Forensics and Security III, San Francisco Airport, CA, USA, January 24-26, 2011, Proceedings*, ser. SPIE Proceedings, vol. 7880. SPIE, 2011, p. 78800L.

[24] J. Kodovský, J. J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 432–444, 2012. [Online]. Available: https://doi.org/10.1109/TIFS.2011.2175919

[25] R. Cogranne and J. J. Fridrich, "Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 12, pp. 2627–2642, 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[28] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[29] G. Xu, H. Wu, and Y. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, 2016.

[30] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2545–2557, 2017.

[31] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-net: An efficient CNN for spatial steganalysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*. IEEE, 2018, pp. 2092–2096.

[32] M. Boroumand, M. Chen, and J. J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 5, pp. 1181–1193, 2019.

[33] X. Deng, B. Chen, W. Luo, and D. Luo, "Fast and effective global covariance pooling network for image steganalysis," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2019, Paris, France, July 3-5, 2019*. ACM, pp. 230–234.

[34] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "CALPA-NET: channel-pruning-assisted deep residual network for steganalysis of digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 131–146, 2021.

[35] J. Hestness, S. Narang, N. Ardalani, G. F. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *CoRR*, vol. abs/1712.00409, 2017.

[36] M. Yedroudj, M. Chaumont, F. Comby, A. O. Amara, and P. Bas, "Pixels-off: Data-augmentation complementary solution for deep-learning steganalysis," in *IH&MMSec '20: ACM Workshop on Information Hiding and Multimedia Security, Denver, CO, USA, June 22-24, 2020*. ACM, 2020, pp. 39–48.

[37] H. Ruiz, M. Chaumont, M. Yedroudj, A. O. Amara, F. Comby, and G. Subsol, "Analysis of the scalability of a deep-learning network for steganography "into the wild"," in *Pattern Recognition. ICPR International Workshops and Challenges - Virtual Event, January 10-15, 2021, Proceedings, Part VI*, ser. Lecture Notes in Computer Science, vol. 12666. Springer, 2020, pp. 439–452.

[38] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG image steganalysis using hybrid deep-learning framework," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1200–1214, 2018.

[39] M. Yedroudj, M. Chaumont, and F. Comby, "How to augment a small learning set for improving the performances of a cnn-based steganalyzer?" in *Media Watermarking, Security, and Forensics 2018, Burlingame, CA, USA, 28 January 2018 - 1 February 2018*. Ingenta, 2018.

[40] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 6022–6031.

[41] I.-J. Yu, W. Ahn, S.-H. Nam, and H.-K. Lee, "Bitmix: data augmentation for image steganalysis," *Electronics Letters*, vol. 56, no. 24, pp. 1311–1314, 2020.

[42] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[43] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[44] P. Bas, T. Filler, and T. Pevný, ""break our steganographic system": The ins and outs of organizing BOSS," in *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 6958. Springer, 2011, pp. 59–70.

[45] P. Bas and T. Furon, "Bows-2," 2007.

[46] C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Liu, "Freelb: Enhanced adversarial training for natural language understanding," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[47] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, p. 60, 2019.

[48] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*. ISCA, 2019, pp. 2613–2617.

[49] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[50] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[51] S. Bernard, P. Bas, T. Pevný, and J. Klein, "Optimizing additive approximations of non-additive distortion functions," in *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*. ACM, 2021, pp. 105–112.

[52] J. J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, pp. 868–882, 2012.

[53] V. Holub, J. J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP J. Inf. Secur.*, vol. 2014, p. 1, 2014.

[54] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, ser. JMLR Proceedings, vol. 9. JMLR.org, 2010, pp. 249–256.

[55] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[56] C. Zhang, P. Benz, A. Karjauv, G. Sun, and I. S. Kweon, "UDH: universal deep hiding for steganography, watermarking, and light field messaging," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[57] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[58] J. Mielikainen, "LSB matching revisited," *IEEE signal processing letters*, vol. 13, no. 5, pp. 285–287, 2006.