Normalizing Flows-Based Latent Variable Rearrangement for Generative Image Steganography

Sifan Wu[1], Li Dong [1*], Teng Sio Hong[2], Jiale Chen[3], Diqun Yan[1] and Rangding Wang[1*]
[1] Ningbo University, Ningbo, China
E-mail: {2211100283, dongli, yandiqun, wangrangding}@nbu.edu.cn
[2] Faculty of Humanities and Social Sciences, Macao Polytechnic University, Macau, China
E-mail: p1308598@mpu.edu.mo
[3] Beijing Institute of Technology, Beijing, China
E-mail: 3120245681@bit.edu.cn

***Abstract*— Image steganography is a technique for concealing secret messages within images. Traditional steganography methods typically involve modifying the cover image, making them vulnerable to detection and attacks. Recently, generative image steganography, which embeds messages during image generation, has emerged as a more robust solution to this challenge. However, these methods often face a trade-off between the visual quality of the generated images and the effectiveness of message concealment. Current approaches are frequently criticized for their low steganographic payloads and poor extraction accuracy. To address these issues, we propose a generative steganography method based on Normalizing Flows. Our approach rearranges the values of Gaussian latent variables, used for image generation, according to the secret messages and predefined mapping rules. The secret messages are embedded in the adjusted Gaussian latent variables, while the image generation process remains unaffected, as the latent variable values and their distribution are preserved. Experimental results demonstrate the superior performance of our method in terms of imperceptibility, steganographic capacity, and extraction accuracy.**

I. INTRODUCTION

Image steganography is a technique of hiding the secret message into the image without being detected [1]. Currently, the main challenge of steganography is to maintain a trade-off in steganographic capacity, imperceptibility, and robustness [2].

Traditional steganography relies on the cover image to embed messages. Ideally, the message is embedded by minimizing distortion to the cover image. Initially, Least-Significant Bit (LSB) algorithm embeds a secret message by modifying the least significant bit of the image [3]. This method is simple and useful, but it is easy to be detected and attacked. Subsequently, adaptive steganography methods were proposed. Adaptive steganography based on Syndrome-Trellis Codes (STC) improves the security of steganography by embedding messages in a statistically secure manner [4]. Messages are not only embedded in the spatial domain of cover images but also embedded in the frequency domain [5]. With the development of deep learning, steganography methods based on deep neural networks have developed. These methods achieved high steganographic payload and undetectable [6]. However, the steganography method with cover images usually selects an existing image as the cover image. Then, the image is distorted to embed the secret message. However, the image distortion makes them susceptible to detection and attack.

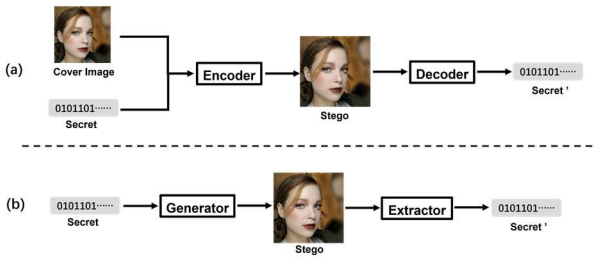

Fig. 1 Existing steganography frameworks: (a) Steganography with cover image. (b) Generative image steganography.

To address this problem, generative image steganography conceals messages during image generation and is difficult to detect. The workflows of the traditional and generative steganography methods are shown in Fig. 1. In comparison with traditional image steganography, the distribution of secret messages is mapped to the distribution of the latent variable used to generate the image. These methods not only improve the steganographic payload but also enhance the anti-detection ability and the visual quality of the generated images.

Generative Adversarial Networks (GANs), as a generative model, are widely used in generative steganography. The generator is trained to generate stego images that contain hidden messages, while a discriminator ensures that the generated stego images are imperceptibility from the images generated by latent variables without any mapping for the secret messages [7]. In addition, Normalizing Flows (NF) is also used as a generative model in the generative steganography. These methods embed secret messages into images using an invertible network

that allows for both embedding and extraction of the hidden messages [8]. They perform lossless mapping between the secret messages and the stego image. These methods retain high exact accuracy and are undetectable.

In this paper, a new generative steganography method based on NF is proposed. The performance and effectiveness of the proposed method are verified through experiments. The main contributions of this paper are as follows:

1). We propose a coverless steganography scheme based on Real-valued non-volume preserving (Real NVP).

2). A novel mapping method from secret information to latent variables is introduced.

3). Experimental results demonstrate that our scheme achieves higher undetectability compared to existing methods.

The rest of this paper is as follows: the second part introduces the related work. The third part describes the details of the proposed generative steganography framework. The fourth part shows the experimental results and analysis. The last part summarizes the research findings and discusses future research directions.

II. Related Work

A. Normalizing Flows

NF is a generative model that can map simple distributions into complex distributions through a series of invertible functions [9] [10] [11] [12] [13] [14]. NF was first proposed as a way to enhance the flexibility of variational inference [9]. Real NVP transformations are implemented using a series of coupled affine layers [12]. Real NVP offers efficient forward and inverse computations, making it suitable for tasks requiring reversible mappings, such as density estimation and generative modeling, with enhanced scalability and flexibility.

The fundamental of NF is the change of variable theorem. A series of neural network blocks f_i are connected sequentially to constitute the flows. The latent variable $\mathbf{z}_0$ is transformed to the latent variable $\mathbf{z}_k$ by the flows. The probability density function of the latent variable $\mathbf{z}_k$ can be modeled by using the change of variables theorem:

$$p_k(\boldsymbol{z}_k) = p_0(\boldsymbol{z}_0)\left|det\left(J_{f^{-1}}\right)\right|, \tag{1}$$

where $J_{f^{-1}}$ is the Jacobian matrix of the inverse transformation. The determinant $det(\cdot)$ quantifies the volume change caused by the transformation, ensuring that the probability density adjusts appropriately to reflect this change.

The goal of the normalized flows model is to generate images $\boldsymbol{x_i}$ that obey the distribution p_{data} of a given image dataset. The model as an image generator $\mathcal{G}$ can be expressed as:

$$\mathcal{G}^* = \underset{\mathcal{G}}{argmax} \sum_{i=1} logp_{\mathcal{G}}(\boldsymbol{x}_i) \tag{2}$$

NF model is trained to fit the log-likelihood function of the probability distribution of the dataset. Combine (1) and (2):

$$log\ p_{\mathcal{G}}(\boldsymbol{x}_i) = log\ p_z\left(\mathcal{G}^{-1}(\boldsymbol{x}_i)\right) + log\left|det\left(J_{\mathcal{G}^{-1}}\right)\right| \tag{3}$$

B. Generative Steganography Based on Normalizing Flows

NF is a powerful generative model, and the network is completely invertible. It is suitable for the task of image steganography.

NF can be used for steganography with cover images. It can implement high-capacity steganography which hides images in images [8]. Robust steganography can be implemented by adding some auxiliary network modules in the network framework [15]. Meanwhile, the Normalizing flows model can also be used to implement generative steganography. Some methods embed messages into the Gaussian latent variables which are used to generate the images. For example, the Gaussian variable values are encoded as float numbers, and the messages are hidden in the lower bits [16]. Gaussian variables are artificially constructed with the guidance of secret messages instead of random sampling [17]. Using conditionally invertible neural networks to guide image coloring, and the messages are hidden in the colors [18].

III. Proposed Method

We propose a novel mapping scheme from secret messages to latent variables, utilizing a latent variable rearrangement method. First, latent variables are sampled, and then they are rearranged based on the secret messages. This method ensures that the data distribution of the latent variables remains unchanged, thereby preserving high image generation quality. The overall framework details are outlined as follows.

A. Generating the Image with Embedded Message

The generating process is shown at the top of Fig. 2. It is organized into two steps: secret message embedding and stego image generation.

A binary secret message of length L bits is $\mathbf{S} \in \{0,1\}^L$. After preprocessing, encoding and encryption operations, the secret message $\mathbf{S}$ is converted to $\mathbf{x_s}$ of size $H \times W \times C$, and it obeys uniform distribution:

$$\mathbf{x_s} = \mathcal{F}(\mathbf{S}) \tag{4}$$

A Gaussian variable $\mathbf{z_0}$ of size $H \times W \times C$ randomly sample from the prior distribution $\mathbf{z} \sim N\left(\boldsymbol{\mu_0}, \boldsymbol{\sigma_0^2}\right)$. Subsequently, the values of $\mathbf{z_0}$ are rearranged by the mapping module $\mathcal{M}$ under the guidance of $\mathbf{x_s}$. The latent variable $\mathbf{z_s}$ which contains a secret message is obtained:

$$\mathbf{z_s} = \mathcal{M}(\mathbf{x_s}, \mathbf{z_0}) \tag{5}$$

The $\mathbf{z_s}$ is input to the image generative network $\mathcal{G}$ based on NF, and we can generate the stego image $\mathbf{y_{stego}}$ of size $H \times W \times C$:

$$\mathbf{y_{stego}} = \mathcal{G}(\mathbf{z_s}) \tag{6}$$

B. Extracting Process

The process of extracting the secret message is shown at the bottom of Fig. 2. It is the inverse process of the generating process, including Gaussian latent variable recovery and secret message inverse mapping.

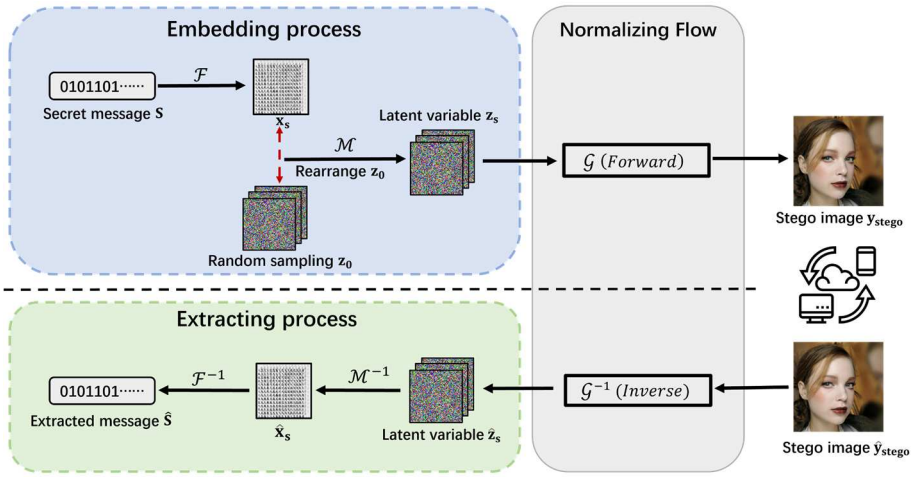

Fig. 2 The overview of the proposed method.

The stego image $\hat{\mathbf{y}}_{\mathbf{stego}}$ is received after transmitting. We input $\hat{\mathbf{y}}_{\mathbf{stego}}$ into the NF and inverse $\mathcal{G}^{-1}$. The recovered Gaussian variable $\hat{\mathbf{z}}_{\mathbf{s}}$ is obtained:

$$\hat{\mathbf{z}}_{\mathbf{s}} = \mathcal{G}^{-1}(\hat{\mathbf{y}}_{\mathbf{stego}}) \tag{7}$$

According to the mapping rule of the mapping module $\mathcal{M}$, we inverse map $\hat{\mathbf{x}}_{\mathbf{s}}$ with $\hat{\mathbf{z}}_{\mathbf{s}}$:

$$\hat{\mathbf{x}}_{\mathbf{s}} = \mathcal{M}^{-1}(\hat{\mathbf{z}}_{\mathbf{s}}) \tag{8}$$

Finally, the secret message is extracted by $\hat{\mathbf{x}}_{\mathbf{s}}$:

$$\hat{\mathbf{S}} = \mathcal{F}^{-1}(\hat{\mathbf{x}}_{\mathbf{s}}) \tag{9}$$

The whole processes of the embedding and extraction are organized in Algorithm 1 and 2.

Algorithm 1 Embedding process

Input: Secret message $\mathbf{S}$, Random Gaussian variable $\mathbf{z_0}$, Encoding operations $\mathcal{F}$, Mapping module $\mathcal{M}$, Normalizing Flows $\mathcal{G}$.

Output: Stego image $\mathbf{y_{stego}}$.

1: Encoding operations, $\mathbf{x_s} = \mathcal{F}(\mathbf{S})$.

2: Embedding secret message, $\mathbf{z_s} = \mathcal{M}(\mathbf{x_s}, \mathbf{z_0})$.

3: Generate the stego image, $\mathbf{y_{stego}} = \mathcal{G}(\mathbf{z_s})$.

4: Return $\mathbf{y_{stego}}$.

Algorithm 2 Extracting process

Input: Received stego image $\hat{\mathbf{y}}_{\mathbf{stego}}$, Decoding operations $\mathcal{F}^{-1}$, Inverse mapping module $\mathcal{M}^{-1}$, Inverse Normalizing Flows $\mathcal{G}^{-1}$.

Output: Extracted secret message $\hat{\mathbf{S}}$.

1: Normalizing Flows inverse, $\hat{\mathbf{z}}_{\mathbf{s}} = \mathcal{G}^{-1}(\hat{\mathbf{y}}_{\mathbf{stego}})$.

2: Inverse mapping, $\hat{\mathbf{x}}_{\mathbf{s}} = \mathcal{M}^{-1}(\hat{\mathbf{z}}_{\mathbf{s}})$.

3: Decoding operations, $\hat{\mathbf{S}} = \mathcal{F}^{-1}(\hat{\mathbf{x}}_{\mathbf{s}})$.

4: Return $\hat{\mathbf{S}}$.

C. *Message Mapping Module*

In the embedding process, the secret message $\mathbf{S}$ needs to be pre-processed firstly. Depending on the different steganographic payloads, it is encoded and resized to $\mathbf{x_s}$ of size $H \times W \times C$. When the payload is less than 3 bit/pixel, we encode $\mathbf{S}$ redundantly using BCH codes, repetition or neural network modules. When payload is more than 3 bit/pixel, state compression coding is required. The n-bits binary numbers are encoded into an integer of $[0, 2^n - 1]$. However, in practical scenarios, $\mathbf{S}$ is not ideally uniformly distributed after encoding. Hence, the $\mathbf{x_s}$ needs to be encrypted before the embedding, such as ChaCha20 [19].

The main idea for the mapping module $\mathcal{M}$: in order to embed the secret message and ensure that $\mathbf{z_s}$ and $\mathbf{z_0}$ have the same data distribution, the value of the Gaussian variable does not change. But the values of the Gaussian variables can be rearranged. Thus, the secret message cloud be embedded into the rearranged Gaussian variable $\mathbf{z_s}$. Details of how to realize the rearranged mapping are inspired by the different encoding methods of digital signals transmitted in physical layer, and we have two different mapping schemes:

1) Real-Valued Rearrangement Mapping (RVRM)

Referring to Non-Return-Zero Code (NRZ) code in signal transmission: a high-level signal represents 1 and a low-level signal represents 0. We do a similar mapping between $\mathbf{x_s}$ and $\mathbf{z_0}$. To preserve the distribution unchanged, the values of $\mathbf{z_0}$ should not be modified. Based on the integer values $[0, n]$ of $\mathbf{z_0}$, we divide $\mathbf{z_0}$ into n parts according to the probability density distribution. The values of $\mathbf{x_s}$ are mapped to correspond to the values of the n regions of $\mathbf{z_0}$. Thus, the values of $\mathbf{z_0}$ would be rearranged to obtain $\mathbf{z_s}$ according to the guidance of $\mathbf{x_s}$. In this way, the information of $\mathbf{x_s}$ is embedded into the Gaussian variable $\mathbf{z_s}$, and $\mathbf{z_s}$ have the same data distribution of $\mathbf{z_0}$.

However, the randomly sampled Gaussian variable is not a standard normal distribution. Dividing $\mathbf{z_0}$ into n parts based

on the probability density distribution will cause problems. The number of $\mathbf{z_0}$ values in the corresponding region does not match the number of integer values in the corresponding $\mathbf{x_s}$. Therefore, the values of $\mathbf{z_0}$ need to be slightly modified to match the number of values in the corresponding region.

Additionally, instead of dividing regions based on the probability density distribution, you can first sort the values in $\mathbf{z_0}$. Then we find the threshold value for each region based on the corresponding element amount in each region.

2) Difference Rearrangement Mapping (DRM)

The DRM is simple and accessible. However, it is vulnerable to rounding and precision loss. Thus, with reference to the Non-Return-Zero-Inverted Code (NRZI), the level signal flips to represent 0, and the level signal remains unchanged to represent 1. We similarly can map the values according to the difference between the i-th and $(i+1)$-th element values of $\mathbf{z_0}$. If the difference of adjacent element is more than the threshold α, it represents 1. Otherwise, it is less than the threshold β, it represents 0. Thus, we rearrange the values of $\mathbf{z_0}$ to get $\mathbf{z_s}$ with the guidance of $\mathbf{x_s}$. In this case, the $\mathbf{x_s}$ is embedded based on the difference of adjacent elements of $\mathbf{z_s}$.

At the same time, this scheme based on the change in the differences between adjacent elements can be extended. For example, we can map the values based on the differences between the elements at the same position but in different channels. It is even possible to map through the differences between the elements at the same position of the different variables.

In summary, two different schemes do not affect the quality of the generated images. It mainly affects the capacity of steganography and extraction accuracy. The real-valued rearrangement scheme is easier to implement mapping and has better steganography capacity. The difference rearrangement scheme has better extraction accuracy. We choose the suitable scheme according to different scenarios.

D. Image Generation Module

The image generation model $\mathcal{G}$ is based on Normalizing Flows. The model $\mathcal{G}$ is trained to use a randomly sampled Gaussian variable to generate a high-quality image. The generative image obeys the data distribution p_{data}.

When training the NF, the image dataset is used as the input of the model $\mathcal{G}^{-1}$. The model maximizes the likelihood of the prior distribution. The loss function of the model is:

$$\mathcal{L} = \log\ p_{\mathbf{z_0}}\left(\mathcal{G}^{-1}(\mathbf{y_{data}})\right) + \log\ |\det(J_{\mathcal{G}^{-1}})| \tag{10}$$

During the experiments, we use the pre-trained Real NVP model as the generative model $\mathcal{G}$ [18].

IV. Experiments

To evaluate the performance of the proposed generative steganography method, this section includes details of the experimental setup, performance evaluation, and comparison with baseline methods.

A. Implementation Details

All the experiments are conducted on an NVIDIA RTX 2080Ti GPU platform using Pytorch with Python.

1) Training Details

In our experiments, the generative model is Real NVP. The model trained on the CelebA dataset [20] to generate 64×64 images with 200 epochs. The batch size is set to 64. Features in residual blocks of the first few layers are set to 32. Number of residual blocks is set to 2. The model uses bottleneck, skip architecture, weight normalization, and affine coupling. The model does not use batch-norm coupling layer output. Adam is the optimizer, with the initial learning rate of 1e-3.

2) Evaluation metrics

In the experiments, we adopt the following evaluation metrics to evaluate the steganographic capacity, extraction accuracy, and anti-detectability.

To evaluate the visual quality of the generative image, we use no-reference image quality assessment, including FID [21] and Brisque score [22].

The steganographic capacity is measured by BPP, which represents the bits of information embedded in per pixel (bit/pixel) in the stego image:

$$\mathrm{BPP} = \frac{len(\mathrm{S})}{H \times W \times 3} \tag{11}$$

Where $len(\cdot)$ is the length of the hidden secret message, and H and W are the height and width of the steganographic image.

ACC is used to measure the extraction accuracy:

$$\mathrm{ACC} = 1 - \frac{||\hat{\mathrm{S}} - \mathrm{S}||_1}{\mathrm{len(S)}} \tag{12}$$

B. Comparison to Baselines

We evaluate the performances of the proposed methods in the aspects of extraction accuracy, anti-detectability, and imperceptibility under different hiding payloads, and compare with the following baselines: UDH [23], HiNet [24], SWE [25], and S2IRT [17].

1) Extraction accuracy

The information extraction accuracy is an important metric to measure the performance of steganography methods. TABLE I shows the information extraction accuracy at different hiding payloads.

The proposed steganography method with real-valued rearrangement mapping scheme (RVRM): when the hiding payloads are in the range of 1-2 BPP, the secret message is preprocessed and encoded.

The extraction accuracy can be achieved 100% by redundant coding and error correction code like BCH. At 3 BPP, the extraction accuracy can achieve 98%. As the hiding payloads continue to increase, a compression coding process is required during pre-processing. However, the extraction accuracy will be

reduced. If the recovery accuracy of the secret message is not required to be high, the steganography capacity can achieve 6-9 BPP.

TABLE I

The Extraction Accuracy with Different Hiding Payloads

Methods	0.25bpp	1 bpp	2 bpp	3 bpp
UDH	**100%**	98.21%	-	-
HiNet	-	92.94%	76.85%	74.31%
SWE	73.01%	71.34%	71.20%	71.22%
S2IRT	**100%**	**100%**	**100%**	99.43%
RVRM	**100%**	**100%**	**100%**	98.23%
DRM	**100%**	**100%**	**100%**	**99.98%**

The proposed steganography method with DRM has better anti-perturbation performance through difference mapping. It can effectively resistance to rounding and precision loss, and the extraction accuracy will be higher. Compared with RVRM, this method is more robust, but the steganography payload will be reduced.

2) Visual quality

To evaluate the quality of stego images, we use no-reference image quality evaluation strategies with FID [21] and Brisque score [22]. Our method generates 1,000 clean images by using randomly sampled latent variables and the corresponding stego images. The image quality scores are calculated in TABLE II. The lower scores indicate better performance. The visual quality scores of the stego images are compared with those of some existing steganography methods. As we can see from the results, the stego images generated by the proposed method have good visual quality. The Fig. 3 has some examples of randomly generated images.

The quality of the generated stego images by the proposed steganography method is mainly determined by the generation ability of the NF. The secret message embedding process only changes the order of the Gaussian variable values, which ensures that the Gaussian variables embedded with the secret message will not change the original data distribution.

TABLE II

The Scores of Stego Images with No-reference Image Quality Evaluation

Methods	FID↓	BRISQUE↓
SWE	-	9.07
S2IRT	-	**7.69**
Clean	108.35	19.56
Ours	**115.48**	17.45

We can see that the visual quality scores of the stego images are almost no different from the clean images. Therefore, the quality of the generated images will not be affected by the embedded information and the mapping schemes. The model used in our experiments is Real NVP. It is also feasible to change the generative model to the more advanced Normalizing flows model. And the generated images will have better visual quality.

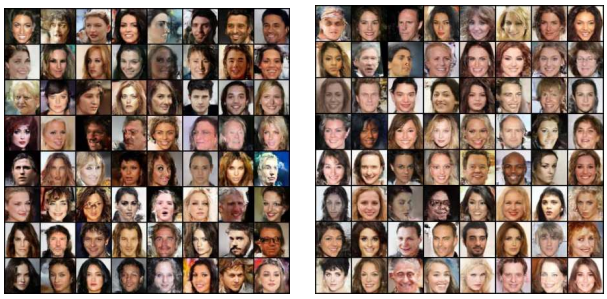

Fig. 3 The generated clean images (left) and stego images (right).

3) Steganalysis

The security is important to steganography algorithms. In order to evaluate the anti-detection ability of the proposed steganography method, we use the steganalysis methods Xu-Ne [26], SR-Net [27] to detect the stego images generated by our method. The dataset includes the clean images and corresponding stego images. TABLE III shows the statistical results of steganalysis and the comparison with existing methods. The detection accuracy is optimal when it approaches 50%. The statistics show that the proposed steganography method has higher security, and the detection accuracy is close to 65%. The stego image generation process of our proposed method is almost imperceptibility from the random generation process of clean images.

TABLE III

The Detection Accuracy (%) with Different Steganalysis Methods

Methods	Xu-Net	SR-Net
UDH	100%	100%
HiNet	100%	100%
SWE	73.53%	72.41%
S2IRT	74.90%	73.05%
Ours	**65.41%**	**61.11%**

In summary, the generated stego images using the proposed method are almost indistinguishable from clean images. The generated stego images can be considered the process of random image generation.

V. CONCLUSIONS

We propose a generative steganography method based on Normalizing flows. There are two mapping schemes to embed secret messages into random latent variables. The embedding and extraction will not change the data distribution of the stego images. Then the security of steganography can be guaranteed. The experiment results show that the quality of the stego images is determined by the NF, and it is not influenced by the information embedding. The generative stego images are almost imperceptibly from the clean images. In terms of extraction accuracy and payloads, the method in this paper has a good

comprehensive performance. In the future, we will continue to work on coverless steganography with the more advanced generative model and improve the robustness of this method.

VI. Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62171244, 61901237), Zhejiang Provincial Natural Science Foundation of China (Grant No. LY23F020011), Ningbo Natural Science Foundation- Young Doctoral Innovation Research Project (Grant No. 2022J080) (Corresponding author: Li Dong and Rangding Wang).

References

[1] Filler T, Judas J, Fridrich J. Minimizing additive distortion in steganography using syndrome-trellis codes[J]. IEEE Transactions on Information Forensics and Security, 2011, 6(3): 920-935.

[2] Zhang C, Lin C, Benz P, et al. A brief survey on deep learning based data hiding[J]. arXiv preprint arXiv:2103.01607, 2021.

[3] Anderson R J, Petitcolas F A P. On the limits of steganography[J]. IEEE Journal on selected areas in communications, 1998, 16(4): 474-481.

[4] Filler T, Judas J, Fridrich J. Minimizing additive distortion in steganography using syndrome-trellis codes[J]. IEEE Transactions on Information Forensics and Security, 2011, 6(3): 920-935.

[5] Lian S. Multimedia content encryption: techniques and applications[M]. Auerbach Publications, 2008.

[6] Baluja S. Hiding images in plain sight: Deep steganography[J]. Advances in neural information processing systems, 2017, 30.

[7] Volkhonskiy D, Nazarov I, Burnaev E. Steganographic generative adversarial networks[C]//Twelfth international conference on machine vision (ICMV 2019). SPIE, 2020, 11433: 991-1005.

[8] Jing J, Deng X, Xu M, et al. Hinet: Deep image hiding by invertible network[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 4733-4742.

[9] Rezende D, Mohamed S. Variational inference with normalizing flows[C]//International conference on machine learning. PMLR, 2015: 1530-1538.

[10] Kingma D P, Salimans T, Jozefowicz R, et al. Improved variational inference with inverse autoregressive flow[J]. Advances in neural information processing systems, 2016, 29.

[11] Papamakarios G, Pavlakou T, Murray I. Masked autoregressive flow for density estimation[J]. Advances in neural information processing systems, 2017, 30.

[12] Dinh L, Sohl-Dickstein J, Bengio S. Density estimation using real nvp[J]. arXiv preprint arXiv:1605.08803, 2016.

[13] Kingma D P, Dhariwal P. Glow: Generative flow with invertible 1x1 convolutions[J]. Advances in neural information processing systems, 2018, 31.

[14] Grathwohl W, Chen R T Q, Bettencourt J, et al. Ffjord: Free-form continuous dynamics for scalable reversible generative models[J]. arXiv preprint arXiv:1810.01367, 2018.

[15] Lan Y, Shang F, Yang J, et al. Robust image steganography: hiding messages in frequency coefficients[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(12): 14955-14963.

[16] Wei P, Luo G, Song Q, et al. Generative steganographic flow[C]//2022 IEEE international conference on multimedia and expo (ICME). IEEE, 2022: 1-6.

[17] Zhou Z, Su Y, Li J, et al. Secret-to-image reversible transformation for generative steganography[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 20(5): 4118-4134.

[18] Ren Y, Liu T, Zhai L, et al. Hiding data in colors: Secure and lossless deep image steganography via conditional invertible neural networks[J]. arXiv preprint arXiv:2201.07444, 2022.

[19] Bernstein D J. ChaCha, a variant of Salsa20[C]//Workshop record of SASC. 2008, 8(1): 3-5.

[20] Liu Z, Luo P, Wang X, et al. Deep learning face attributes in the wild[C]//Proceedings of the IEEE international conference on computer vision. 2015: 3730-3738.

[21] Heusel M, Ramsauer H, Unterthiner T, et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium[J]. Advances in neural information processing systems, 2017, 30.

[22] Mittal A, Moorthy A K, Bovik A C. No-reference image quality assessment in the spatial domain[J]. IEEE Transactions on image processing, 2012, 21(12): 4695-4708.

[23] Zhang C, Benz P, Karjauv A, et al. Udh: Universal deep hiding for steganography, watermarking, and light field messaging[J]. Advances in Neural Information Processing Systems, 2020, 33: 10223-10234.

[24] Jing J, Deng X, Xu M, et al. Hinet: Deep image hiding by invertible network[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 4733-4742.

[25] Hu D, Wang L, Jiang W, et al. A novel image steganography method via deep convolutional generative adversarial networks[J]. IEEE access, 2018, 6: 38303-38314.

[26] Xu G, Wu H Z, Shi Y Q. Structural design of convolutional neural networks for steganalysis[J]. IEEE Signal Processing Letters, 2016, 23(5): 708-712.

[27] Boroumand M, Chen M, Fridrich J. Deep residual network for steganalysis of digital images[J]. IEEE Transactions on Information Forensics and Security, 2018, 14(5): 1181-1193.