Generative Steganography Network

Ping Wei
pwei17@fudan.edu.cn
Fudan University
Shanghai, China

Sheng Li
lisheng@fudan.edu.cn
Fudan University
Shanghai, China

Xinpeng Zhang∗
zhangxinpeng@fudan.edu.cn
Fudan University
Shanghai, China

Ge Luo
18110240026@fudan.edu.cn
Fudan University
Shanghai, China

Zhenxing Qian∗
zxqian@fudan.edu.cn
Fudan University
Shanghai, China

Qing Zhou
21110240055@m.fudan.edu.cn
Fudan University
Shanghai, China

ABSTRACT
Steganography usually modifies cover media to embed secret data. A new steganographic approach called generative steganography (GS) has emerged recently, in which stego images (images containing secret data) are generated from secret data directly without cover media. However, existing GS schemes are often criticized for their poor performances. In this paper, we propose an advanced generative steganography network (GSN) that can generate realistic stego images without using cover images. We firstly introduce the mutual information mechanism in GS, which helps to achieve high secret extraction accuracy. Our model contains four sub-networks, i.e., an image generator \( G \), a discriminator \( D \), a steganalyzer \( S \), and a data extractor \( E \). \( D \) and \( S \) act as two adversarial discriminators to ensure the visual quality and security of generated stego images. \( E \) is to extract the hidden secret from generated stego images. The generator \( G \) is flexibly constructed to synthesize either cover or stego images with different inputs. It facilitates covert communication by concealing the function of generating stego images in a normal generator. A module named secret block is designed to hide secret data in the feature maps during image generation, with which high hiding capacity and image fidelity are achieved. In addition, a novel hierarchical gradient decay (HGD) skill is developed to resist steganalysis detection. Experiments demonstrate the superiority of our work over existing methods.

CCS CONCEPTS
• Information systems → Multimedia information systems;
• Security and privacy → Security services.

KEYWORDS
Steganography, Generative steganography, Data hiding, GANs

∗Xinpeng Zhang and Zhenxing Qian are the corresponding authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM MM ’22, October 10–14, 2022, Lisboa, Portugal.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9203-7/22/10...
https://doi.org/10.1145/3503161.3548217

Figure 1: Schematic diagram of generative steganography. Input secret data is converted to natural stego images by a generator. Then these images are transmitted through lossless channels. The hidden secret data can be recovered from received stego images by an extractor.

1 INTRODUCTION
Steganography is a technique that hides secret data in cover media for converting communication[8]. Various types of cover media have been investigated for steganography[13], including the digital audio [38], image [32], video [35] and text [2], where image is the most popular media for data hiding. Early steganographic methods often modify the pixel values of the cover image’s least significant bits to hide secret data[25]. Later, researchers paid more attention to the syndrome trellis coding (STC) framework[7, 16], which aims to minimize the distortion caused by modifying cover images. These schemes can be called the carrier-modified-based methods, as cover images are modified to hide secret data.

Recently, a few deep learning (DL) based steganographic methods have been proposed[30]. Tang[31] applies a network to learn the probability map of pixel alteration for data embedding. In [10], an adversarial training strategy is employed according to the 3-players game. Chu[5] studies how CycleGAN[48] hides a source image into the generated images in an imperceptible way. Zhu[47] and Zhang[43] propose two novel networks to hide random binary secret data in the cover images. Baluja[1] presents a system to hide color images inside another with minimal quality loss. Zhang[42] proposes a novel universal deep hiding architecture (UDH) to disentangle the encoding of secret images from the cover image. Yu[39]
we propose a flexible image generator that can flexibly synthesize with tailored GS schemes, these DL-based GS methods can generate will cause visual or statistical distortions, making the stego images easily detectable by steganalysis tools[3, 9, 37]. Once detected, the behavior of covert communication fails.

To solve the problem, a new steganography manner called generative steganography (GS) emerges [27]. Instead of modifying the cover image to embed secret data, it aims to synthesize stego images directly from secret data, as illustrated in Fig.1. Cover images are not required in GS, thus, steganalysis tools will become ineffective. Several approaches[17, 24, 26, 36] have been proposed to synthesize some particular stego images to hide secret data, like texture image[24, 34] and fingerprint image[17]. They can be called tailored GS methods for short. Nevertheless, only some special image types can be synthesized by them. Their data hiding capacities are much lower compared to traditional carrier-modified based works.

To make it more practical, researchers now use neural networks to generate natural stego images[28]. They often map the secret data to the input labels[18, 44] or noise vectors[12, 40] of GANS by a pre-trained generator rule. Then stego images can be generated with the assigned noise vectors or labels according to the mapping rule. Data receivers can extract the hidden secret data from received stego images by a pre-trained extractor. For simplicity, we term such schemes as deep learning (DL) based GS solutions. Compared with tailored GS schemes, these DL-based GS methods can generate more natural stego images. However, their data hiding capacities are much lower due to the limitation of label number and noise vector size. Meanwhile, stego images generated by these methods are often visually poor. Transmitting such images may arouse suspicion of the monitor in covert communication.

In this paper, we propose a novel generative steganography network (GSN) integrated with mutual information mechanism. To disguise transmitting secret messages through stego images, we propose a flexible image generator that can flexibly synthesize cover images or stego images according to the inputs. The main contributions of our work are:

1) We propose a holistic steganography solution for covert communication, in which mutual information is first introduced in generative steganography.
2) We hide the function of generating stego images in a flexibly constructed generator, which can generate either cover or stego images depending on the inputs.
3) A novel technique called hierarchical gradient decay (HGD) is proposed to improve the steganalysis resistance.
4) Proposed GSN achieves better performances than state-of-the-art works. It can synthesize realistic stego images with high hiding capacity, secret extraction accuracy, and security.

2 RELATED WORKS
Most of the published steganographic works are carrier-modified based solutions that require cover images[1, 30]. But this paper focuses on generative steganography (GS) that doesn’t require cover images. Existing GS schemes can be roughly classified into two categories: tailored GS and deep learning (DL) based GS.

Tailored generative steganography In these methods, stego images are synthesized according to some handcrafted procedures, where the secret data is encoded into specific textures or patterns. Otori[24] first proposes encoding secret data into dotted patterns, then stego texture images are synthesized by painting these patterns. In [34], the authors propose a secret-oriented texture synthesis solution, where secret data is embedded by pasting proper source texture on different locations of the synthesized images, referring to an index table. In [36], the authors suggest using marbling images to hide secret data. Secret messages are printed on the background of an image, and then this image is deformed into different marbling patterns using reversible functions. In [17], secret data is encoded as the positions and polarities of minutia points in fingerprint images. Then, stego fingerprint images can be constructed using a phase demodulation model using these encoded minutiae positions. The main disadvantage of these tailored GS schemes is that their generated stego image contents are unnatural, which is unsafe or even suspicious in covert communication.

DL-based generative steganography To produce natural stego images, some researchers propose to synthesize images with networks. Works[5, 6] use GANs to transform secret images into meaning-normal stego images, and these stego images can be converted back with another image generator. But they cannot convey random secret data as other steganographic methods do. It is more practical to transmit secret data in steganography. Therefore, Liu[18] and Zhang[44] propose to map binary secret data to the class labels of ACGAN[23]. By which, different stego images can be generated with the corresponding labels according to the given secret data and mapping rule. The labels of stego images can be extracted using a classifier, and then hidden secret data is recovered according to the mapping rule. Similarly, Hu[12] establishes a mapping rule between the secret data and the input noise vector of DCGAN[29]. Then stego images can be generated with the mapped noise values per the mapping rule and given secret data. In work [46], the authors propose an image inpainting-based GS solution. Where secret messages are embedded in the remaining region of a corrupted image with Cardan grille, then this image is fed into a pre-trained generator for stego image generation. Wang[33] proposes to generate stego images from the concatenation of binary secret data and noise vector using GANS, in which secret data is input to the generator directly. Though the DL-based GS schemes can produce realistic stego images, they are rather rudimentary with poor performance. The stego images generated by them are often of low visual quality. In addition, their hiding capacities are often limited to several hundred bits per image. They can not be improved with the increase of stego image size due to the limitation of GANs’ input dimensions. For example, only 6 bits of secret data are conveyed by every 28×28 stego image in work [18], and each 32×32 stego image carries 400 bits of secret data in [33].

3 PROPOSED METHOD GSN
The architecture of the proposed GSN is given in Fig.2, which consists of a generator (G), a discriminator (D), a steganalyzer (S) and an extractor (E). D and S are used as two discriminators in GANs, which can ensure the visual quality and reduce the difference between generated cover/stego images, respectively. The inputs of
where, $L_3.1$ Problem Formulation

In our scheme, a stego image can be generated when $(z, n)(z, d)$ is input. $D$ and $S$ act as dual discriminators to ensure the visual quality and statistical imperceptibility of cover/stego images. $E$ aims to reveal the hidden secret from generated stego images.

GSN include a latent vector $z$, a noise matrix $n$ or a three dimensional matrix of secret data $d$. The generator can produce either a cover image $x^c$ or stego image $x^s$, depending on which of $(z, n)$ and $(z, d)$ is input. Then, real image and the generated stego image are sent to the discriminator to decide whether they are real or fake. Meanwhile, the generated cover/stego images are fed into the steganalyzer for differentiation. The generated stego images are input to the extractor, and $d'$ is the predicted secret.

3.1 Problem Formulation

In our scheme, a stego image can be generated with secret data $d$ and latent $z$, i.e. $x^s = G(z, d)$. Secret data influences the image content, which should be recovered exactly from the generated stego image $x^s$. From the perspective of information theory, the mutual information between $d$ and $x^s$ is expected to be maximized, i.e., $\max_{d} I(d, G(z, d))$. That is to say, input secret data and the generated stego images are closely related. Different stego images should be generated when the input secret data is varied, and the hidden secret is hoped be extracted accurately from generated stego images. Therefore, we incorporate the mutual information into GANs for data hiding. The loss function can be defined as:

$$\min_{G} \max_{D} \mathcal{L}_{adv}(D, G) - \lambda \cdot I(d, G(z, d)),$$  \hfill (1)

where, $\mathcal{L}_{adv}(D, G)$ is the adversarial loss of GANs, i.e. $\mathcal{L}_{adv} = \mathbb{E}_{x,p_d \sim p_d} \log D(x) + \mathbb{E}_{x^s, p_{d^s} \sim p_{d^s}} \log (1 - D(G(z, d)))$. Here, $\mathbb{E}$ stands for the expectation. $I(d, G(z, d))$ is the mutual information between secret data and the generated stego image. $G$ wants to minimize while $D$ expects to maximize this loss function.

But mutual information $I(d, G(z, d))$ is hard to be achieved as it requires the posterior distribution $p(d \mid G(z, d))$. Inspired by Infogan[4], a variation lower bound $\mathcal{L}_{d}(G, E)$ is used to approximate $I(d, G(z, d))$:

$$I(d, G(z, d)) = H(d) - H(d \mid G(z, d)) = H(d) + \mathbb{E}_{x^s \sim G(z, d)} \mathbb{E}_{d \sim p_{d}(d \mid x^s)} \log p(d \mid x^s) = H(d) + \mathbb{E}_{x^s \sim G(z, d)} \left[ D_d(p(\cdot \mid x^s)||q(\cdot \mid x^s)) + \mathbb{E}_{d \sim p_{d}(d \mid x^s)} \log q(d \mid x^s) \right] \geq 0 = H(d) + \mathbb{E}_{x^s \sim G(z, d)} \mathbb{E}_{d \sim p_{d}(d \mid x^s)} \log q(d \mid x^s) = H(d) + \mathbb{E}_{d \sim p_d, x^s \sim G(z, d)} \log q(d \mid x^s) = \mathcal{L}_{d}(G, E).$$  \hfill (2)

In fact, $\mathcal{L}_{d}(G, E)$ can be computed by $G$ and $E$:

$$\max_{G,E} \mathcal{L}_{d}(G, E) = H(d) + \mathbb{E}_{d \sim p_{d}, x \sim p_{x \mid d}} \log [E(G(z, d))],$$  \hfill (3)

where, $H(d)$ is the entropy of secret data, which has a constant value. $D_d(\cdot)$ is the KL divergence. $d \sim p_d$ means sampling secret data $d$ from distribution $p_d \sim \mathcal{N}(n, 0.5)$. $z \sim p_z$ means sampling latent $z$ from normal distribution $p_z \sim \mathcal{N}(0, 1)$. $x^s \sim G(z, d)$ means sampling stego images. $q(d \mid x^s)$ is an auxiliary distribution used to approximate the true posterior probability $p(d \mid G(z, d))$. $\mathcal{L}_{d}(G, E)$ can be calculated with the Monte Carlo simulation: randomly sample a secret tensor $d$ and a noise tensor $z$ to synthesize a stego image $x^s$ using the generator $G$, and then extract the hidden secret with extractor $E$. Both $G$ and $E$ hope to maximize the lower bound. $E(G(\cdot))$ means extracting the hidden secret from generated stego image, the values of which fall in $(0, 1)$ after a Sigmoid operation. Only when the extracted secret is equal to the input binary secret data $d$, the maximum value of $\mathcal{L}_{d}(G, E)$ is reached, and thus mutual information $I(d, G(z, d))$ is maximized.

In our scheme, $G$ can synthesize cover image ($x^c \equiv G(z, n)$) and stego image ($x^s \equiv G(z, d)$). A steganalysis algorithm SR-net[3] is used as the backbone of steganalyzer $S$, which aims to minimize the statistical difference between generated cover/stego images. The adversarial loss between $G$ and $S$ is written as $\mathcal{L}_{S}(S, G)$. Different to $\mathcal{L}_{d}(G, E)$, $S$ hopes to output the correct predictions ($[0,1]$ or $[1,0]$) with an binary cross-entropy loss, while $G$ wants $S$ to output $[0.5, 0.5]$ for both cover and stego images, as described in Eq.7 and Eq.10. Both $G$ and $S$ aim to minimize these two losses.

To generate realistic stego images with high secret extraction rate and good undetectability, we combine the loss functions above and set the overall optimization object as:

$$\min_{G,S,E} \max_{D} \mathcal{L}_{total}(D, G, S, E) = \mathcal{L}_{adv}(D, G) - \lambda \cdot \mathcal{L}_{d}(G, E) + \beta \cdot \mathcal{L}_{S}(S, G),$$  \hfill (4)

here, both $G$, $S$ and $E$ expect to minimize $\mathcal{L}_{total}(D, G, S, E)$, while $D$ wants to maximize it. $\lambda$ and $\beta$ are two hyper-parameters.

3.2 Loss Functions

In this section, we decompose $\mathcal{L}_{total}(D, G, S, E)$ into specific loss functions for each sub-network.

Generator’s loss The loss of generator takes two adversarial training processes and a regularization item into consideration:

$$\text{Loss}_G = \mathcal{L}_{adv} + \lambda_1 \cdot \mathcal{L}_{steg} + \lambda_2 \cdot \mathcal{R}_{PL},$$  \hfill (5)
To effectively train our GSN, sub-networks $G$, $D$, $S$ and $E$ are optimized sequentially as illustrated in Algorithm 1. $G$ and $E$ are optimized simultaneously to improve the secret extraction accuracy, where a hierarchical gradient decay (HGD) skill will be introduced in Sec. 3.7) is applied to improve the resistance against steganalysis methods. We optimize $G$ with $\text{Loss}_G$ and $\text{Loss}_E$ separately, mainly to decrease the differences between generated cover/stego images. The real images are only used to train $D$.

### 3.3 Training strategy

To effectively train our GSN, sub-networks $G$, $D$, $S$ and $E$ are optimized sequentially as illustrated in Algorithm 1. $G$ and $E$ are optimized simultaneously to improve the secret extraction accuracy, where a hierarchical gradient decay (HGD) skill will be introduced in Sec. 3.7) is applied to improve the resistance against steganalysis methods. We optimize $G$ with $\text{Loss}_G$ and $\text{Loss}_E$ separately, mainly to decrease the differences between generated cover/stego images. The real images are only used to train $D$.

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p_x} \log [1 - D(G(z, d))] + \mathbb{E}_{x \sim p_x} \log D(x),$$

$$\mathcal{L}_{ste} = \mathbb{E}_{x \sim p_x, n \sim p_n} \left[0.5 - S(G(z, n)) \right]^2 + \left[0.5 - S(G(z, d)) \right]^2,$$

$$\mathcal{R}_{PL} = \mathbb{E}_{z, \omega \sim N(0, 1)} \left\| \mathcal{J}_\omega \right\| - \eta,$$

here, $\mathcal{L}_{adv}$ is the adversarial loss between $D$ and $G$. $\mathcal{L}_{ste}$ is the adversarial loss of $G$ against $S$, which ensures the outputs of $S$ are close to 0.5 for both cover and stego images (i.e., $S$ could not distinguish the source of images). $\mathcal{R}_{PL}$ is a regularization loss used in [15] to improve the training stability and disentangle the dlatents space. $\lambda_1$ and $\lambda_2$ are two hyper parameters; $p_x$, $p_n$ and $p_d$ refer to the distributions of latent, noise and real image; $\psi$ is a randomly generated noise image with normal distribution, and $\omega$ is a dlatents variable as shown in Fig.3; $\|D(x)\|^2$ is the Jacobian matrix of $\psi$ with respect to $\omega$, and $\eta$ is a weighted value calculated with mean $\|F\|^2$. $\| \cdot \|_2$ means L-2 norm.

#### Discriminator’s loss

Discriminator’s loss is defined as:

$$\text{Loss}_D = -\mathcal{L}_{adv} + \alpha \cdot \mathcal{R}_1,$$

$$\mathcal{R}_1 = \frac{\beta}{2} \mathbb{E}_{x \sim p_x} \left\| \mathcal{D}(x) \right\|^2,$$

here, $\mathcal{L}_{adv}$ is the adversarial loss of Eq.5. $\alpha$ is a hyper parameter and $\mathcal{R}_1$ is the R1 regularization item given by [21]; $\|D(x)\|^2$ means the squared gradient of discriminator’s output $D(x)$ with respect to input real image $x$, and $\beta$ is a constant.

#### Steganalyzer’s loss

We adapt the binary cross-entropy loss in steganalyzer ($S$). $S$ outputs a two-dimension vector rather than a scalar like GANs, which is trained to output the correct predictions([0, 1] or [1, 0]) for input cover/stego images.

$$\text{Loss}_S = -\mathbb{E}_{x \sim p_x, n \sim p_n} \left[y_1 \cdot \log S(G(z, n)) + y_2 \cdot \log S(G(z, d))\right],$$

where, $y_1, y_2$ refers to the ground truth of cover/stego image.

#### Secret extraction loss

The loss of extracting hidden secret is computed by the binary cross entropy, with the prediction result $F$ (as shown in Fig.4), added noise and the input binary data $d$:

$$\text{Loss}_E = -\mathbb{E}_{x \sim p_x, d \sim p_d} \cdot \text{log}[E(G(z, d) + \text{noise})] = -\mathbb{E}_{x \sim p_x, d \sim p_d} \cdot \text{log}[\text{Sigmoid}(F + \text{noise})]$$

where, $p_d$ denotes the Bernoulli distribution of binary secret data. $\text{Sigmoid}()$ force the results to fall into (0, 1). Here, we add random noise ($\sim N(0, 0.01)$) to generated stego images for improving the robustness. Both $G$ and $E$ are optimized to minimize this loss.

#### 3.4 Structure of Generator

To facilitate covert communication, we propose a flexible image generator that can produce either cover or stego images. Fig.3 illustrates the architecture of proposed generator. It is improved based on stylegan2 [15], which consists of a mapping network and a synthesis network. The former network maps the input latent vector $z$ into an intermediate dlatents vector $\omega$ using eight fully connected (FC) layers. Next, $\omega$ is passed through a module called weight demodulation, which controls the image style. The synthesis network contains several general blocks and a newly designed secret block. The input of $G$ is a 512×4×4 trainable tensor initialized with normal distribution. Higher resolution feature maps can be

---

Figure 3: The architecture of proposed generator. The generator can synthesize either cover or stego image, when secret block is input with n or d, respectively.
We newly designed a data extractor to extract the hidden secret data from generated stego images. As shown in Fig.4, it contains several data extraction blocks, two 1×1 convolution layers, and a binarization operation. Each data extraction block includes some convolution/Lrelu operations. After $n$ (is set to 3 in our scheme) generated after up-sampling. The general blocks are made up of an upsample operation and several convolution layers. Noise matrix $N (\sim N(0, 1), \text{shape: } 1 \times h \times w)$ is added to the feature maps.

We design a module called secret block to enable the generator to synthesize cover and stego images, as shown in Fig.3. It contains three 1×1 convolution layers (followed by a LeakyRelu operation), three 3×3 convolution layers (with LeakyRelu), two upsample operations and four data merging operations (denoted as red @) and a low pass filter. The weight demodulation module regulates each convolutional layer to adjust the image content. Here, we add a noise matrix $n (\sim N(0, \sigma), \text{shape: } 1 \times H \times W)$ into feature maps to generate cover images as original stylegan2, and add a three-dimension secret matrix $d (\sim B(n, 0.5), B \times H \times W)$ to generate stego images. The generator can synthesize common cover images without secret as stylegan2, or it can be used to generate stego images. As a result, a cover image without secret data is generated.

The function of the secret block lies on two aspects: 1) it enables the generator to synthesize cover or stego images when noise or secret data is input; 2) it removes the defects on image contents and improves the quality of stego images. The input secret data can be written as $d = \{0, 1\}^{B \times H \times W}$. Here $H/W$ is the height/width of $d$, which is the same size as output stego images. $B$ refers to the channel number, which determines the payload of generated stego images. In the secret block, convolutional feature maps of each layer can be expressed as $F = \{f_i\}_{i=1}^N$ where $N$ denotes the total channel number of feature maps. The data merging operation @ aims to add $d$ to feature maps $F$, which can be described as:

$$f'_i = f_i + p \cdot d_j,$$

where, $d_j$ is the $j$th ($j = 1, 2, \ldots, B$) channel of the input secret data $d$, $f_i$ is the $i$th ($i = 1, 2, \ldots, N$) channel of feature maps, and $p$ is a parameter automatically learned to adjust the strength of merging (see Fig.3 p). “+” denotes the mathematical addition operation. $d_j$ and $f_i$ are of the same height and width with output stego images. The subscript $i = j + kB, i \in [1, N], j \in [1, B]$ and $k$ is a non-negative integer.

When random noise $n$ is input, the data merging operation shrinks to pixel-wise addition between $n$ and each feature map $f_i$. As a result, a cover image without secret data is generated. Secret data $d$ and noise $n$ may introduce mosaics or defects into the generated images. To mitigate their impacts, we adapt a low pass filter in the secret block:

$$filter = \frac{1}{64} \cdot [1, 3, 3, 1]^T \times [1, 3, 3, 1],$$

here, $\times$ denotes the mathematical matrix multiplication.

### 3.5 Structure of Extractor

We newly designed a data extractor $E$ to extract the hidden secret data from generated stego images. As shown in Fig.4, it contains several data extraction blocks, two 1×1 convolution layers, and a binarization operation. Each data extraction block includes some convolution/Lrelu operations. After $n$ (is set to 3 in our scheme)

Figure 4: The architecture of proposed data extractor.
We evaluate our GSN with metrics Acc, Fid and Pe. Different secret payloads are obtained by changing the size of input secret data d (B x H x W), i.e., vary the value of B (1 to 8) with H and W fixed, as shown in Fig.3. Different GSN models have been trained from scratch with the same setting apart from the payloads and datasets. Then the three metrics are tested with each trained GSN model. Their best performances are recorded in Table 1. The PE values are obtained by SR-net[3] trained individually in each configuration.

As shown in table 1, secret data’s extraction accuracy (Acc) decreases gradually as the payload increases. When the payload is 1 bpp, the Acc values are over 97% for both datasets. We can employ error correction codes further to increase the Acc values in real applications. Our scheme achieves excellent undetectability regardless of payloads, where the PE values are close to optimal 0.5. Fig.5 and Fig.6 give some stego image samples. These stego images look real and are hard to be distinguished from real ones.

### Table 1: The performance of proposed GSN.

| Datasets | bpp (B) | 1     | 2     | 4     | 6     | 8     |
|----------|--------|-------|-------|-------|-------|-------|
| CelebA 128×128 | Acc (%) ↑ | 97.53 | 81.61 | 70.14 | 61.15 | 59.28 |
|           | Fid ↓  | 13.29 | 15.17 | 16.21 | 16.83 | 18.16 |
|           | Pe → 0.5 | 0.479 | 0.499 | 0.501 | 0.502 | 0.498 |
| Bedroom 256×256 | Acc (%) ↑ | 97.25 | 83.19 | 72.13 | 64.17 | 60.94 |
|           | Fid ↓  | 15.21 | 14.56 | 15.77 | 16.89 | 18.80 |
|           | Pe → 0.5 | 0.500 | 0.499 | 0.502 | 0.488 | 0.499 |

4 EXPERIMENTAL RESULTS

Our GSN is trained on TensorFlow 1.14 with four Nvidia 1080Ti GPU. It is evaluated on datasets CelebA[19] and Lsun-bedroom[41]. Adam is used as the optimizer. We set λ₁ = λ₂ = 1 for LossG in Eq.5; α = 1, β = 10 for LossD in Eq.9; δ = 10 for the HGD skill in Eq.15. Input noise’s distribution is n ~ N(0, σ), where σ is set to 1 for training and 0.1 for testing.

Frechet inception distance (Fid) [11], extraction accuracy (Acc) and detection error (Pe) are used to evaluate the visual quality of generated stego images, secret extraction accuracy and the security of our work, respectively. Lower Fid means better image quality.

Acc is calculated as: Acc = \( \frac{d \cdot d'}{\text{len} \cdot \text{len}'} \) where, d and d’ are the input and extracted secret data, ⊕ is the element-wise XNOR operation. Pe is a common indicator to evaluate the undetectability of stego images, which is defined as: Pe = min(PFA + 1/2 PMD), where, PFA and PMD are the false alarm rate and missed detection rate. Pe ranges in [0, 1], and its optimal value is 0.5. When Pe is equal to 0.5, the steganalysis tool cannot distinguish the source of images.

All our generated images are saved in PNG format. If the length of input secret data is less than 1 bpp, we can add zeros after it.
Figure 7: ROC curves of different steganalysis algorithms.

figure), and the optimal AUC value is 0.5. As we can see, our ROC curves and AUC values are close to the ideal results, which indicates our proposed GSN is very safe.

4.3 Influence of Different Inputs on Stego Image

Our stego images are generated with two inputs, i.e., latent vector \( z \) and secret data \( d \) (see Fig.2). In this section, we analyze their impacts on the generated stego images.

**Influence of secret \( d \)**

We generate two stego images (payload: 1 bpp) with different \( d \) but the same \( z \), using a well-trained GSN model. As shown in Fig.8, when \( d \) is varied, these two stego images only differ in image details, such as texture and edges.

![Figure 8: Two stego images generated with different \( d \) but the same \( z \). The right two images show their colored and gray-scaled differences.](image)

**Influence of latent \( z \)**

Fig.9 gives four stego images (1 bpp) generated with different \( z \) but the same \( d \). By modifying the input latent vector \( z \), the appearances of stego images are changed drastically.

![Figure 9: Four stego images containing the same secret \( d \).](image)

4.4 Ablation Study

We apply an ablation study to demonstrate the effectiveness of proposed modules and techniques, i.e., the secret block, steganalyzer, and hierarchical gradient decay (HGD) skill. We rebuild a new GSN model without the low pass filter, steganalyzer and HGD skill, named baseline. To verify the function of the secret block, we cancel the low pass filter (defined in Eq.13) in it. Then these three modules/techniques are added to the baseline one by one cumulatively, and the performance of each new model is demonstrated in Tab.2. All the results are acquired from 128×128 stego images (1 bpp) on CelebA, and the Pe values are detected by SR-net[3].

By incorporating the low pass filter in the secret block, defects on images are removed, and image quality is enhanced significantly. Fig.10 further demonstrates the image comparison. Adding the steganalyzer will slightly improve the undetectability of stego images with higher Pe. However, it still cannot resist steganalysis detection. The HGD skill improves the Pe value drastically to 0.479 (the ideal value is 0.5), at the cost of reducing Fid and Acc slightly. Fig.11 compares the differences of cover/stego image pairs before and after using this skill. As we can see, the mean absolute error decreases from 6.32 to 1.41 after employing the HGD skill. Meanwhile, the differences are more randomly distributed in image contents.

![Figure 10: Stego images without (left) / with (right) the low pass filter in secret block.](image)

![Figure 11: Differences between the generated cover/stego images before (top) and after (bottom) using the hierarchical gradient decay skill. The mean absolute error of cover/stego image pairs is 6.32 (top) and 1.41 (bottom).](image)

4.5 Comparison with State-of-the-art

We compare our work with two types of steganographic methods:

1) Traditional steganography (TS) schemes based on deep learning

| Configurations | CelebA, 128×128 |
|----------------|----------------|
|                | Fid ↓ | Acc (%) ↑ | Pe → 0.5 |
| Baseline       | 19.45 | 99.23     | 0        |
| + Low Pass Filter | 12.71 | 99.56     | 0        |
| + Steganalyzer  | 13.03 | 99.67     | 0.035    |
| + Hierarchical gradient decay | 13.29 | 97.53     | 0.479    |
We compare GSN with two SOTA tailored GS schemes as shown in Table 3. This work was supported by the National Natural Science Foundation of China (U20B205, U20A20178, U19A20178, U19A20178, U20A20178, U19A20178).
