Hiding Secret in Color: Deep Colorizing Steganography via Conditional Invertible Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep steganography is a data hiding technology via neural networks. However, existing work can't control semantic content during transmission, and can't extract messages accurately when images are saved as integer format after rounding operations. To control the generated image content, this paper proposes steg-Cinn, a deep steganography via conditional invertible network, which can hide messages during colorization guided by gray-scale images. To achieve accurate extraction after integer format storage, this paper proposes an alternative training scheme, where embedder and extractor are updated asynchronously with reconstruction loss. Extensive experiments are conducted to evaluate the effectiveness of the proposed method. For visual quality, the colored images generated by steg-Cinn is controllable: colors can match semantic content in gray-scale images. For bit accuracy in message extraction after integer format storage, steg-Cinn with alternative training is about 7% higher than that without alternative training. Combined with error correcting code, it can be extracted accurately.

1 Introduction

Steganography is a data hiding technology for concealing the existence of communication, in which images carried with secret are often called **stego** and the corresponding images without secret are called **cover**. Traditional methods usually use adaptive encoding according to distortion cost designed by human [1, 2, 3, 4, 5, 6] or DNN [7, 8, 9, 10], which require much expert knowledge.

To get rid of complex artificial rules, researchers use neural networks to design steganography, which is called deep steganography. For deep steganography, the early exploration is based on embedding, which means that a real image is used as cover and secret is embedded into cover to get stego. These works generally adopt an encoder-decoder structure, and have been successfully applied to hiding images within images [11, 12, 13] and hiding binary bits in images [14, 15]. They perform well in visual quality, but can't guarantee accurate extraction which is the basic principle of steganography.

To guarantee accurate extraction, researchers begin to apply invertible neural network(INN) into deep steganography[16, 17, 18], since the invertible structure is naturally suitable for steganography. That's to say, it can be regarded as not only embedder but also extractor, thus bringing convenience to accurate extraction. However, existing work for deep image steganography based on INN[16] suffer two problems. (1)The image content isn't controllable, thus resulting in security issues in behavior. (2)There are errors in extraction after integer format storage such as "bmp" and "png". Only when images are saved as float format such as "tiff", "exr" and "raw", accurate extraction is achieved.

Considering two problems above, the contributions of this paper are summarized as follows.

- To control the generated image content, this paper proposes steg-Cinn, a deep steganography via conditional invertible network, which can hide messages during colorization guided by gray-scale images.
- To achieve accurate extraction after integer format storage, this paper proposes an alternative training scheme, where embedder and extractor are updated asynchronously with reconstruction loss.
- As far as we know, we are the first to apply conditional invertible network into deep image steganography and this novel method can bring vitality to data hiding community.

2 Background

2.1 Deep steganography

For steganography, the general approach is to use a real image as cover and embeds a secret in it to get stego. Traditional methods use adaptive encoding according to distortion cost designed by human [1, 2, 3, 4, 5, 6] or neural networks [7, 8, 9, 10], which require much expert knowledge. To get rid of complex artificial rules, researchers use encoder-decoder networks [11, 12, 13] or adversarial networks(encoder-decoder with a discriminator) [14, 15] to hide data, which is called deep steganography. They show great success in visual quality, but can't guarantee accurate extraction. It is because some works in deep steganography aim to hide secret images within images[11, 12, 13]: as long as the revealed secret image looks good visually, the limit of accurate extraction can be relaxed. However, accurate extraction is the basic principle of steganography, so their approach violates the principle and isn't rigorous. Another works in deep steganography aim to hide binary messages in images[14, 15] and can't guarantee accurate extraction either, though they use message reconstruction loss. Given the analysis above, the common problem of deep steganography is that the secret can't be extracted accurately. To solve this problem, researchers begin to apply invertible neural network(INN) into deep steganography, since the invertible structure is naturally suitable for steganography.

INN based deep steganography is a way of hiding secret during data generation without real images. If INN uses noise(latent variable) z as the driving force to synthesize image x, then x is cover; if the noise is mapped with information(denoted as z'), then x is stego. For INN, the state-of-the-art is Glow[19]. In Glow based steganography[16], the problem is that it can't control the image content thus the transmission behavior isn't secure. Inspired by conditional invertible network(cINN) that has good performance in controllable image generation[20], this paper introduces cINN into image steganography to make transmission behavior controllable.

2.2 Conditional invertible network(cINN)

The idea of conditional invertible network(cINN) [20] is derived from INN(also known as "flow based models" [21, 22, 19]), which is to represent the complex distribution $q(x)$ of real-world data x (e.g. images) with a simple distribution of the latent variable z (e.g. the standard normal distribution). Steps in inference are: 1) sample $z \sim \pi(z)$. where $\pi(z) = N(z; 0, I)$; 2) generate $x = f^{-1}(z; c)$, where f refers to mapping by cINN and c refers to condition(gray image in this paper). Note that f^{-1} or f consists of a series of invertible transformations f_i. For instance, $z = f(x; c)$ can be written as $z = f_k \circ f_{k-1} \cdots f_2 \circ f_1(x; c)$, where f_i is designed as affine coupling layer (ACL).

The framework of cINN is shown in Fig. 1. If f is composed of a single f_i, the transformations used for $z = f(x; c)$ are listed in Eq. (1), and those for $x = f^{-1}(z; c)$ are listed in Eq. (2), where s and t are arbitrary DNNs, $shuf()$ means to shuffle along the channel dimension, $split()$ means to split the input into two halves, $cat()$ means concatenation into one tensor, and $\odot$ refers to element-wise multiplication.

$$\begin{aligned} x_1, x_2 &= split(x) \\ z_1 &= x_1 \\ x_1 &= cat(x_1, c) \\ z_2 &= x_2 \odot s(x_1) + t(x_1) \\ z &= shuf(cat(z_1, z_2)) \end{aligned} \tag{1}$$

$$\begin{aligned} z_1, z_2 &= split(z) \\ x_1 &= z_1 \\ x_1 &= cat(x_1, c) \\ x_2 &= (z_2 - t(x_1))/s(x_1) \\ x &= shuf(cat(x_1, x_2)) \end{aligned} \tag{2}$$

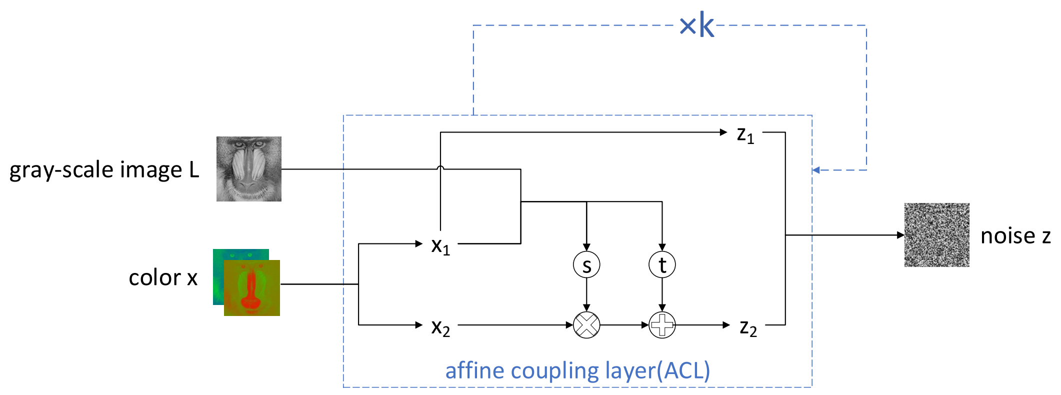


Figure 1: Framework of conditional invertible network(cINN) consisting k affine coupling layers.

The training of cINN is essentially maximum likelihood training. The training objective is to make $z = f(x; c)$ close to $\pi(z)$, where x is the input image from training set. Since cINN is invertible, as long as $z = f(x; c)$ is close to $\pi(z)$, $x = f^{-1}(z; c)$ will be close to $q(x)$, where $q(x)$ is the distribution of real-world data x (related to the training set). According to the idea of maximum likelihood estimation (MLE), the training loss L is the negative log-likelihood (NLL), see Eq.(3), where $\left|det\frac{\partial z}{\partial x}\right|$ is determinant of jacobi matrix. For convenience, in this paper, if any models whose training ideas derive from MLE, their loss is denoted as **NLL(z)**. Here z is model output.

The applications of cINN are widely used in guided image generation[20] and text-to-speech based audio generation[23]. For image steganography based on cINN, there are no explorations yet. For audio steganography based on cINN, there have been two works[18, 17] and the audio quality is good. However, two works suffer rounding error, that's to say, they only work when audios are saved as float format: when audios are saved as integer format, hidden data can't be extracted correctly.

3 Proposed methods

Based on the analysis above, to make steganography behavior controllable, this paper proposes steg-Cinn, a deep colorizing steganography that introduces conditional invertible network(cINN) into image steganography for the first time. To achieve accurate extraction suffering rounding error in image storage, this paper proposes an alternative training scheme with reconstruction loss.

3.1 Framework of steg-Cinn

The framework of proposed steg-Cinn is shown as Fig.2, which include hiding process and revealing process.

For hiding process, a sender uses secret message **m** (binary bitstream) and gray-scale image **L** (condition) as input to get **stego** (RGB image). First, **m** is transformed into a latent variable **z**(which obeys standard normal distribution) by a mapping module M. Under the guidance of **L**, another mapping module $cINN^{-1}$ converts **z** into color information **ab**. Next, **L** and **ab** are concatenated and converted to RGB image (floating point number). After rouning operation, the image can be sent as **stego**.

For revealing process, a receiver can extract secret as **m'** from stego. First, stego is converted from RGB to Lab color space to obtain **L'** and **ab'**, where **L'** represent reconstructed gray-scale information, and **ab'** represent reconstructed color information. Under the guidance of **L'**, mapping module $cINN$ maps **ab'** to **z'**. Using mapping module M, **m'** can be extracted from **z'**.

In both processes, two mapping modules are used: M and $cINN$.

For mapping module M, the pseudo code is shown in Algorithm. 1 and 2. It's main idea is to let the sign of **z** represent **m** (bit 1 or bit 0), where **m** is binary bit stream that obeys a uniform distribution, and **z** is latent variable that obeys a standard normal distribution. For error tolerance, an interval parameter α is defined so that z $\in [-\alpha,\alpha]$ is rejected when sampling **z**, where $\alpha \in [0,0.5]$. Therefore,

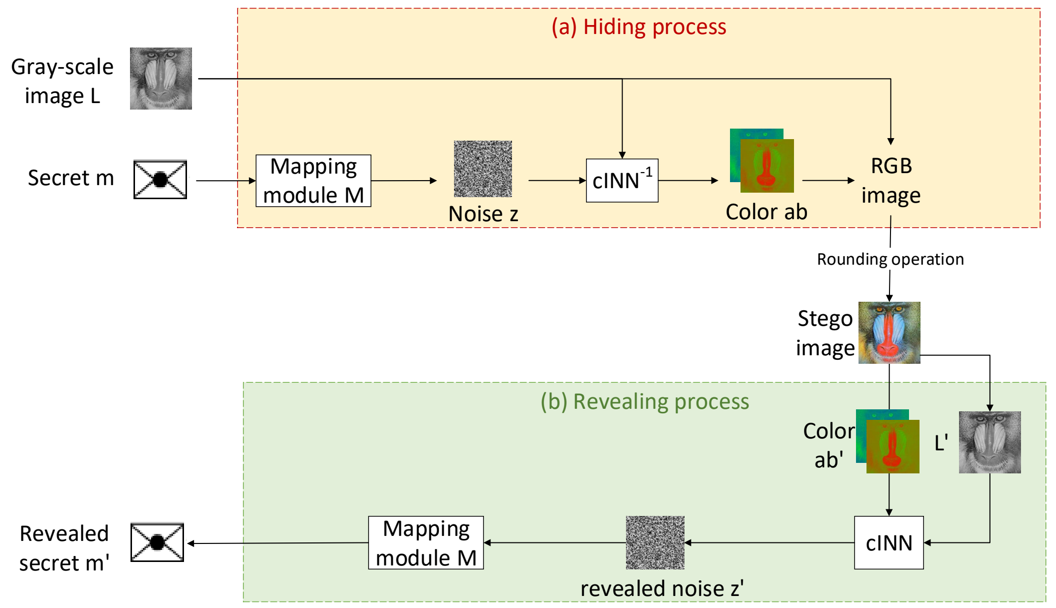


Figure 2: Proposed framework of deep colorizing steganography(DCS); (a) Hiding process; (b) Revealing process. Note that (a) and (b) both use mapping module M(shown in algorithm1-2) and conditional invertible network(cINN, shown in Fig.1).

in the hiding process, there is a "hole" with $2 * \alpha$ wide in the distribution of **z**, as shown in Fig 3 (a). Considering that the rounding operation in image storage will cause information loss, in the revealing process, the extracted **z'** may be as shown in Fig 3 (b). In this way, as long as the values of z and z' are close(as shown in Eq. (4)), **m** can be accurately extracted.

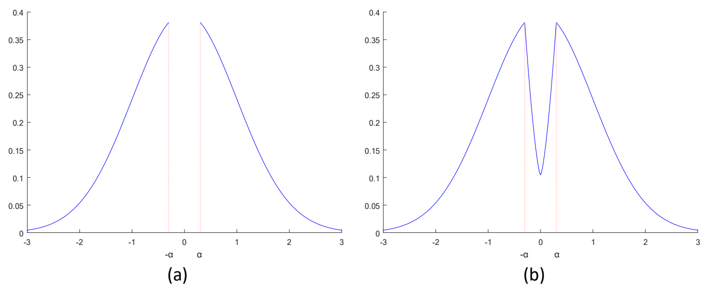


Figure 3: (a) Distribution of noise z in hiding; (b) Distribution of extracted noise z' in revealing.

Algorithm 1 Mapping module M in hiding

```
Input: secret msg={0,1,...,0,1}, a∈[0,0.5]
1: for all m in msg do
2:     if m==0 then
3:         Sample z from N(0,I) until $z < -\alpha$
4:     end if
5:     if m==1 then
6:         Sample z from N(0,I) until $z > \alpha$
7:     end if
8: end for
Output: latent variable z
```

Algorithm 2 Mapping module M^{-1} in revealing

```
Input: latent variable z
1: msg=[]
2: for all zz in z do
3:     if zz < 0 then
4:         msg.insert(0)
5:     else
6:         msg.insert(1)
7:     end if
8: end for
Output: extract msg
```

For mapping module $cINN$, the function is to do mapping: $\{L, z\} \to ab$, $\{L, ab\} \to z$. To make it reader-friendly, they are denoted as $x = f^{-1}(z; c)$ and $z = f(x; c)$, here $x = ab$, $c = L$. The framework of $cINN$ is shown in Fig. 1. Note that $cINN$ and $cINN^{-1}$ share the same architecture and parameters, but the data flow direction is different: the sender uses $cINN^{-1}$ in hiding while the receiver uses $cINN$ in revealing.

3.2 Training of steg-Cinn

As is mentioned above, existing deep steganography based on INN or cINN suffer rounding error: they can't extract messages accurately when images are saved as integer format. Therefore, this subsection will explain how to solve this problem through the training of steg-Cinn.

The training of steg-Cinn is divided into stage I and stage II. For stage I, maximum likelihood training is used to generate images with natural colors, which follows cINN. For stage II, alternative training with reconstruction loss is used to solve the problem of rounding error.

3.2.1 Stage I: maximum likelihood training

The training idea of stage I has followed INN, which is based on maximum likelihood method. Therefore, similarly, loss in stage I can be expressed as $loss_I = NLL(z)$, see Eq.(3). Here, L refers to gray image and ab refers to color information, the meaning of other symbols is explained in subsection 2.2.

$$\begin{aligned} loss_I &= NLL(z) = -\log q(x; L) \\ q(x; L) &= \pi(z) \left| det \frac{\partial z}{\partial x} \right| \\ z &= f(x; L) \\ x &= ab \end{aligned} \tag{3}$$

3.2.2 Stage II: alternative training

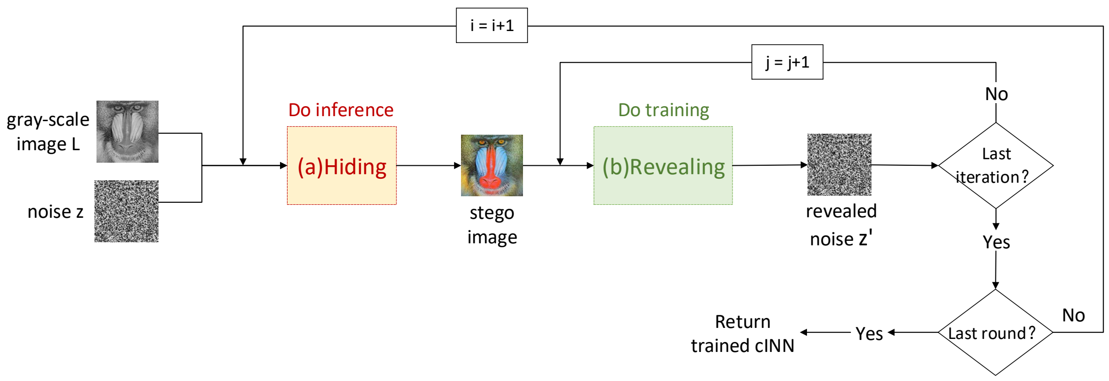


Figure 4: Proposed alternative training(stage II) of steg-Cinn. The idea is to train the "hiding + revealing" pipeline for i rounds (i ∈ [5,10]), and each round has j iterations (j ∈ [2000,3000]). In each round, the embedder $cINN^{-1}$ in hiding and the extractor $cINN$ in revealing are updated asynchronously: hide once (the embedder is updated once), and reveal j times (the extractor is updated j times). In fact, only the extractor does training. What the embedder does is inference, and its update comes from the copy of the extractor trained in the previous round. (a)Hiding process, see Fig.2(a) for details. (b)Revealing process, see Fig.2(b) for details.

Stage II is alternative training, and the overview is shown in Fig.4. (Note $cINN^{-1}$ and $cINN$ are both mapping modules mentioned in subsection 3.1. Here, we use term "embedder" and "extractor" to distinguish them in hiding and revealing.) From the perspective of the flowchart, stage II training can be regarded as a double loop, that is to say, the embedder $cINN^{-1}$ and the extractor $cINN$ are updated asynchronously. The hiding process is regarded as the entrance of the *outer loop*, and the revealing process is regarded as the *inner loop*. The *outer loop* does inference. Sample **z** from N(0,I), and **z** is fed to the embedder to generate an image (floating point). After the rounding operation, **img** (integer) is obtained. At this time, {**z**, **img**} is recorded in pairs. The *inner loop* does training. Using {**z**, **img**} as paired training data, the extractor is updated j iterations and copied to the embedder. Then go on next outer loop until the number of rounds runs out.

For training loss in stage II, $loss_{II}$, it can be written as Eq. (4), because only inner loop does training. The first item derives from the idea of INN. The second item is reconstruction loss, whose purpose is to improve the extraction rate under rounding errors. Through the lens of steganography, z is secret, and z'(z' is output of inner loop, as shown in Fig. 4) is revealed secret. Therefore, z and z' are expected to be close enough.

$$loss_{II} = NLL(z) + |z - z'|^2 \tag{4}$$

In addition, there are some details that may cause confusion, so explanations are made here. 1) Due to invertibility, the embedder and extractor are expected to be updated synchronously. However, for stability in training, they are updated asynchronously. In fact, the parameters of the embedder and the extractor are different at each moment of training; however, after training, their parameters are shared. 2) Since the mapping module M is designed by artificial rules, the training process only involves $cINN$, not M. 3) For alternative training, the settings are i=5 and α=0.1. Next section will analyze how these two factors affect the experimental results.

4 Experiment

To prove that the proposed deep colorizing steganography(DCS) framework outperforms existing deep embedded steganography(DES) and deep generative steganography(DGS), experiments in visual quality and statistical security are evaluated. To prove that the proposed alternative training scheme is effective, experiment on extraction rate under rounding operations is evaluated.

4.1 Dataset and baselines

Dataset For "DES", 10,000 training images and 5,000 testing images are chosen from Coco [24] dataset. For proposed "DCS", 10,000 training images are chosen from Coco [24] dataset and 5,000 testing images are chosen from BossBase [25] gray-scale dataset. They are all resized to a resolution of 128×128. For "DGS", 10,000 images are chosen from celebA [26] as training set, [1] which are resized to a resolution of 64×64 due to resource limitations. Since "DGS" derives from unconditional image generation which only use noise to synthesize images, the concept of testing set doesn't exist here.

Baselines The baselines used in this paper are steg-glow[16], ddh[11] and udh[13], in which steg-glow[16] belongs to "DGS",[2] ddh[11] and udh[13] belong to "DES".

4.2 Metrics

Capacity Capacity is measured with bit per pixel (bpp), which is the number of message bits hidden per pixel(RGB pixel) of stego.

Imperceptibility Imperceptibility measures the visual quality of stego. This paper uses two no-reference image quality assessment (NR-IQA) methods: Brisque [27] and hyperIQA [28]. Brisque and hyperIQA scores are usually in the range [0,100]. For Brisque, a lower score indicates better quality; for hyperIQA, a higher score indicates better quality. [3]

[1]The reason why "DGS" use celebA [26] instead of Coco[24] is expained here. If training set has more than one category, such as Coco[24], the images synthesized by trained model are confusing(can't be understood by human).

[2]Since "DGS" depends on generative models, this paper select a Glow[19] based work steg-glow[16], where Glow(an instance of INN) is known as a rising star in generative models.

[3]This paper uses NR-IQA instead of R-IQA(based on similarity). The reasons are as follows: "DES" has ground truth (GT), which is suitable for R-IQA; "DGS" doesn't have GT, which is suitable for NR-IQA. For "DCS", there are two cases. 1) When original color is unknown (no GT), R-IQA cannot be used. 2) When original color is known (with GT), the R-IQA is still not proper. Because this paper pursues that stego and GT are close in statistics rather than on a pixel-wise level. (Stego is expected to be natural in color, not exactly the same as GT). Therefore, this paper uses NR-IQA to fairly compare the stego quality synthesized by three classes.

Statistical security Statistical security is measured with detection rate, which represents the ability to evade the detection of steganalyzers. The lower detection rate is, the more secure the steganographic scheme is and the easier it is to evade detection. This paper chooses Ke-Net[29] as the steganalyzer, which is recognized as the state-of-the-art steganalysis method based on contrastive learning.

Extraction rate Extraction rate is measured with bit accuracy, which is the number of identical bits between the input message and the revealed message.

4.3 Experimental results

This subsection shows the performance of the proposed method comparing with baselines on four metrics mentioned previously. For completeness, how do influencing factors work on results are also analysed here.

4.3.1 Capacity

For a stego with a size of 3×N×N, the capacity of the proposed method and baselines is shown in Table 1. For steg-glow [16], the total dimension of the secret is the same as the total dimension of stego, so the capacity is 3 bpp. For "DCS", the total dimension of **ab** is 2×N×N, and the total dimension of the secret is the same as **ab**, so the capacity is 2 bpp. For ddh [11] or udh [13], each 3×1×1 patch hides one bit, so the capacity is 1 bpp.

Attentions. 1)Note that the original task of ddh[11] and udh[13] is to hide images in images, thus their capacity is 24 bpp as claimed. However, such task relaxes the constraint of perfect extraction and statistical security while focuses on a trade-off between visual quality and capacity, which don't strictly follow the basic rule of steganography. In this paper, the purpose is to hide bit streams in images. Therefore, for fairness, the original task of ddh[11] and udh[13] is converted to hiding a binary image in image. That's to say, use pixel 0 to embed bit 0 and pixel 255 to embed bit 1; extract bit 0 if the average value of each 3*1*1 patch is less than 128, otherwise extract bit 1. 2)Due to rounding operations used in image storage, extraction errors will occur. To achieve extraction rate up to 100 %, this paper uses error correcting codes. Therefore, the capacity of "DCS" will be lower than 2 bpp, which will be discussed in "Extraction rate" part.

Table 1: Capacity comparisons

Model	Secret	Capacity(bit per pixel)
ddh [11] or udh [13]	binary image	1
steg-glow [16]	bit stream	3
DCS(proposed)	bit stream	2

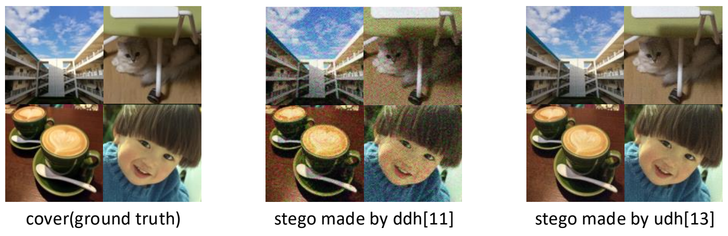
4.3.2 Imperceptibility

Imperceptibility measures the visual quality of stegos which are synthesized by three classes of steganography. For ease of understanding, before observing imperceptibility results, readers are advised to read Fig.**??** to review basic differences between three classes.

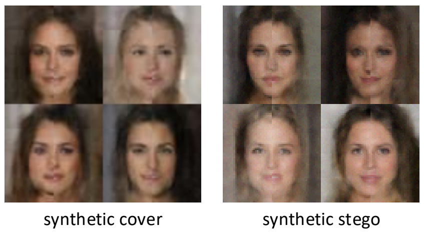
Results on imperceptibility are shown in Fig.5. For steg-glow[16], stego looks blurry in quality,[4] and cover and stego are same in category but not in content(not the same face of a person). For ddh[11] and udh[13], stego is required to be visually similar to cover(GT). For DCS(proposed), with condition(arbitrary grayscale image), stego synthesized by DCS looks natural in color. That's to say, data hiding is completed during colorization. Note that cover and stego are not exactly the same, it is because they are required to be close in distribution rather than on a pixel-wise level.

To objectively measure visual quality of three classes, two NR-IQA metrics are used here. In Table 2, it can be seen that DCS(proposed) shows competitive result. Under hyperIQA[28], he best one is DCS and it performs better than steg-glow[16] under both metrics, which is because the generation in DCS

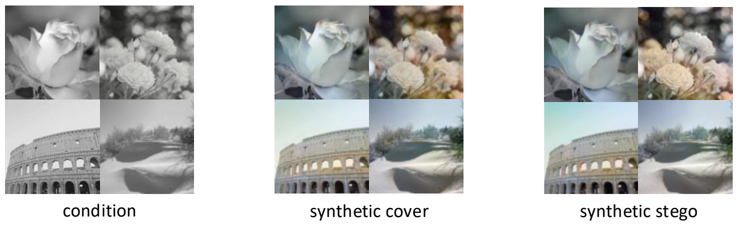
[4]Though Glow [19] get high quality in image generation, however, when applied to steganography, the visual quality isn't good.



(a)



(b)



(c)

Figure 5: Visual quality in three classes of steganography. (a)deep embedded steganography(DES); (b)deep generative steganography(DGS); (c)proposed deep colorizing steganography(DCS).

is guided by condition(gray-scale image) while the generation in steg-glow[16] is unconditional and is only driven by single noise. Under Brisque[27], the best one is udh[13] and it performs better than ddh[11] under both metrics, which is because the hiding for udh[13] is processed in a cover-agnostic manner.

4.3.3 Statistical security

For statistical security, steg-glow[16] and DCS(proposed) perform much better(detection rate around 50 % means random guess) than ddh[11] and udh[13](detection rate around 100 % means steganalyzer can accurately distinguish between cover and stego). It's because their goals are different. The steg-glow[16] and DCS(proposed) pursue that cover and stego are same in statistics: they can resist detection as long as cover and stego are driven by the similar distributed noise and synthesized by the same model. While ddh[11] and udh[13] pursues that cover and stego are same on a pixel-wise level instead of in statistics, thus they are easy to be detected, see Table 3.

Table 2: NR-IQA of three classes steganography.

Model	↓Brisque [27]	↑hyperIQA [28]
ddh [11]	49.9708	20.27
udh [13]	**18.3516**	36.11
steg-glow [16]	33.2320	27.31
DCS(proposed)	21.7449	**38.46**

Table 3: Detection rate with Ke-Net [29] of three classes steganography.

Model	↓Detection rate
ddh [11]	100.0 %
udh [13]	99.04 %
steg-glow [16]	57.09 %
DCS(proposed)	**56.28 %**

4.3.4 Extraction rate

Since the structure of steg-glow [16] or DCS(proposed) is invertible, secret can be recovered accurately (extraction rate is 100 %) under hypothesis of lossless channel, as it is shown in Table 4. In contrast, the structure of ddh [11] or udh [13] is based on feedforward networks (encoder-decoder), so there are errors in extraction (a remedy is sacrificing capacity to reduce errors). Note that the extraction rate of ddh is higher than that of udh, which may be because ddh is cover-dependent and udh is cover-agnostic. Due to practical scenario under rounding operations in image storage, there are decreases in performance, which is considered as common phenomenon. In this situation, DCS(proposed) achieves highest extracion rate as shown in Table 4.

Table 4: Extraction rate comparisons under two scenarios.

Model	Extraction rate under lossless channel	Extraction rate under rounding operation
ddh [11]	98.21 %	56.40 %
udh [13]	89.95 %	60.34 %
steg-glow [16]	**100.00 %**	87.83 %
DCS(proposed)	**100.00 %**	**96.04 %**

4.3.5 Analysis of factors affecting performance for DCS

Analysis of factor(α) affecting statistical security In Table 5, the factor affecting statistical security is α(the parameter of the mapping module M): as α increases, security performance decreases(detection rate goes up). This is because, the noise(noted as z_c) that drives the cover synthesis strictly obeys normal distribution, while the noise(noted as z_s) that drives the stego synthesis is a "hollow" normal distribution(use interval α to ensure error tolerance), as shown in Fig. 6. The greater difference between z_c and z_s, the greater difference between cover and stego in statistics, which makes it easier to be detected.

Table 5: Detection rate of "DCS" with Ke-Net [29] when α(parameter of mapping module M) differs.

Parameter α	0	0.1	0.15	0.2
Detection rate	51.09%	56.28%	64.02%	71.82%

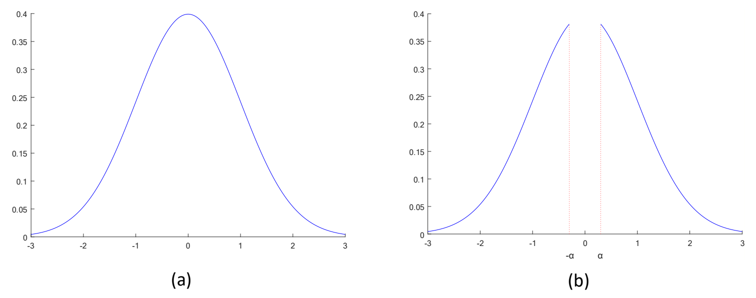


Figure 6: (a) Distribution of noise z_c which drives generating cover; (b) Distribution of noise z_s which drives generating stego.

Analysis of factors(i and α) affecting extraction rate The extraction rate is affected by two factors, one is i (the number of rounds of stage II training), and the other is α (the parameter of the mapping module M), see Table 6. When i gets larger, extraction rate gets higher, indicating that alternative training is effective. When α gets larger, extraction rate gets higher, which is because α indicates the error-tolerant interval of the driving noise.

Considering the model converges when i$\geq$5 and statistical security decreases(as previously mentioned) when $\alpha \geq 0.1$, the settings for stage II are i=5 and α=0.1. At this time, extraction rate is 96%, which hasn't reached 100%. Therefore, it is necessary to combine error correcting codes to achieve perfect extraction. Of course, it will cause a decrease in capacity, and the amount of decrease depends on the setting of error correcting code. For example, as long as the extraction rate exceeds 95%, the error correction[30, 31] can be successful. At this time, 5/12 of the capacity is sacrificed as redundancy. [5]

Table 6: Extraction rate for "DCS" under rounding operations.

Round of stage II (alternative training)	Extraction rate			
	α=0	α=0.1	α=0.15	α=0.2
i=0	86.05 %	88.83 %	89.96 %	91.11 %
i=1	88.69 %	91.60 %	92.84 %	93.88 %
i=2	90.27 %	93.15 %	94.37 %	95.36 %
i=3	91.73 %	94.56 %	95.74 %	96.45 %
i=4	92.43 %	95.44 %	96.39 %	97.04 %
i=5	93.08 %	**96.04 %**	96.86 %	97.39 %

5 Conclusion

Works in this paper include two folds. 1) First, a deep colorizing steganography based on cINN is proposed, which strikes a good balance between security and visual quality. In addition, proposed method also has competitive performance in capacity. Due to invertibility, secret can be accurately extracted under the lossless channel hypothesis. 2) Second, "alternative training" is used under practical scenarios (rounding operations used in image storage) to increase extraction rate. Combined with error correcting codes, extraction rate can be further increased to 100 %. For rigor, through factor analysis, the impacts of mapping parameter α on security and extraction rate are analyzed, thus giving advice to users on how to make trade-offs.

Future work include two aspects. 1) The color of synthetic images are conservative, so it is expected to increase the color richness in the future. 2)For practicality, more channel attacks will be considered, including but not limited to rounding operations: such as image compression, "non-original" transmission on social media, etc.

References

[1] Vojtech Holub and Jessica Fridrich. "Designing steganographic distortion using directional filters". In: *2012 IEEE International workshop on information forensics and security (WIFS)*. IEEE. 2012, pp. 234–239.

[2] Tomas Pevn'y, Tomas Filler, and Patrick Bas. "Using high-dimensional image models to perform highly undetectable steganography". In: *International Workshop on Information Hiding*. Springer. 2010, pp. 161–177.

[3] Bin Li et al. "A new cost function for spatial image steganography". In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 4206–4210.

[4] V. Holub, J. Fridrich, and Tomá Denemark. "Universal distortion function for steganography in an arbitrary domain". In: *EURASIP Journal on Information Security* 2014.1 (2014), pp. 1–13.

[5] Linjie Guo, Jiangqun Ni, and Yun Qing Shi. "Uniform embedding for efficient JPEG steganography". In: *IEEE transactions on Information Forensics and Security* 9.5 (2014), pp. 814–825.

[6] Linjie Guo et al. "Using statistical image model for JPEG steganography: Uniform embedding revisited". In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2669–2680.

[7] Weixuan Tang et al. "Automatic Steganographic Distortion Learning Using a Generative Adversarial Network". In: *IEEE Signal Process. Lett.* 24.10 (2017), pp. 1547–1551. DOI: 10.1109/LSP.2017.2745572.

[5]Strictly speaking, Stegastamp [31] belongs to LFM (light field messageing) rather than steganography. So, it is not fair to use it as a baseline.

[8] Jianhua Yang et al. “Spatial image steganography based on generative adversarial network”. In: *arXiv preprint arXiv:1804.07939* (2018).

[9] Jianhua Yang et al. “An embedding cost learning framework using GAN”. In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 839–851.

[10] Weixuan Tang et al. “An Automatic Cost Learning Framework for Image Steganography Using Deep Reinforcement Learning”. In: *IEEE Trans. Inf. Forensics Secur.* 16 (2021), pp. 952–967. DOI: `10.1109/TIFS.2020.3025438`.

[11] Shumeet Baluja. “Hiding images in plain sight: Deep steganography”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 2066–2076.

[12] Shumeet Baluja. “Hiding images within images”. In: *IEEE transactions on pattern analysis and machine intelligence* 42.7 (2019), pp. 1685–1697.

[13] Chaoning Zhang et al. “UDH: Universal deep hiding for steganography, watermarking, and light field messaging”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10223–10234.

[14] Jamie Hayes and George Danezis. “Generating steganographic images via adversarial training”. In: *NIPS*. 2017.

[15] Jiren Zhu et al. “HiDDeN: Hiding data with deep networks”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 657–672.

[16] Kejiang Chen et al. “When provably secure steganography meets generative models”. In: *arXiv preprint arXiv:1811.03732v2* (2018).

[17] Kejiang Chen et al. *Distribution-Preserving Steganography Based on Text-to-Speech Generative Models*. 2020. arXiv: `1811.03732v3 [cs.MM]`.

[18] Hyukryul Yang et al. “Hiding Video in Audio via Reversible Generative Models”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1100–1109.

[19] Diederik P Kingma and Prafulla Dhariwal. “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: *NeurIPS*. 2018.

[20] Lynton Ardizzone et al. “Guided image generation with conditional invertible neural networks”. In: *arXiv preprint arXiv:1907.02392* (2019).

[21] Laurent Dinh, David Krueger, and Yoshua Bengio. “Nice: Non-linear independent components estimation”. In: *arXiv preprint arXiv:1410.8516* (2014).

[22] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using Real NVP”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[23] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. “Waveglow: A Flow-based Generative Network for Speech Synthesis”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 3617–3621. DOI: `10.1109/ICASSP.2019.8683143`. URL: `https://doi.org/10.1109/ICASSP.2019.8683143`.

[24] T. Y. Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European Conference on Computer Vision*. 2014.

[25] P. Bas, T. Filler, and T. Pevn. “"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*. 2011.

[26] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[27] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. “No-reference image quality assessment in the spatial domain”. In: *IEEE Transactions on image processing* 21.12 (2012), pp. 4695–4708.

[28] Shaolin Su et al. “Blindly assess image quality in the wild guided by a self-adaptive hyper network”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3667–3676.

[29] Weike You, Hong Zhang, and Xianfeng Zhao. “A Siamese CNN for Image Steganalysis”. In: *IEEE Transactions on Information Forensics and Security* 16 (2020), pp. 291–306.

[30] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. “On a class of error correcting binary group codes”. In: *Information and control* 3.1 (1960), pp. 68–79.

[31] Matthew Tancik, Ben Mildenhall, and Ren Ng. “Stegastamp: Invisible hyperlinks in physical photographs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2117–2126.