Large-capacity Image Steganography Based on Invertible Neural Networks

Shao-Ping Lu[1*] Rong Wang[1*] Tao Zhong[1] Paul L. Rosin[2]
[1]TKLNDST, CS, Nankai University, Tianjin, China
[2]School of Computer Science & Informatics, Cardiff University, UK
slu@nankai.edu.cn; nkwangrong@163.com; zei.t@qq.com; RosinPL@cardiff.ac.uk

Abstract

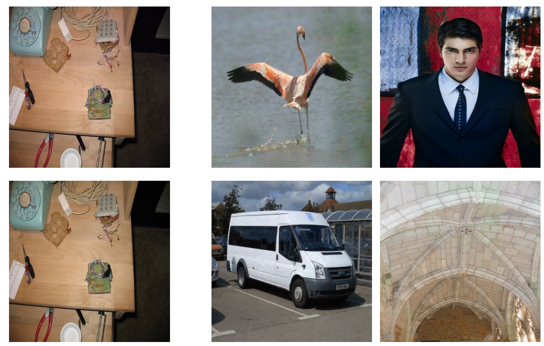
Many attempts have been made to hide information in images, where one main challenge is how to increase the payload capacity without the container image being detected as containing a message. In this paper, we propose a large-capacity Invertible Steganography Network (ISN) for image steganography. We take steganography and the recovery of hidden images as a pair of inverse problems on image domain transformation, and then introduce the forward and backward propagation operations of a single invertible network to leverage the image embedding and extracting problems. Sharing all parameters of our single ISN architecture enables us to efficiently generate both the container image and the revealed hidden image(s) with high quality. Moreover, in our architecture the capacity of image steganography is significantly improved by naturally increasing the number of channels of the hidden image branch. Comprehensive experiments demonstrate that with this significant improvement of the steganography payload capacity, our ISN achieves state-of-the-art in both visual and quantitative comparisons.

1. Introduction

Steganography is the art of hiding some secret data by embedding it into a host medium that is not secret. Different from cryptography which hides the meaning of the data (or makes it unintelligible), steganography aims to hide the existence of the data [11,42]. Accordingly, image steganography refers to the process of hiding data within an image file. The image chosen for hosting the hidden data named the *host-* or *cover-image*, and the image generated by steganography is called the *container-* or *stego-image*. Nowadays, image steganography is used in digital communication, copyright protection, information certification, e-commerce, and many other practical fields [11].

A well-designed image steganography system is expected to have both the imperceptibility and payload capacity requirements [33]. Firstly, the container image should avoid arousing suspicion. This means that the hidden data should not be detected under *steganalysis*, which is the countermeasure of steganography. As shown in Fig. 1, when the hidden images are embedded into the host image, if the generated container image appears similar to the host image in terms of its color and other features, then it would be difficult for image steganalysis techniques [18,24] to distinguish between the host and container images. Therefore, image steganography essentially asks for a powerful image representation mechanism that can effectively approximate the host image with the "noise" of the hidden images. This process is also expected to be reversible, because the hidden images should be well recovered from the container image in the decoding process of image steganography. Besides that, to make image steganography applications more efficient in practice, another important aspect is to embed as much hidden data as possible into the host image.

Existing image steganography solutions [8, 40, 62] still cannot perfectly simultaneously achieve good imperceptibility with high payload capacity. Traditional methods



(a) Host/container (b) 4 revealed images

Figure 1. We generate a container image by hiding 4 other images into the host image. Guess which is the container image in the left column? Answer: the top-left and bottom-left are the container and host images, respectively. (b): 4 hidden images revealed from the container image. These 6 images have the same resolution.

*indicates equal contribution.

usually hide messages in the spatial, transform, or some adaptive domains [33], with the payload capacities around $0.2 \sim 4$ bits per pixel (bpp). For most of them, the hidden data is embedded into the least significance bits (LSBs) [8] or insensitive areas that are detected with low-level vision descriptors, meaning that only a small amount of hidden information can be embedded. Several recent deep learning-based hiding methods [4, 5] successfully find a potential route to increase the hiding capacity. However, once the image steganography system consists of different neural networks that are separately designed for the preprocessing, steganography, and recovery tasks, the components of the whole system are independent of each other and the parameters are not shared. It would thus be difficult to find a trade-off between making the container statistically indistinguishable and recovering the high-quality hidden image.

In this paper, we introduce a large-capacity image steganography approach based on the invertible neural network (INN) [14,15,58]. We take the task of hiding an image as a special image domain transformation task, where the container image should be as close as possible to the host image. In its reverse process, the hidden images should also be well reconstructed from the container image. Therefore, we take image steganography and recovery as a pair of inverse problems, and thus introduce an *Invertible Steganography Network* (ISN) to effectively solve them. Our novel solution takes the same ISN for both steganography and recovery, where all the parameters are fully shared in such two tasks. This methodology enables us to efficiently generate both the container image and the revealed hidden image(s). Our ISN network consists of two branches, naturally corresponding to the input hidden and host images, respectively. Moreover, in our architecture, the steganography capacity can be substantially improved by increasing the number of channels of the hidden image branch. Comprehensive experiments demonstrate that our method can generate a desired container image with high payloads for hiding images, and with the same framework, we successfully reveal such multiple hidden images (Fig. 1 shows hiding 4 images).

In summary, the main contributions of this paper are:

- We introduce an *Invertible Steganography Network* (ISN) to effectively solve image steganography and recovery problems. Our bijective transformation model uses a single network to efficiently hide and reveal images.
- Our method significantly improves the steganography payload capacity to $24 \sim 120$ *bpp*, and it can be easily adapted to hide multiple images with high imperceptibility.
- A comprehensive set of qualitative and quantitative experiments show that our method achieves state-of-the-art steganography and recovery results.

2. Related Work

Image hiding has been extensively studied in the academic community [9, 33]. Here we briefly discuss some representative work on image steganography and the most relevant techniques on invertible neural networks.

Traditional image steganography methods. Image steganography techniques can be briefly classified into three types: spatial-based [8, 31, 36–38, 43, 52, 56], transform-based [19, 26, 41, 42, 45] and adaptive steganography methods [22, 23, 27–29, 35, 40]. A commonly used spatial steganography algorithm is the LSB steganography [8], where the information is embedded by modifying the LSBs of the host image. However, this leaves traces in the statistics of the container image that can be easily detected by some steganalysis methods [18, 24, 61]. Other spatial steganography methods are based on pixel value differencing (PVD) [38, 56], histogram shifting [43, 52], multiple bit-planes [31, 36], palettes [31, 37] and so on. Transform steganography applies image hiding in various transform domains [9, 33]. For instance, JSteg [42] embeds the data into the LSBs of the discrete cosine transform (DCT) coefficients of the host image. In general, DCT steganography techniques [19, 26, 41, 45] share a low steganography payload capacity.

Adaptive steganography normally adopts a general framework for data embedding, where the problem can be decomposed into embedding distortion minimization and data coding. A well-known framework of this class of method was proposed in [40], where the subtractive pixel adjacency matrix feature [39] and syndrome-trellis codes [17] are utilized for adaptive steganography. Similarly, some other adaptive methods [22, 23, 27–29, 35] are designed with different cost functions. These methods have good imperceptibility, but still with a common limitation in payload capacity.

Deep learning-based image steganography. Various deep learning-based image steganography schemes have been introduced recently. These methods can be categorized into four families [10]: the family by synthesis [46, 53], the family by generation of the modifications probability map [50,59], the family by adversarial-embedding [49] and the family by 3-player game [5, 25, 60, 62].

In the family of image synthesis, [46] and [53] both use generative adversarial networks (GANs) to create a more suitable container. Compared with traditional steganography, these methods have no significant improvement in the aspect of steganography payload capacity. In the family of modifications probability map generation, most methods focus on generating various cost functions satisfying minimal-distortion embedding [40]. In [50] a GAN-based distortion learning framework is introduced for steganography, while in [59] a generator with U-Net architecture is used to convert an input image into a container image. In the family of adversarial-embedding, [49] presents an adversarial scheme under the distortion minimization framework [40]. In the family 3-player game, HiDDeN [62] and SteganoGAN [60] adopt the encoder-decoder structure to perform information embedding and recovery. To resist steganalysis, they include a third network that plays the role of adversary.

Recently, an excellent approach called Deep Steganography [4, 5] successfully hides an image within another image of the same size. This method uses a fully convolutional network consisting of three components: the preparation, hiding, and revealing networks. These three different networks are trained in an end-to-end manner. In contrast, our method utilizes an invertible network to train all shared parameters of the hiding and revealing tasks.

Applications. Many steganography based applications have been proposed. For instance, Chen *et al.* [12] integrate image steganography into style transfer. Wengrowski *et al.* [55] introduce light field messaging (LFM) for message transmission using hiding, recovering, and distortion simulation networks. Tancik *et al.* [48] present a steganographic system called StegaStamp. This system could be applied to provide extra information in addition to perceivable image contents. Besides that, there is some interesting work [51] focusing on hiding some objects or textures by making them similar to the target image.

Invertible Neural Networks (INN). In recent years, the invertible neural network has attracted much attention, as it is one of the effective schemes for reversible image transformation. INN learns a stable invertible mapping between the data distribution p_X and a latent distribution p_Z. Instead of constructing a cycle loss to train two generators to implement bidirectional mapping such as in CycleGAN [63], INN involves the forward and back propagation operations in the same network, such that it realizes both the feature encoder and the image generator.

Pioneering research on INN-based mapping can be seen in NICE [14] and RealNVP [15]. In [20] a further explanation for the invertibility is explored. INNs have also been proved to share some advantages in estimating the posterior of an inverse problem [2]. In [47], flexible INNs are constructed with masked convolutions under some composition rules. An unbiased flow-based generative model is also introduced in [13]. Besides that, FFJORD [21], Glow [34], i-RevNet [32] and i-ResNet [6] further improve the coupling layer for density estimation, achieving better generation results. Because of the powerful network representation, INNs are also used for various inference tasks, such as image colorization [3], image rescaling [58], image compression [54], and video super-resolution [64]. We take the advantage of INN's bijective construction and efficient invertibility for our steganography issue.

3. Proposed Approach

3.1. Overview

Our image steganography framework aims to effectively embed multiple hidden images into the host image, and conversely, it enables us to reveal the hidden images with high-quality from the container image, as shown in Fig. 2 (b). Formally, we set the host image and the hidden image(s) as x_{ho} and x_{hi}, respectively, and the corresponding container image is y_{co}. As mentioned before, we regard embedding and extracting the hidden images as a pair of inverse problems, we thus formulate the procedure as:

$$\begin{aligned} &y_{co} = f(x_{hi}, x_{ho}), \\ &(\hat{x}_{ho}, \hat{x}_{hi}) = f^{-1}(y_{co}), \end{aligned} \tag{1}$$

where $\hat{x}_{ho}, \hat{x}_{hi}$ are respectively the recovered host and hidden images from the container image. Therefore, suitable optimizers should be designed to ensure that y_{co} and $\hat{x}_{hi}$ are as close as possible to x_{ho} and x_{hi}, respectively.

In our system, we introduce a single network to simultaneously perform feature transformation, image steganography and revealing. Therefore, we use the forward mapping of this network to fit the stenography function $f(\cdot)$, and the reverse mapping to fit the recovery function $f^{-1}(\cdot)$, as defined in Eq. (1). Specifically, on the forward propagation the host image x_{ho} and hidden images x_{hi} are set as input to get the container image y_{co}. On the back propagation, such y_{co} is set as input to reveal $\hat{x}_{hi}$. Our two tasks are processed in the same network, benefiting from all fully shared parameters of the two invertible propagation operations.

3.2. Invertible Steganography Network (ISN)

Our ISN architecture, where hiding and revealing images are efficiently solved in the same network, is inspired by the latest INNs [14, 15, 58]. As shown in Fig. 2 (b), our ISN consists of several invertible blocks. In INNs, the basic invertible coupling layer is the additive affine transformations proposed by NICE [14]. For this model, the l-th invertible block, the input b^l is divided into b_1^l and b_2^l along the channel axis and the corresponding output is b_1^{l+1} and b_2^{l+1}. For the forward operation,

$$\begin{aligned} b_1^{l+1} &= b_1^l + \phi(b_2^l), \\ b_2^{l+1} &= b_2^l + \eta(b_1^{l+1}), \end{aligned} \tag{2}$$

where $\phi(\cdot)$ and $\eta(\cdot)$ are arbitrary functions. For the backward operation, given [b_1^{l+1}, b_2^{l+1}], it is easy to calculate [b_1^l , b_2^l] as:

$$\begin{aligned} b_2^l &= b_2^{l+1} - \eta(b_1^{l+1}), \\ b_1^l &= b_1^{l+1} - \phi(b_2^l). \end{aligned} \tag{3}$$

For our image-into-image steganography, the forward propagation operation is to embed x_{hi} into x_{ho}, the input

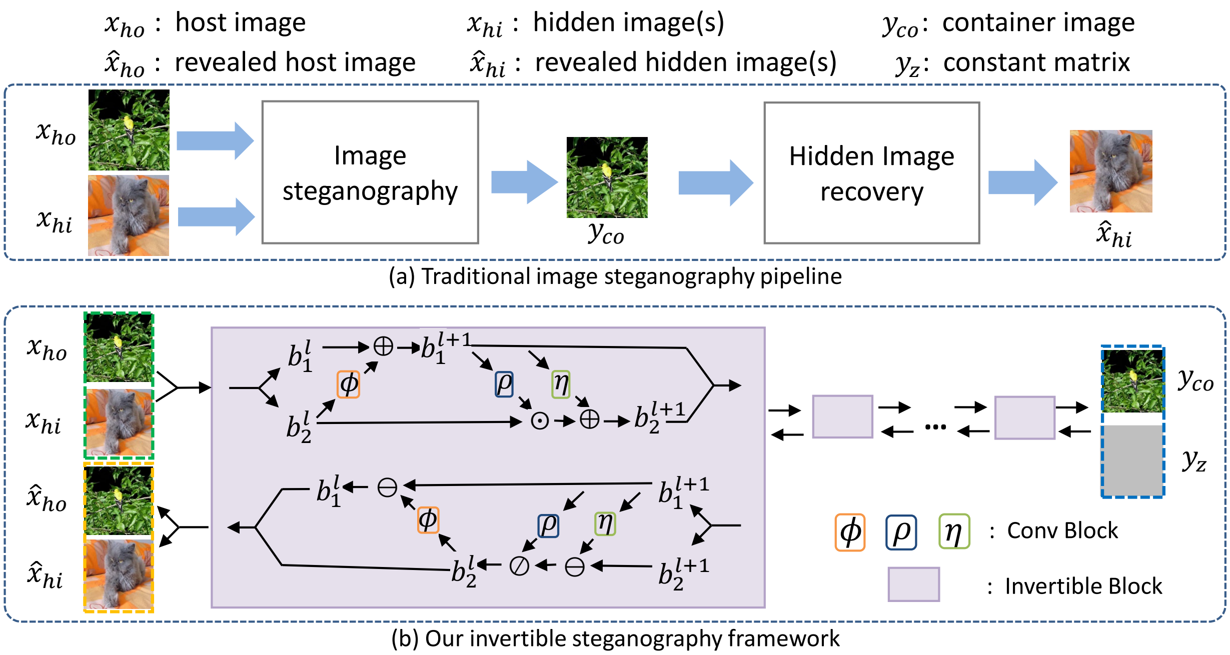


Figure 2. System pipeline. Unlike traditional methods (a) where steganography and recovery of the hidden image are processed separately, we introduce an invertible steganography framework (b). The multiple hidden images are concatenated with the host image, serving as a forward input to the trainable invertible network. The container image is then generated using several invertible blocks sharing the same structures. Conversely, the backpropagation effectively recovers the hidden images with high quality from the container image.

of our ISN naturally consists of two parts, which exactly match the splitting of b_1^l and b_2^l. To increase the representational capacity of the network, an affine coupling layer [15] is frequently used. Following [58], we use an additive transformation for the host image branch b_1^l, and employ an enhanced affine transformation for the hidden image branch b_2^l. Therefore, we adopt the bijection of the forward propagation, and Eq. 2 is reformulated as

$$\begin{aligned} b_1^{l+1} &= b_1^l + \phi(b_2^l), \\ b_2^{l+1} &= b_2^l \odot \exp(\rho(b_1^{l+1})) + \eta(b_1^{l+1}), \end{aligned} \tag{4}$$

where $\exp(\cdot)$ and $\rho(\cdot)$ are Exponential and arbitrary functions, respectively. $\odot$ is the Hadamard product. Thus, this is a variant of the augmented invertible block. Accordingly, our backward propagation operation is

$$\begin{aligned} b_2^l &= (b_2^{l+1} - \eta(b_1^{l+1})) \odot \exp(-\rho(b_1^{l+1})), \\ b_1^l &= b_1^{l+1} - \phi(b_2^l). \end{aligned} \tag{5}$$

The corresponding invertible blocks are shown in Fig. 2 (b). Note that $\exp(\cdot)$ of $\rho(\cdot)$ is omitted in the figure.

In our ISN, when generating the container image y_{co}, a constant matrix y_z is introduced (see the right of Fig. 2 (b)). When we attempt to hide an RGB image into another RGB image, there are 6 feature channels for the input and output of the invertible blocks, which means that the forward output also has 6 channels. However, we only need 3 feature channels to represent y_{co}. To keep the consistency of the channel number and feature information on both sides of the invertible network, we thus set the remaining 3 channels besides y_{co} as a constant matrix y_z.

It is noticeable that our ISN can be flexibly adapted to embed multiple x_{hi}. To achieve that, we directly concatenate such multiple x_{hi} in the channel dimension, and simultaneously increase the number of feature channels in the b_2 hidden branch, without changing the network architecture.

3.3. Loss Functions

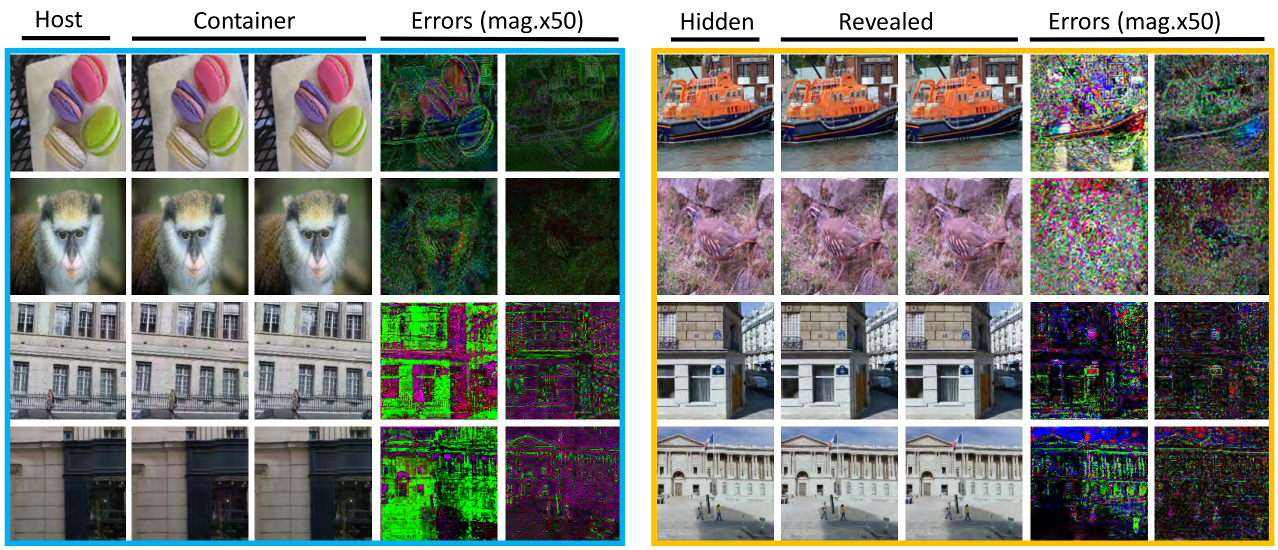
We aim to ensure that both the container image y_{co} and revealed images $\hat{x}_{hi}$ are as close as possible to the host x_{ho} and the hidden images x_{hi}, respectively. Therefore, we introduce the following two losses for y_{co} and x_{hi}:

$$\begin{aligned} \mathcal{L}_{co} &= \mathcal{F}(x_{ho}, y_{co}), \\ \mathcal{L}_{hi} &= \mathcal{F}(\hat{x}_{hi}, x_{hi}). \end{aligned} \tag{6}$$

Here $\mathcal{F}$ is the pixel-level distance function. Besides that, the following two losses are constructed respectively for the revealed host image x_{ho} and the constant matrix y_z,

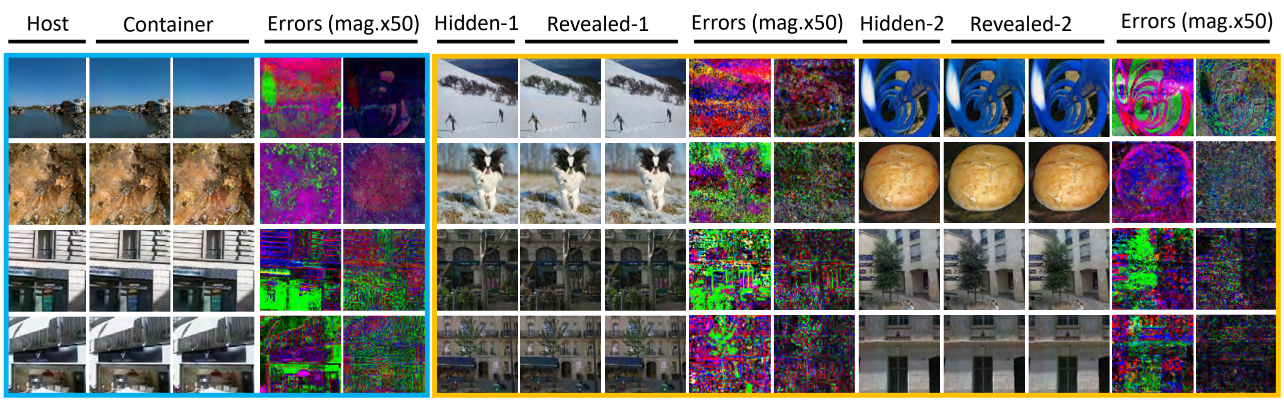
$$\begin{aligned} \mathcal{L}_{ho} &= \mathcal{F}(\hat{x}_{ho}, x_{ho}), \\ \mathcal{L}_z &= \mathcal{F}(\hat{y}_z, y_z). \end{aligned} \tag{7}$$

These two losses further constrain the system towards a unique solution for reconstructing the desired images. Fol-



(a) Original (b) [5] (c) Ours (d) [5] (e) Ours (f) Original (g) [5] (h) Ours (i) [5] (j) Ours

Figure 3. Visual comparisons for hiding and revealing an image.



(a) Original (b) [5] (c) Ours (d) [5] (e) Ours (f) Original (g) [5] (h) Ours (i) [5] (j) Ours (k) Original (l) [5] (m) Ours (n) [5] (o) Ours

Figure 4. Visual comparisons for hiding and revealing two images.

lowing [58], we use the l_2 and l_1 loss functions for the forward and backward propagation operations, respectively. In summary, our final loss function is

$$\mathcal{L} = \alpha_{co}\mathcal{L}_{co} + \alpha_z\mathcal{L}_z + \alpha_{ho}\mathcal{L}_{ho} + \alpha_{hi}\mathcal{L}_{hi}, \tag{8}$$

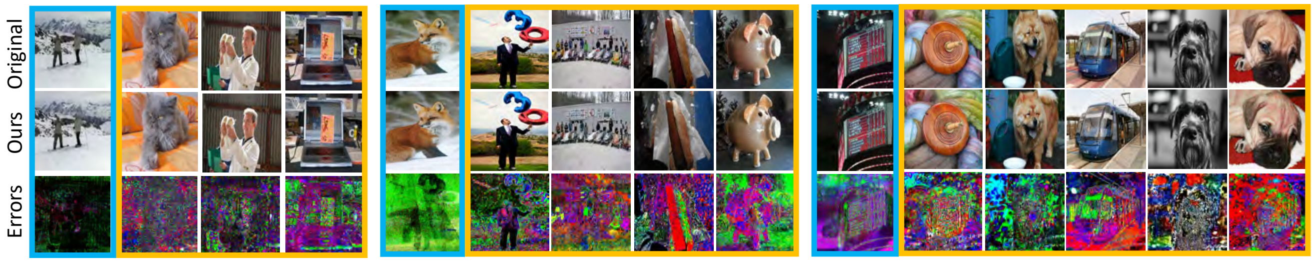
where $\alpha_{co}, \alpha_z, \alpha_{ho}, \alpha_{hi}$ are the weights of the corresponding losses presented above.

4. Experimental Results

4.1. Implementation Details

Our ISN is implemented with PyTorch, and an Nvidia Titan 2080Ti GPU is used for acceleration. We use the AdaMax optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 0.0002 and a mini-batch of size 2 to train our model. Our network contains several invertible blocks, each of them uses three 5-layer DenseNet blocks as the ϕ, ρ, and η sub-modules, respectively. The number of invertible blocks and the weights of our loss function are related to the steganography payload capacity, i.e. the number of hidden images (more details are in Sec. 4.4).

We train and test our network on the ImageNet [44] and Paris StreetView [16] datasets, which contain various natural and man-made scenarios. We randomly select 100,000 and 1,000 images from ImageNet as the training and testing sets, respectively. From the Paris StreetView dataset, we get 14,900 training images and 100 testing images. We randomly crop 144×144 patches for training, while flipping and rotation are also used for data augmentation. We alternate the forward and back propagation operations of the network during training. For each iteration, our network firstly performs forward calculation $F(x_{ho}, x_{hi})$ to obtain $(y_{co}, \hat{y}_z)$, secondly performs reverse calculation $F^{-1}(y_{co}, y_z)$, and then calculates the corresponding 4 losses and updates the parameters.



(a) 3 hidden images. (b) 4 hidden images. (c) 5 hidden images.

Figure 5. Results for hiding multiple images. Sub-figure (a), (b) and (c) respectively represent the results of hiding 3 $\sim$5 images, with a blue border on the host images and an orange border on the hidden images. In each sub-figure, the top row is the original images and the middle row is our generated results, while the third row is the $\times 50$ magnified errors between them.

Table 1. Objective comparison using PSNR/SSIM. -$_{h1}$ and -$_{h2}$ means to hide 1 and 2 images respectively. (c) means cross-domain testing, *i.e.* the model trained on another dataset is tested directly without fine-tuning.

method	ImageNet		Paris StreetView	
	Container	Revealed	Container	Revealed
Ours-$_{h1}$	**38.05/.954**	**35.38/.955**	**40.49**/.980	**43.33/.991**
Ours-$_{h1}$ (c)	36.48/.940	34.92/.950	39.28/.977	40.41/.985
[5]-$_{h1}$	36.02/.946	32.75/.933	36.80/**.986**	39.03/.984
[5]-$_{h1}$ (c)	30.12/.938	29.53/.897	38.29/.975	35.86/.971
Ours-$_{h2}$	**36.86/.945**	**32.21**/.920	**39.14**/.971	**39.05/.982**
Ours-$_{h2}$ (c)	35.57/.932	32.04/**.926**	38.69/.969	35.12/.962
[5]-$_{h2}$	30.18/.919	29.17/.898	37.14/**.978**	34.73/.964
[5]-$_{h2}$ (c)	29.85/.931	25.19/.833	35.20/.963	33.23/.955

An ISN for hiding an image takes approximately one day to train for 500,000 iterations. When performing inference, the entire process of hiding and revealing an image with 380$\times$380 resolution takes about 0.07 seconds. We also implement our model on MindSpore [1] and other platforms. More specifically, the inference speed of our model is increased by 12% on the Jittor deep learning framework [30].

4.2. Comparison

Here we conduct some comparison tests especially with the latest method proposed in [5]. Some other CNN-based methods like HiDDeN [62] and SteganoGAN [60] are not involved, because they still achieve traditional payload capacities (<4.5 bpp). We reimplemented the model in [5] using PyTorch, and trained it on both ImageNet and Paris StreetView datasets. The PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity) metrics are used to objectively evaluate the images. Note that the calculated values of the model trained by us are slightly lower than that reported in [5] (Tab. 1). This could be due to different testing data randomly chosen from the dataset. When hiding two images, we measure the reconstruction quality of their revealed results using their average PSNRs. The results in Tab. 1 indicate that our approach performs better in both hiding single image and multiple images. Interestingly, Tab. 1 also shows that when our model is specifically trained on Pairs StreetView with a small amount of data, the testing results obtained on ImageNet are still acceptable.

Visualization comparisons between our ISN and [5] are shown in Fig. 3. Due to the space limitation, here we only show two examples for each dataset (more examples are in the supplementary). To illustrate the difference between the original and generated images, we magnify the pixel-wise errors by 50 times. One can observe that both our generated container and revealed hidden images contain smaller errors than that of [5], which is consistent with the objective comparison. In general, these experiments show that our ISN obtains the optimal results both quantitatively and qualitatively, when hiding one or two images.

4.3. Hiding Multiple Images

Here we explore the steganography payload capacity of our ISN by embedding multiple images. Firstly, we embed two images into the host, and the visual comparison can be seen in Fig. 4. Furthermore, Fig. 5 visualizes the results with 3$\sim$5 hiding images, with a blue border around host images and an orange border around the hidden images. Clearly, at such a high steganography payload capacity, our ISN still obtains satisfactory container images, and moreover, it reveals all hidden images with high quality.

In Fig. 6, we further calculate the average PSNRs for the containers and revealed images when hiding different numbers of images. In each class of experiments, we randomly select 100 images for the test. Again, the PSNR corresponds to the average of all hidden images for every container image. As shown in Fig. 6, the average PSNR values of the revealed images decrease with the increasing number of the hidden images. It is easy to understand that it becomes more difficult to hide and reveal the information of more hidden images. Nevertheless, even for the extreme case of 5 hidden images, the PSNR of the revealed hidden images is still higher than 31 dBs, while the container images are with good visual imperceptibility ($\sim$36 dBs).

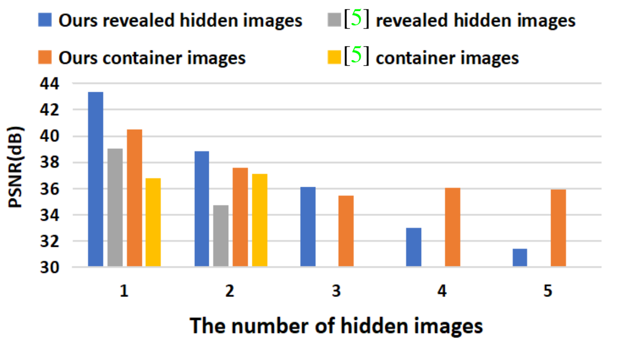


Figure 6. Average PSNRs of the revealed hidden images and the container images for embedding 1∼5 images.

Table 2. Ablation experiments for hiding 1∼ 2 images.

α_{co}	1 hidden image		2 hidden images	
	Container	Revealed	Container	Revealed
2	27.64/.908	41.38/**.994**	27.08/.856	38.04/.981
4	29.10/.935	42.26/.994	28.84/.894	38.15/**.986**
8	33.30/.961	41.16/.990	29.86/.922	37.48/.983
16	35.64/.974	41.99/.990	35.52/.932	**39.26**/.984
32	40.49/.980	**43.33**/.991	37.60/.958	38.87/.982
64	**42.40/.986**	40.73/.988	**39.14/.971**	39.05/.982

Table 3. Ablation experiments for hiding 4 images.

α_{co}	8 InvBlocks		16 InvBlocks	
	Container	Revealed	Container	Revealed
4	27.58/.779	**32.90/.945**	26.97/.787	**34.66/.960**
32	33.63/.928	31.61/.934	34.58/.923	33.22/.949
64	**36.53/.957**	31.12/.928	**36.03/.955**	33.02/.942

4.4. Ablation Experiments

The ablation experiments are performed on Paris StreetView. Here we mainly discuss the loss weight of the container image and the number of invertible blocks, which greatly impact the final results. For more detailed experiments on sub-modules selection and loss function adjustment, please see the supplementary.

As reported in Tab. 2, our ISN easily reveals high-quality images when embedding 1 or 2 images. By simply adjusting α_{co}, the weight of the container image in loss function Eq. (8), our network still gets desired container images. When hiding 2 images, without decreasing the quality of the revealed images (still higher than 38 dBs), changing α_{co} from 2 to 64 makes the container image gain +12.06 dBs and +0.115 for the PSNR and SSIM metrics, respectively.

Similarly, when hiding 4 images, increasing α_{co} can significantly improve the quality of the container image (see Tab. 3). However, it is difficult to do so for the revealed images. Still in this table, if we only use an ISN with 8 invertible blocks, the average PSNRs of the 4 revealed images are always less than 33 dBs. By increasing the number of invertible blocks from 8 to 16, the revealed images gain

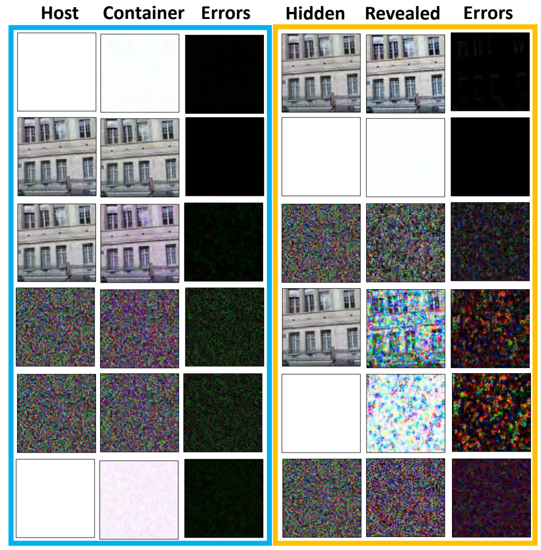


Figure 7. Visual results for some extreme cases, where the host or hidden images are monochrome, natural or random noise images.

+1.9 dBs, and the container image is higher than 36 dBs (the bottom row in Tab. 3). In other words, when dealing with more hidden images, the steganography and recovery capability of our method could be improved by appropriately increasing the number of blocks. According to Tab. 2 and Tab. 3, we set α_{co} to 32 for 1 hidden image and α_{co} to 64 for multiple hidden images, to ensure that the container image is sufficiently similar to the host image in most cases, and $\alpha_z, \alpha_{ho}, \alpha_{hi}$ are all set to 1.

5. Discussions

5.1. Extreme Cases

To explore the steganography capability of our proposed approach, we conduct experiments on some extreme images, including a natural image, a monochrome image and a random noise image. For every two images, we firstly select one of them to embed into the other. After that, we repeat the above experiment by switching these two images in our system. From the first two rows of Fig. 7, it could be observed that our method performs well when embedding a natural image into a monochrome image or vice versa. However, other results (the last four rows) show that if the noise image is used as a hidden or host image, it is difficult for our method to reveal the hidden image.

5.2. Passive Attack Analysis

Here we conduct passive attack analysis on container images generated by our method, and we employ two widely-used open source tools [7, 57] for this analysis. The first tool is ManTra-Net [57], which is designed to detect 385 image manipulation types. We compare the detection re-

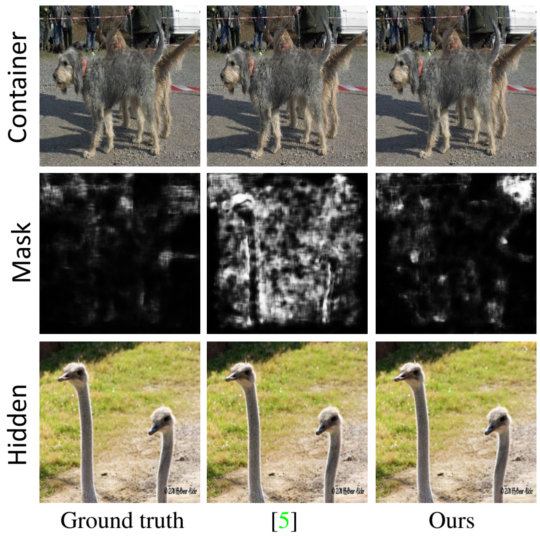


Figure 8. Forgery detection. The masks in the second row are the corresponding results of the container images in the first row, respectively, detected by ManTra-Net [57]. The third row shows the hidden images.

sults of our method against [5] using this image forgery tool. As shown in Fig. 8, the indicated abnormal information of our container image detected by [57] is close to that of the original host image. On the contrary, the result using [5] (the center of Fig. 8) shows more information of the hidden image. This proves the effectiveness of our steganography approach.

Another detection tool is named StegExpose [7], which is devised for LSB steganography detection, and includes four well-known steganalysis approaches. As shown in Fig. 9, the detection results with StegExpose on [5] and ours are shown in the form of receiver operating characteristic (ROC) curves. These two comparable curves indicate that StegExpose detection does not work well on both ours and the method in [5]. It is also interesting that the detection curve on [5] is slightly better than ours, while it is opposite for the PSNRs metrics reported before.

5.3. Encryption

As mentioned before, we force the output of the forward propagation in our hidden image branch to a constant matrix y_z during the training. For all our experiments in Sec. 4, all elements in y_z are set as 0.5, such that the consistency of the network structure is well preserved, and as expected the hidden image branch information is transferred to the host image branch.

Can y_z be further used as a key for hidden image extraction? We set y_z with other texture patterns during training, and try to reveal the hidden image under the assumption that y_z is unknown. Fig. 10 shows some revealed results of the hidden image when y_z is set with different textures. We can

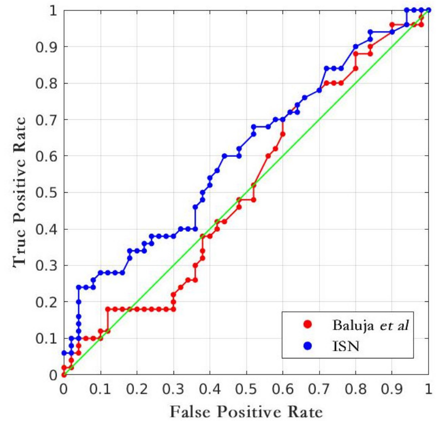


Figure 9. The ROC curve produced by setting different thresholds in StegExpose [7] when detecting the container images generated by [5] and our method.

Figure 10. Extracting the hidden image by setting y_z with different texture patterns. The top-left and bottom-left are respectively the container and hidden images. The remaining 5 images of the first row are texture patterns of y_z, and the bottom of them are the corresponding results revealed from the container images.

see that only with the correct y_z, the hidden image could be revealed with high quality. Note also that although the images revealed by incorrect y_z are distorted, the hidden contents are still partially recognizable.

6. Conclusion

In this paper, we have proposed an Invertible Steganography Network (ISN) for image steganography, where the forward and backward propagation operations of the same network are leveraged to embed and extract hidden images, respectively. Our method significantly improves the steganography payload capacity, and can be easily adapted to hide multiple images with high imperceptibility. Comprehensive experiments demonstrate that with significant improvement of the steganography payload capacity, our ISN method achieves state-of-the-art both visually and quantitatively.

Acknowledgements. We would like to thank reviewers and ACs for their valuable comments. This work is funded in part by NSFC (No. 61972216) and Tianjin NSF (No. 18JCYBJC41300 and No. 18ZXZNGX00110). The corresponding author of this paper is Shao-Ping Lu.

References

[1] MindSpore. https://www.mindspore.cn/, 2020. 6

[2] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *ICLR*, 2018. 3

[3] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019. 3

[4] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *NeurIPS*, pages 2069–2079, 2017. 2, 3

[5] Shumeet Baluja. Hiding images within images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 2, 3, 5, 6, 7, 8

[6] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *ICML*, pages 573–582, 2019. 3

[7] Benedikt Boehm. Stegexpose - a tool for detecting LSB steganography. *arXiv preprint arXiv:1410.6656*, 2014. 7, 8

[8] Chi-Kwong Chan and Lee-Ming Cheng. Hiding data in images by simple LSB substitution. *PR*, 37(3):469–474, 2004. 1, 2

[9] Yambem Jina Chanu, Kh Manglem Singh, and Themrichon Tuithung. Image steganography and steganalysis: A survey. *Int. J. Comput. Vision.*, 52(2), 2012. 2

[10] Marc Chaumont. Deep learning in steganography and steganalysis from 2015 to 2018. *arXiv preprint arXiv:1904.01444*, 2019. 2

[11] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3):727–752, 2010. 1

[12] Hung-Yu Chen, I-Sheng Fang, Chia-Ming Cheng, and Wei-Chen Chiu. Self-contained stylization via steganography for reverse and serial style transfer. In *IJCAI*, March 2020. 3

[13] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *NeurIPS*, pages 9916–9926, 2019. 3

[14] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 2, 3

[15] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016. 2, 3, 4

[16] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What Makes Paris Look like Paris? *ACM Trans. Graph.*, 31(4), 2012. 5

[17] Tomáš Filler, Jan Judas, and Jessica Fridrich. Minimizing embedding impact in steganography using trellis-coded quantization. In *Media forensics and security II*, volume 7541, page 754105, 2010. 2

[18] Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting LSB steganography in color, and gray-scale images. *IEEE Trans. Multimedia*, 8(4):22–28, 2001. 1, 2

[19] Jessica Fridrich, Tomáš Pevnỳ, and Jan Kodovskỳ. Statistically undetectable JPEG steganography: dead ends challenges, and opportunities. In *workshop on Multimedia & security*, pages 3–14, 2007. 2

[20] Anna C Gilbert, Yi Zhang, Kibok Lee, Yuting Zhang, and Honglak Lee. Towards understanding the invertibility of convolutional neural networks. In *IJCAI*, pages 1703–1710, 2017. 3

[21] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 3

[22] L. Guo, J. Ni, and Y. Q. Shi. An efficient JPEG steganographic scheme using uniform embedding. In *WIFS*, pages 169–174, 2012. 2

[23] L. Guo, J. Ni, and Y. Q. Shi. Uniform embedding for efficient JPEG steganography. *IEEE Trans. Inf. Forensics Secur.*, 9(5):814–825, 2014. 2

[24] Tariq Al Hawi, MA Qutayri, and Hassan Barada. Steganalysis attacks on stego-images using stego-signatures and statistical image properties. In *TENCON*, pages 104–107, 2004. 1, 2

[25] Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. In *NeurIPS*, pages 1954–1963, 2017. 2

[26] Stefan Hetzl and Petra Mutzel. A graph–theoretic approach to steganography. In *IFIP international conference on communications and multimedia security*, pages 119–128, 2005. 2

[27] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *WIFS*, pages 234–239, 2012. 2

[28] Vojtěch Holub and Jessica Fridrich. Digital image steganography using universal distortion. In *workshop on Information hiding and multimedia security*, pages 59–68, 2013. 2

[29] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1, 2014. 2

[30] Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Information Sciences*, 63(222103):1–222103, 2020. 6

[31] Shoko Imaizumi and Kei Ozawa. Multibit embedding algorithm for steganography of palette-based images. In *PSIVT*, pages 99–110, 2013. 2

[32] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *ICLR*, 2018. 3

[33] Inas Jawad Kadhim, Prashan Premaratne, Peter James Vial, and Brendan Halloran. Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research. *Neurocomputing*, 335:299–326, 2019. 1, 2

[34] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, pages 10215–10224, 2018. 3

[35] B. Li, M. Wang, J. Huang, and X. Li. A new cost function for spatial image steganography. In *ICIP*, pages 4206–4210, 2014. 2

[36] Bui Cong Nguyen, Sang Moon Yoon, and Heung-Kyu Lee. Multi bit plane image steganography. In *IWDW*, pages 61–70, 2006. 2

[37] Michiharu Niimi, Hideki Noda, Eiji Kawaguchi, and Richard O Eason. High capacity and secure digital steganography to palette-based images. In *ICIP*, volume 2, pages II–II, 2002. 2

[38] Feng Pan, Jun Li, and Xiaoyuan Yang. Image steganography method based on pvd and modulus function. In *ICECC*, pages 282–284, 2011. 2

[39] Tomáš Pevny, Patrick Bas, and Jessica Fridrich. Steganalysis by subtractive pixel adjacency matrix. *IEEE Trans. Inf. Forensics Secur.*, 5(2):215–224, 2010. 2

[40] Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *International Workshop on Information Hiding*, pages 161–177, 2010. 1, 2, 3

[41] N. Provos. Defending against statistical steganalysis. In *Usenix security symposium*, volume 10, pages 323–336, 2001. 2

[42] N. Provos and P. Honeyman. Hide and seek: an introduction to steganography. *IEEE Security Privacy*, 1(3):32–44, 2003. 1, 2

[43] Chuan Qin, Chin-Chen Chang, Ying-Hsuan Huang, and Li-Ting Liao. An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism. *IEEE Trans. Circuits Syst. Video Technol.*, 23(7):1109–1118, 2012. 2

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision.*, 115(3):211–252, 2015. 5

[45] Phil Sallee. Model-based steganography. In *IWDW*, pages 154–167, 2003. 2

[46] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. SSGAN: secure steganography based on generative adversarial networks. In *PCM*, pages 534–544, 2017. 2

[47] Yang Song, Chenlin Meng, and Stefano Ermon. Mintnet: Building invertible neural networks with masked convolutions. In *NeurIPS*, pages 11004–11014, 2019. 3

[48] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *CVPR*, June 2020. 3

[49] Weixuan Tang, Bin Li, Shunquan Tan, Mauro Barni, and Jiwu Huang. CNN-based adversarial embedding for image steganography. *IEEE Trans. Inf. Forensics Secur.*, 14(8):2074–2087, 2019. 2, 3

[50] W. Tang, S. Tan, B. Li, and J. Huang. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10):1547–1551, 2017. 2

[51] Qiang Tong, Song-Hai Zhang, Shi-Min Hu, and Ralph R Martin. Hidden images. In *NPAR*, pages 27–34, 2011. 3

[52] Piyu Tsai, Yu-Chen Hu, and Hsiu-Lien Yeh. Reversible image hiding scheme using predictive coding and histogram shifting. *Signal processing*, 89(6):1129–1143, 2009. 2

[53] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev. Steganographic generative adversarial networks. In *ICMV*, volume 11433, page 114333M, 2020. 2

[54] Yaolong Wang, Mingqing Xiao, Chang Liu, Shuxin Zheng, and Tie-Yan Liu. Modeling lost information in lossy image compression. *arXiv preprint arXiv:2006.11999*, 2020. 3

[55] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *CVPR*, June 2019. 3

[56] Da-Chun Wu and Wen-Hsiang Tsai. A steganographic method for images by pixel-value differencing. *Pattern recognition letters*, 24(9-10):1613–1626, 2003. 2

[57] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *CVPR*, pages 9543–9552, 2019. 7, 8

[58] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. *ECCV*, 2020. 2, 3, 4, 5

[59] J. Yang, D. Ruan, J. Huang, X. Kang, and Y. Shi. An embedding cost learning framework using GAN. *IEEE Trans. Inf. Forensics Secur.*, 15:839–851, 2020. 2

[60] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. SteganoGAN: High capacity image steganography with GANs. *arXiv preprint arXiv:1901.03892*, 2019. 2, 3, 6

[61] Li Zhi, Sui Ai Fen, and Yang Yi Xian. A LSB steganography detection algorithm. In *PIMRC*, volume 3, pages 2780–2783, 2003. 2

[62] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, pages 657–672, 2018. 1, 2, 3, 6

[63] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, Oct 2017. 3

[64] Xiaobin Zhu, Zhuangzi Li, Xiao-Yu Zhang, Changsheng Li, Yaqi Liu, and Ziyu Xue. Residual invertible spatio-temporal network for video super-resolution. In *AAAI*, volume 33, pages 5981–5988, 2019. 3