Hiding Images in Deep Probabilistic Models

Haoyu Chen
Department of Computer Science
City University of Hong Kong
haoychen3-c@my.cityu.edu.hk

Linqi Song
Department of Computer Science
City University of Hong Kong
linqi.song@cityu.edu.hk

Zhenxing Qian
School of Computer Science
Fudan University
zxqian@fudan.edu.cn

Xinpeng Zhang
School of Computer Science
Fudan University
zhangxinpeng@fudan.edu.cn

Kede Ma*
Department of Computer Science
City University of Hong Kong
kede.ma@cityu.edu.hk

Abstract

Data hiding with deep neural networks (DNNs) has experienced impressive successes in recent years. A prevailing scheme is to train an autoencoder, consisting of an *encoding* network to embed (or transform) secret messages in (or into) a carrier, and a *decoding* network to extract the hidden messages. This scheme may suffer from several limitations regarding practicability, security, and embedding capacity. In this work, we describe a different computational framework to hide images in deep probabilistic models. Specifically, we use a DNN to model the probability density of cover images, and hide a secret image in one particular location of the learned distribution. As an instantiation, we adopt a SinGAN, a pyramid of generative adversarial networks (GANs), to learn the patch distribution of one cover image. We hide the secret image by fitting a deterministic mapping from a fixed set of noise maps (generated by an embedding key) to the secret image during patch distribution learning. The stego SinGAN, behaving as the original SinGAN, is publicly communicated; only the receiver with the embedding key is able to extract the secret image. We demonstrate the feasibility of our SinGAN approach in terms of extraction accuracy and model security. Moreover, we show the flexibility of the proposed method in terms of hiding multiple images for different receivers and obfuscating the secret image.

1 Introduction

Data hiding generally refers to the process of hiding a form of secret message in another form of cover media, while minimizing the introduced distortions to the cover media [15, 52]. For human eavesdroppers, the measured distortion should be consistent with human judgments, penalizing errors that are most perceptually or cognitively noticeable [60, 5]; for machine eavesdroppers, the distortion should be "invisible" in a way that bypasses digital steganalysis tools such as StegExpose [10] and more recent deep learning-based ones [11]. Only the informed receiver typically with a shared embedding key (through a secure subliminal channel [55]) is able to extract the secret message. The

*Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

form of the secret message can be encrypted bit streams [16], texts [33], audio signals [41], images [7], and videos[57]. Similarly, the cover media can also be texts [9], audio signals [34], images [7], videos [57], neural networks [5, 61], and even human behaviours[72].

Like many problems in signal and image processing, data hiding has been revolutionized by the remarkable development of DNNs [7, 74, 31]. These methods typically follow an autoencoder approach with two key components: an *encoding* network and a *decoding* network. For *secret-in-image* hiding [7, 74], the encoding network takes the cover image and the secret message as inputs, and generates a stego image with the hidden message (see Fig. 1 (a)). For *secret-in-network* hiding [5, 61], the cover media becomes some pre-selected weight layers of the encoding network, and the secret message usually serves as a watermark (see Fig. 1 (b)). For *constructive* (or generative) image hiding [31], the encoding network directly maps the secret message to the stego image without reliance on any cover image (see Fig. 1 (c)). In all cases, the decoding network is responsible for extracting the secret message. Despite demonstrated success, the autoencoder scheme may suffer from three main drawbacks. First, the decoding network, whose size may be significantly larger than that of the secret message, must be sent to the receiver side via the subliminal channel[2], making the paradigm less practical. Second, it is not hard to re-train existing DNN-based steganalysis methods [11, 62] to identify stego images (or stego weight matrices), making the paradigm less secure. Third, it is difficult to hide multiple images for different receivers via the same encoding and decoding networks, making the paradigm less flexible.

Our Contributions. In this paper, we propose to hide images in deep probabilistic models, which is substantially different from the previous autoencoder scheme (see Fig. 1 (d)). The key idea is to use a DNN to model the high-dimensional probability density of training cover images, and hide the secret image in one particular location of the learned distribution. The stego DNN for density estimation is publicly communicated, from which we may draw samples that look like training cover images. Only guided sampling by the embedding key (shared between the sender and receiver) is able to reproduce the secret image.

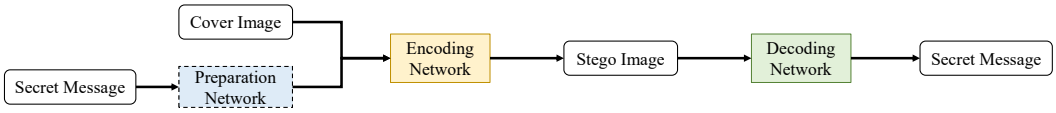
We construct a specific example under the proposed probabilistic image hiding framework. Specifically, we adopt a SinGAN [54], a pyramid of generative adversarial networks (GANs) [23], to implicitly learn the patch distribution of a single cover image. During distribution learning, we use the same SinGAN to fit a deterministic mapping from a fixed set of noise maps (generated by the shared embedding key) to the secret image, which completes the image hiding process. The stego SinGAN that behaves like the original one is publicly communicated. A single forward propagation suffices to extract the secret image by the receiver with the embedding key, and no decoding network is trained and transmitted. Experiments demonstrate that the proposed method is 1) feasible, extracting the secret image with improved accuracy (compared to the autoencoder-based methods), 2) secure, behaving normally as the original SinGAN in several aspects, and 3) flexible, hiding multiple images for different receivers and obfuscating the secret image with graceful performance degradation.

2 Related Work

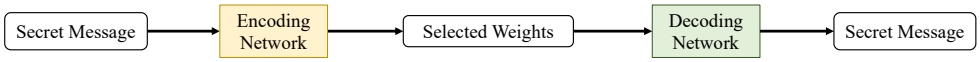
Depending on the applications, data hiding can be broadly divided into two subtopics: watermarking and steganography. Watermarking [26] aims to embed a digital watermark in a multimedia file for copyright protection and content management on social networks. Thus, watermarking techniques focus primarily on the robustness and perceptibility aspects of the embedded watermarks. Steganography [36, 52] aims to conceal a secret message within a cover media mainly for covert communication. Thus, steganography pays more attention to the trade-off among embedding capacity, extraction accuracy, and model security. Here we provide a concise review of image hiding techniques, and refer the interested readers to [52, 15, 69] for more comprehensive surveys of the field.

Secret-in-Image Hiding. The most common image steganography modifies the least significant bits (LSBs) of images, either uniformly [14, 47] or adaptively [51, 28], guided by the design of novel distortion functions. Other representative techniques include pixel value differencing [65], histogram shifting [50, 70], and recursive code construction [71]. Transform domain steganography [53, 30, 8]) have also been proposed with improved capacity and security. Often, these methods leave traces in the form of certain statistical irregularities, which can be easily revealed by simple

[2]The decoding network, if shared via a public channel, will be suspectable, as it is only trained to extract the secret message rather than performing typical machine learning tasks.

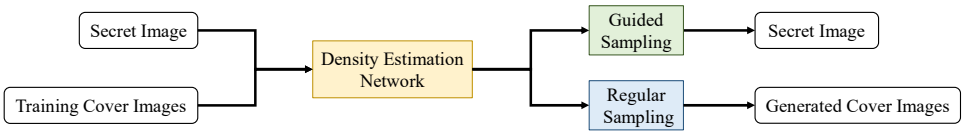


(a) Secret-in-image hiding. The preparation network is optional.



(b) Secret-in-network hiding. The secret message is usually in the form of a watermark.

(c) Constructive (or generative) image hiding.



(d) Proposed framework for hiding images in deep probabilistic models.

Figure 1: Paradigms for hiding data using DNNs.

steganalysis algorithms as countermeasures [21, 46, 29]. Zhu *et al.* [74] proposed one of the first DNN-based autoencoder for unified watermarking and steganography with an optional noise layer. Baluja [7] extended it to "image-in-image" steganography with a preparation network to preprocess the secret image. Weng *et al.* [64] further extended it to "video-in-video" steganography via temporal residual modeling. Normalizing flow-based invertible architectures [20] have also been investigated for multiple image hiding [45, 35, 24]. Here, we describe a different framework to accomplish a similar but more challenging goal - multiple image hiding for different users - with several other advantages.

Secret-in-Network Hiding. A prerequisite for secret-in-network hiding is that the hidden message should not affect the network performance on the given machine learning task. As a result, the message (mostly the watermark for intellectual property protection of the neural network) is commonly embedded during network training. Typical strategies for this purpose include parameter regularization [58], backdooring [5], output watermarking [66], and weight selection [61]. The proposed framework can be seen as a form of secret-in-network hiding, but with different goals (steganography instead of watermarking).

Constructive (Generative) Image Hiding. Traditional methods hide secret messages during the construction of some specific types of images, such as textures [67] and fingerprints [40]. Recent DNN-based constructive image hiding methods mainly aim to construct the mapping between secret messages and stego images of more unconstrained content types [43, 63]. The proposed framework can be seen as a form of constructive image hiding, where we hide a secret image during the "construction" of a probability density function, but with larger embedding capacity and improved model security.

3 Hiding Images in Deep Probabilistic Models

General Framework. Without loss of generality, we describe the general framework of hiding a single image in deep probabilistic models. The straightforward extension to multiple image hiding is described in Sec. 4.3. We assume a cover image dataset $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)}\}$, where

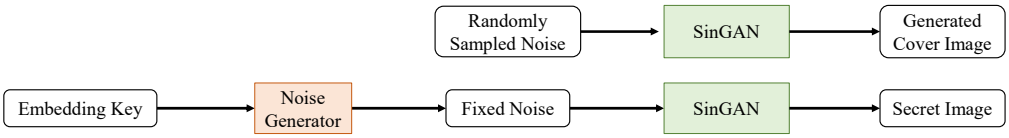


Figure 2: Hiding images in a SinGAN.

each image is drawn independently from an underlying image distribution $p(\boldsymbol{x})$. $\mathcal{D}$ may include a single image (where $\boldsymbol{x}^{(i)}$ becomes the i-th image patch), images of a specific class (*e.g.*, textures, fingerprints, and faces), and images on the natural image manifold. Also given is a secret image $\boldsymbol{x}^{(s)}$ of arbitrary content and an embedding key $\boldsymbol{k}$ shared by the sender and the receiver, which is a typical setting in the prisoner's problem formulated by Simmons [55].

The proposed probabilistic image hiding framework consists of two main steps. First, we learn a probability density function $p_s(\boldsymbol{x})$ over $\mathcal{D}_s = \mathcal{D} \bigcup \{\boldsymbol{x}^{(s)}\}$, *i.e.*, the combination of the cover image dataset and the secret image either explicitly via (approximate) maximum likelihood [39, 20, 56, 38] or implicitly via likelihood-free inference (*e.g.*, density estimation by comparison) [48]. The learned probability model $p_s(\boldsymbol{x})$ is publicly communicated in the proposed framework, which is usually in the form of a DNN (as the scoring function [22] or the sample generating function [48]). We may as well learn a *reference* probability density function $p_c(\boldsymbol{x})$ on the cover image dataset $\mathcal{D}$ solely. To guarantee the security of our framework, $p_s(\boldsymbol{x})$ should be as close to $p_c(\boldsymbol{x})$ as possible in some statistical distance (*i.e.*, distribution-preserving [13]). Provided that the adopted probability density estimator is robust (to the outlier image $\boldsymbol{x}^{(s)}$), which assigns an infinitesimal probability mass to $\boldsymbol{x}^{(s)}$, the proposed framework is perfectly secure in any statistical distance sense. Second, we design a guided sampling procedure (with the help of the embedding key $\boldsymbol{k}$) to draw a sample image $\hat{\boldsymbol{x}}^{(s)}$ from $p_s(\boldsymbol{x})$ that looks identical to the secret image $\boldsymbol{x}^{(s)}$. A third party without the embedding key is only able to generate samples that resemble cover images. It is noteworthy that the proposed framework does not require the learned $p_s(\boldsymbol{x})$ (or equivalently $p_c(\boldsymbol{x})$) to be sufficiently close to the underlying $p(\boldsymbol{x})$, a daunting (if not impossible) task to complete as digital images reside in a very high-dimensional space. In other words, it is perfectly fine that the samples drawn from $p_s(\boldsymbol{x})$ (without guidance) contain visually noticeable distortions as long as such distortions are shared by the samples drawn from $p_c(\boldsymbol{x})$.

Specific Example. Within the general framework of probabilistic image hiding, we provide a specific example that relies on GANs [23], a family of *implicit* probability models, which are represented by a stochastic procedure of data sampling. Although more natural to work with in our context compared to *prescribed* probabilistic models [18], unconditional GANs have the notorious reputation of being difficult to train, especially when modeling image sets with diverse content complexities and rich semantics. Thus, to make our work easily reproducible, we opt for a SinGAN [54] to learn the internal patch distribution of a single cover image $\boldsymbol{x}_0^{(c)}$ at various scales.

A SinGAN consists of a pyramid of generators $\{\boldsymbol{G}_0, \boldsymbol{G}_1 \ldots, \boldsymbol{G}_N\}$, where $\boldsymbol{G}_n$ takes the upsampled version of the generated image by $\boldsymbol{G}_{n+1}$ as well as an additive white Gaussian noise map $\boldsymbol{z}_n$ of the same size as inputs, and produces an image sample at the n-th scale:

$$\hat{\boldsymbol{x}}_n^{(c)} = \boldsymbol{G}_n\left(\boldsymbol{z}_n, \left(\hat{\boldsymbol{x}}_{n+1}^{(c)}\right)\uparrow^r\right), \quad n < N, \tag{1}$$

where $r > 1$ is the pre-defined upsampling ratio. The generation process starts at the coarsest scale (*i.e.*, the N-th scale), where the input is purely noise:

$$\hat{\boldsymbol{x}}_N^{(c)} = \boldsymbol{G}_N\left(\boldsymbol{z}_N\right), \tag{2}$$

and progressively makes use of all generators to produce the finest scale image $\hat{\boldsymbol{x}}_0^{(c)}$ with possibly different size and aspect ratio of $\boldsymbol{x}_0^{(c)}$ [54]. Coupled with the generators is a pyramid of discriminators $\{D_0, D_1, \ldots, D_N\}$, and each D_n is trained to discriminate between patches extracted from $\hat{\boldsymbol{x}}_n^{(c)}$ (in Eq. (1)) and $\boldsymbol{x}_n^{(c)}$, which is a downsampled version of $\boldsymbol{x}_0^{(c)}$ by a factor of r^n (*i.e.*, downsampled n times by a factor of r).

Table 1: Extraction accuracy comparison when hiding one image. "↑": larger is better, and vice versa.

Method	#params	PSNR↑	SSIM↑	DISTS↓
LSB	—	23.06	0.785	0.095
Baluja17	0.48M	25.91	0.874	0.102
HiDDeN	0.38M	27.99	0.897	0.096
Weng19	42.6M	35.64	0.942	0.055
HiNet	4.05M	35.59	0.952	0.047
Ours	0.67M	**36.84**	**0.958**	**0.038**

Table 2: Cover image quality, diversity, and weight distribution similarity between the original and stego SinGANs in terms of SIFID, DS, and KLD.

#images	SIFID↓	DS↑	KLD↓
0 (ref)	0.041	0.407	0
1	0.046	0.430	0.001
2	0.045	0.415	0.004
3	0.047	0.427	0.006
4	0.051	0.438	0.008

The training objective for each scale is a weighted combination of an adversarial term and a reconstruction term:

$$\min_{\boldsymbol{G}_n} \max_{D_n} \ell_{\text{adv}}\left(\boldsymbol{G}_n, D_n; \boldsymbol{x}^{(c)}\right) + \lambda \ell_{\text{rec}}\left(\boldsymbol{G}_n; \boldsymbol{x}^{(c)}\right), \tag{3}$$

where λ is the trade-off parameter. The adversarial loss ℓ_{adv} is for penalizing the statistical difference between the patch distribution of the generated $\hat{\boldsymbol{x}}_n^{(c)}$ and that of the (downsampled) cover image $\boldsymbol{x}_n^{(c)}$. The reconstruction loss ℓ_{rec} is for stabilizing the training process by ensuring that $\boldsymbol{x}_n^{(c)}$ can be reconstructed from a specific set of input noise maps. Following the original SinGAN paper [54], we use the WGAN-GP loss [25] and the mean squared error (MSE) to implement ℓ_{adv} and ℓ_{rec}, respectively.

After training, the SinGAN is capable of generating new image samples that preserve the patch distribution of $\boldsymbol{x}^{(c)}$, with novel and plausible scene configurations and structures. Once the learning procedure of the SinGAN is clear, hiding the secret image $\boldsymbol{x}^{(s)}$ during the patch distribution learning can be straightforwardly done by modifying the training objective from Eq. (3) to

$$\min_{\boldsymbol{G}_n} \max_{D_n} \ell_{\text{adv}}\left(\boldsymbol{G}_n, D_n; \boldsymbol{x}^{(c)}\right) + \lambda \ell_{\text{rec}}\left(\boldsymbol{G}_n; \boldsymbol{x}^{(s)}\right). \tag{4}$$

That is, a reconstruction loss is replaced to enforce that a specific set of input noise maps $\boldsymbol{z}^{(s)} = \{\boldsymbol{z}_0^{(s)}, \boldsymbol{z}_1^{(s)}, \ldots, \boldsymbol{z}_N^{(s)}\}$ is mapped to the secret image $\boldsymbol{x}^{(s)}$ instead of the cover image $\boldsymbol{x}^{(c)}$. $\boldsymbol{z}^{(s)}$ can be generated by a standard Gaussian pseudo-random number generator [12] using the embedding key $\boldsymbol{k}$ as the seed. We may as well put back the reconstruction term for $\boldsymbol{x}^{(c)}$, but we find this makes little difference during training and testing. We conjecture that the reconstruction loss in Eq. (4) not only enables hiding of the secret image, but also plays a similar role in improving the training stability and convergence. For ease of description, we refer to models optimized for Eq. (3) and Eq. (4) as the *original* and *stego* SinGANs, respectively.

To extract the secret image $\boldsymbol{x}^{(s)}$, the receiver uses the shared embedding key $\boldsymbol{k}$ to re-generate the noise maps $\boldsymbol{z}^{(s)}$, which is fed to the publicly transmitted stego SinGAN for secret image extraction via a single forward propagation (see Fig. 2).

4 Experiments

In this section, we perform a series of experiments to verify the promise of our SinGAN approach. First, we evaluate secret image extraction accuracy both quantitatively and qualitatively in comparison to image-in-image hiding methods based on autoencoders. Second, we probe the security of the stego SinGAN by comparing it to the original one in terms of 1) quality and diversity of generated cover images, 2) marginal distribution similarity of model parameters [61], and 3) possibility of secret image leakage. Third, we experiment with our method in two more challenging scenarios: 1) hiding multiple images within a SinGAN for different users and 2) hiding the content-obfuscated image. We implement the generators and discriminators of the SinGAN by medium-size DNNs, whose specifications and training details are given in the Appendix. Our quantitative experiments make use of 200 test image pairs, whose details are also given in the Appendix.

(a) Cover $\boldsymbol{x}^{(c)}$ (b) Secret $\boldsymbol{x}^{(s)}$ (c) Generated $\hat{\boldsymbol{x}}^{(c)}$ (d) Extracted $\hat{\boldsymbol{x}}^{(s)}$ (e) $5 \times |\boldsymbol{x}^{(s)} - \hat{\boldsymbol{x}}^{(s)}|$

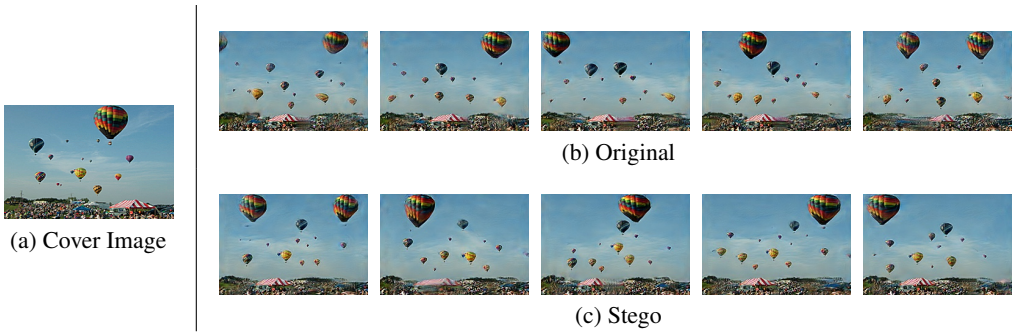
Figure 3: Visual results of our SinGAN approach. The second image of the last row is a pixel-shuffled version of the second image in the first row. Zoom in for improved visibility.

4.1 Extraction Accuracy

In image data hiding, extraction accuracy means how the secret image can be faithfully reproduced at the receiver side. Here, we apply three objective image quality measures to quantitatively evaluate the signal fidelity, perceptual fidelity, and perceptual quality of the extracted secret image: the peak signal-to-noise ratio (PSNR), the structural similarity (SSIM) index [60], and the deep image structure and texture similarity (DISTS) measure [19]. Although our SinGAN approach is the first of its kind in the proposed probabilistic image hiding framework, we compare it with one naïve LSB replacement method, and four image-in-image steganography methods - Baluja17 [7], HiDDeN [74], Weng19 [64] and HiNet[35]. The LSB replacement method simply replaces the four LSB planes of the cover image with the four most significant bit (MSB) planes of the secret image. As the robustness is not our focus, we re-train HiDDeN [74] for image hiding without the noise layer. Weng19 [64] is a DNN-based video steganography method, and only the image hiding branch is used for testing. HiNet [35] relies on a normalizing flow-based invertible DNN to implement the encoder, such that the decoder is simply its inverse. We use the publicly available implementations for Baluja17 [1], HiDDeN[2], Weng19[3] and HiNet[4] with default training and testing[3] settings, and implement the LSB replacement by ourselves.

The extraction accuracy results are shown in Table 1, where we find that the proposed SinGAN approach performs favorably against existing autoencoder-based image-in-image steganography methods. We consider the obtained improvements as significant because secret-in-network hiding is generally considered much more difficult than secret-in-image hiding, where the latter has significant spatial redundancy to be reduced. After all, the most recent secret-in-network hiding method [61] has a limited embedding capacity up to $6,000$ bits. We also show some representative visual results of our method in Fig. 3. It is clear that the stego SinGAN is able to capture the internal patch distribution of the cover image, generating image samples with different but reasonable structures and configurations

[3]It is important to note that the results of autoencoder-based methods in our paper (especially in terms of extraction accuracy in Table 1) may be noticeably different from some of the previous publications. This is because we choose to quantize the stego image from the single-precision floating-point format of 32×3 to 8×3 bits per pixel before transmitting it to the receiver side. If such quantization is not properly enforced, trivial hiding solutions may exist because there are just more space to accommodate the cover and secret images by a simple concatenation.



(a) Cover Image

(b) Original

(c) Stego

Figure 4: Visual comparison of the generated cover images "Fire balloons" by the original and stego SinGANs. In our case, the secret is the "Birds" image, as shown in the second row of Fig. 3.

of the same natural scene. Moreover, the extracted secret image by guided sampling is visually close to the original, which is adequate for use in the majority of real-world steganography applications.

4.2 Model Security

Steganographic analysis (*i.e.*, steganalysis) aims to identify whether there are hidden messages in suspected media, which constitutes an indispensable part of steganography evaluation. Traditional statistics-based and recent DNN-based steganalysis methods [10, 11] are mostly designed to discriminate between cover and stego images, thus not applicable when the cover media is a DNN. To the best of our knowledge, we are not aware of a steganalysis algorithm that accepts a full DNN (with millions of parameters) as input. Instead, we propose to probe the security of our SinGAN approach from the following three different aspects.

Quality and Diversity of Generated Cover Images. In the proposed framework, the stego SinGAN is publicly transmitted, and thus it must function as the original SinGAN [54]. As the primary goal of SinGANs is to model internal patch distributions, we examine and compare the quality and diversity of generated cover images by the original and stego SinGANs. To quantify quality, we adopt the single image Fréchet inception distance (SIFID) metric, as suggested in [54]. To quantify diversity, for each cover image, we compute the diversity score (DS) as the standard deviation (std) of each pixel values over 25 generated samples of the cover content, averaged over all pixels and normalized by the std of pixel intensities of the cover image [54]. The average quality and diversity results over 200 test image pairs are shown in Table 2, where we find that both SIFID and DS of the stego SinGAN are statistically indistinguishable from those of the original SinGAN based on a hypothesis testing using t-statistics [49]. Fig. 4 shows some randomly sampled examples from the original and stego SinGANs, which provides additional visual evidence that they learn very similar internal patch distributions of the cover image.

Marginal Distribution Similarity of Model Weights. As the proposed method essentially hides the secret image in the learned weights of the stego SinGAN, it is natural to ask whether its weight distributions significantly deviate from those of the original SinGAN. Following [61], we compute the Kullback–Leibler divergence (KLD) between the marginal distributions of model parameters of the original and stego SinGANs, as shown in the last column of Table 2. We find that the marginal distributions of the two sets of model parameters are almost identical, as evidenced by a KLD close to zero. Similar results can be obtained if the marginal distributions are compared in a per-stage fashion (see more results in the Appendix with visual comparison of histograms).

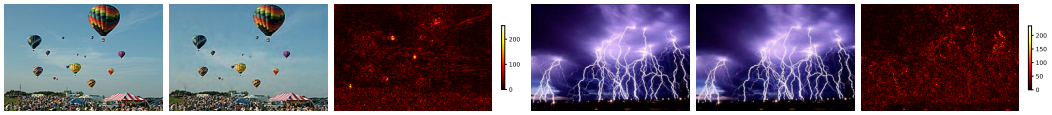
Possibility of Secret Image Leakage. One may also wonder the possibility of secret image leakage if an adversary constantly draws samples from the stego SinGAN. Such steganalysis arises naturally from the fact that there is no theoretical guarantee that the mapping between the secret noise (generated by the embedding key) and the secret image is bijective (*i.e.*, one-to-one). In other words, there might be some other sets of noise that are also mapped to the secret image, or at least some of its semantically meaningful content. Since the SinGAN is trained to be a black-box sampler, it is difficult to provide a theoretical analysis of the possibility of secret image leakage. Nevertheless, we conduct an empirical study, in which we randomly draw $100,000$ samples from each of the 200 trained stego SinGANs. Visual inspection of the thumbnails of generated samples indicates that no secret image

Table 3: Extraction accuracy of our method when hiding multiple images in one SinGAN.

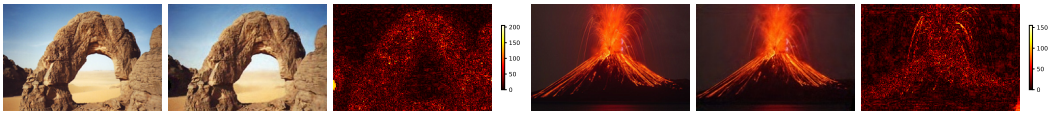
#images	PSNR↑	SSIM↑	DISTS↓
One	36.84	0.958	0.038
Two	35.91	0.946	0.043
Three	34.93	0.935	0.049
Four	34.03	0.923	0.055

Table 4: Extraction accuracy of our method with image obfuscation.

Obfuscation	PSNR↑	SSIM↑	DISTS↓
No	36.84	0.958	0.038
Yes	20.53	0.726	0.172



(a) 1st Original (b) Extracted (c) Error (5×) (d) 2nd Original (e) Extracted (f) Error (5×)



(g) 3rd Original (h) Extracted (i) Error (5×) (j) 4th Original (k) Extracted (l) Error (5×)

Figure 5: Visual comparison between the original and extracted secret images from a SinGAN with four hidden images.

(or images with similar semantics) is revealed. Therefore, it is safe to empirically conclude that the possibility of secret image leakage is less than 0.001%.

In summary, we have empirically proven that the proposed SinGAN approach is secure: hiding a full-size photographic image into a SinGAN does not compromise the quality and diversity of generated cover images, nor skew the weight distribution. It also survives constant sampling by an adversary. Such level of security verifies our claims that the secret image indeed occupies a tiny portion of probability mass of $p_s(\boldsymbol{x})$, and that $p_s(\boldsymbol{x})$ and $p_c(\boldsymbol{x})$ are statistically close.

4.3 Further Extensions

Hiding Multiple Images for Different Receivers. Hiding multiple images in a DNN is challenging; doing so for different receivers is even more challenging, and has not been accomplished before. Here the main difficulties lie not only in that more embedding capacity is required, but also in that each receiver must only extract her/his piece of message, and cannot extract (or even affirm the existence of) other messages [61]. The proposed probabilistic image hiding framework provides a straightforward and elegant extension to hide multiple images for different users, which follows a similar two-step approach. First, learn a probabilistic density function $p_s(\boldsymbol{x})$ over $\mathcal{D}_s = \mathcal{D} \bigcup \{\boldsymbol{x}^{(s_1)}, \ldots, \boldsymbol{x}^{(s_T)}\}$, where T is the number of secret images and is considerably smaller than M, the number of cover images. Second, design T guided sampling procedures using T different embedding keys $\mathcal{K} = \{\boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(T)}\}$, shared to T different receivers. The learning goal remains the same: $p_s(\boldsymbol{x})$ should be close in some statistical distance to the reference distribution $p_c(\boldsymbol{x})$. The described procedure is more easily understood using the SinGAN instantiation, where we just modify the objective function in Eq. (4) to

$$\min_{\boldsymbol{G}_n} \max_{D_n} \ell_{\text{adv}}\left(\boldsymbol{G}_n, D_n; \boldsymbol{x}^{(c)}\right) + \lambda \frac{1}{T} \sum_{t=1}^{T} \ell_{\text{rec}}\left(\boldsymbol{G}_n; \boldsymbol{x}^{(s_t)}\right). \tag{5}$$

After training, the t-th receiver is able to re-generate the t-th specific set of noise maps $\boldsymbol{z}^{(s_t)}$ using the shared embedding key $\boldsymbol{k}^{(t)}$ for the t-th secret image extraction. S/he is, by design, ignorant of the presence (or absence) of other secret images. Even if the receiver is informed in some way that the current SinGAN contains multiple secret images, without extra embedding keys, s/he cannot extract images that are not intended to share with her/him.

We train SinGANs to hide up to four secret images, *i.e.*, $T \in \{2, 3, 4\}$. For each value of T, we train 200 SinGANs to hide different combinations of cover and secret images. The quality and diversity of the cover image, and the weight distribution similarity between the original and stego SinGANs are shown in Table 2, where we find that hiding multiple images does not seem to compromise the security of the proposed method. The extraction accuracy results and visual examples are shown in Table 3 and Fig. 5, respectively. With the number of increasing secret images, the extraction accuracy in terms of the three objective metrics degrades gracefully. The visual appearances of the extracted secret images are similar to those of the original, showing great promise of our method in hiding multiple images for different receivers.

Obfuscating the Secret Image. Inspired by [7], we consider obfuscating the secret image by shuffling its pixels (*i.e.*, image scrambling [59]), as a way of strengthening security. In this case, The shuffling key, together with the embedding key, is shared to the receiver. Table 4 shows the extraction accuracy results, where we see that pixel shuffling significantly increases the difficulty of image hiding. Nevertheless, from the last row of Fig. 3, we observe that the main content is clearly visible, despite somewhat noisy appearance.

5 Conclusion and Discussion

We have described a new computational framework for hiding images in deep probabilistic models, which is in stark contrast to previous steganography schemes. We provided an instantiation, where we used the SinGAN to build the internal patch distribution of the cover image, and hid the secret image during patch distribution learning. We conducted a series of experiments to demonstrate the feasibility of the proposed SinGAN approach in terms of extraction accuracy and model security. Moreover, our method is readily extended to hide multiple images for different receivers, a challenging task that has not been accomplished before. In addition, it works nicely with pixel shuffling, which adds additional security.

The current work opens the door to a new class of image hiding methods, with many interesting problems to be explored. First, the extraction accuracy of our SinGAN approach still has quite some room for improvement. For applications that require precise recovery of the secret image, it is worth exploring more efficient network structures, optimized for perceptual losses that exploit the physiological properties of the human visual system. Second, our method naïvely bypasses the steganalysis tools specifically designed for secret-in-image steganography. Currently, we have designed three different tests to probe the model security. As SinGANs [54] can be applied in a much wider range of image manipulation tasks such as super-resolution and paint-to-image translation, the stego SinGAN should be tested in those applications as well for model security. More importantly, we expect future effort to be dedicated to building steganalysis methods that accept a full DNN as input and assess whether it contains secret messages. Third, so far, we have just given the theoretical intuition of the probabilistic image hiding framework, supplied with empirical evidence. It is of mathematical interest to rigorously measure the statistical distance between $p_c(\boldsymbol{x})$ and $p_s(\boldsymbol{x})$, in an attempt to answer important questions like 1) where the secret image is hidden in the network (or equivalently the learned distribution) and 2) what the maximum number of secret images is allowed for a given distance constraint (*e.g.*, $\mathrm{KLD}(p_c(\boldsymbol{x}) \parallel p_s(\boldsymbol{x})) \leq \epsilon$). Fourth, many other generative modeling methods [56] are worthy of deep investigation within the proposed framework to circumvent some of the limitations of the SinGAN approach. For example, we may model the (Stein) score function [44, 32] of $p_s(\boldsymbol{x})$ (*i.e.*, the gradient of $\log p_s(\boldsymbol{x})$), and develop guided Langevin-type sampling (by the embedding key) to extract the secret image.

Acknowledgments and Disclosure of Funding

The authors would like to thank Song Yang for inspiring discussion. This work was supported in part by the Hong Kong RGC ECS Grants 21213821 (to KDM) and 21212419 (to LQS), the National Natural Science Foundation of China under Grants 62071407, 61936214, U20B2051 and U1936214, the InnoHK initiative, the Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, and the Tencent AI Lab Rhino-Bird Gift Fund.

References

[1] Deep steganography, 2017. [Source code]. Available: https://github.com/alexandremuzio/deep-steg.

[2] HiDDeN: Hiding data with deep networks, 2018. [Source code]. Available: https://github.com/jirenz/hidden.

[3] Pytorch-Deep-Steganography, 2019. [Source code]. Available: https://github.com/muziyongshixin/pytorch-deep-steganography.

[4] HiNet: Deep image hiding by invertible network, 2021. [Source code]. Available: https://github.com/tomtomtommi/hiNet.

[5] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*, pages 1615–1631, 2018.

[6] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[7] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *Neural Information Processing Systems*, 2017.

[8] Mauro Barni, Franco Bartolini, and Alessandro Piva. Improved wavelet-based watermarking through pixel-wise masking. *IEEE Transactions on Image Processing*, 10(5):783–791, 2001.

[9] Krista Bennett. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text. *CERIAS Tech Report*, 2004.

[10] Benedikt Boehm. StegExpose-a tool for detecting LSB steganography. *arXiv preprint arXiv:1410.6656*, 2014.

[11] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193, 2018.

[12] Richard P Brent. Algorithm 488: A Gaussian pseudo-random number generator. *Communications of the ACM*, 17(12):704–706, 1974.

[13] Christian Cachin. An information-theoretic model for steganography. In *International Workshop on Information Hiding*, pages 306–318, 1998.

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

[15] 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.

[16] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2007.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[18] Peter J Diggle and Richard J Gratton. Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):193–212, 1984.

[19] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2567–2581, 2022.

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

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

[22] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Neural Information Processing Systems*, pages 2672–2680, 2014.

[24] Zhenyu Guan, Junpeng Jing, Xin Deng, Mai Xu, Lai Jiang, Zhou Zhang, and Yipeng Li. DeepMIH: Deep invertible network for multiple image hiding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[25] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. *Neural Information Processing Systems*, 30:5769–5779, 2017.

[26] Frank Hartung and Martin Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, 1999.

[27] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021.

[28] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *IEEE International Workshop on Information Forensics and Security*, pages 234–239, 2012.

[29] Vojtěch Holub and Jessica Fridrich. Random projections of residuals for digital image steganalysis. *IEEE Transactions on Information Forensics and Security*, 8(12):1996–2006, 2013.

[30] Chiou-Ting Hsu and Ja-Ling Wu. Hidden digital watermarks in images. *IEEE Transactions on Image Processing*, 8(1):58–68, 1999.

[31] Donghui Hu, Liang Wang, Wenjie Jiang, Shuli Zheng, and Bin Li. A novel image steganography method via deep convolutional generative adversarial networks. *IEEE Access*, 6:38303–38314, 2018.

[32] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[33] Firas A Jassim. A novel steganography algorithm for hiding text in image using five modulus method. *arXiv preprint arXiv:1307.0642*, 2013.

[34] P Jayaram, H R Ranganatha, and H S Anupama. Information hiding using audio steganography–a survey. *The International Journal of Multimedia & Its Applications*, 3:86–96, 2011.

[35] Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. HiNet: Deep image hiding by invertible network. In *IEEE/CVF International Conference on Computer Vision*, pages 4733–4742, 2021.

[36] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

[39] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[40] Sheng Li and Xinpeng Zhang. Toward construction-based data hiding: From secrets to fingerprint images. *IEEE Transactions on Image Processing*, 28(3):1482–1497, 2019.

[41] Xin Li and Hong H Yu. Transparent and robust audio data hiding in cepstrum domain. In *IEEE International Conference on Multimedia and Expo*, pages 397–400, 2000.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[43] Ming-ming Liu, Min-qing Zhang, Jia Liu, Ying-nan Zhang, and Yan Ke. Coverless information hiding based on generative adversarial networks. *arXiv preprint arXiv:1712.06951*, 2017.

[44] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pages 276–284, 2016.

[45] Shao-Ping Lu, Rong Wang, Tao Zhong, and Paul L Rosin. Large-capacity image steganography based on invertible neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10816–10825, 2021.

[46] Siwei Lyu and Hany Farid. Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1(1):111–119, 2006.

[47] Jarno Mielikainen. LSB matching revisited. *IEEE Signal Processing Letters*, 13(5):285–287, 2006.

[48] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[49] Douglas C Montgomery and George C Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2010.

[50] Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):354–362, 2006.

[51] 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.

[52] Niels Provos and Peter Honeyman. Hide and seek: An introduction to steganography. *IEEE Security & Privacy*, 1(3):32–44, 2003.

[53] Joseph Ó Ruanaidh, William J Dowling, and Francis M Boland. Phase watermarking of digital images. In *IEEE International Conference on Image Processing*, pages 239–242, 1996.

[54] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019.

[55] Gustavus J Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67, 1984.

[56] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[57] Mitchell D Swanson, Bin Zhu, and Ahmed H Tewfik. Data hiding for video-in-video. In *IEEE International Conference on Image Processing*, pages 676–679, 1997.

[58] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ACM on International Conference on Multimedia Retrieval*, pages 269–277, 2017.

[59] Dimitri Van De Ville, Wilfried Philips, Rik Van de Walle, and Ignace Lemahieu. Image scrambling without bandwidth expansion. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):892–897, 2004.

[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[61] Zichi Wang, Guorui Feng, Hanzhou Wu, and Xinpeng Zhang. Data hiding in neural networks for multiple receivers. *IEEE Computational Intelligence Magazine*, 16(4):70–84, 2021.

[62] Zihan Wang, Neng Gao, Xin Wang, Ji Xiang, Daren Zha, and Linghui Li. HidingGAN: High capacity information hiding with generative adversarial network. *Computer Graphics Forum*, 38:393–401, 2019.

[63] Ping Wei, Ge Luo, Qi Song, Xinpeng Zhang, Zhenxing Qian, and Sheng Li. Generative steganographic flow. *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2022.

[64] Xinyu Weng, Yongzhi Li, Lu Chi, and Yadong Mu. High-capacity convolutional video steganography with temporal residual modeling. In *International Conference on Multimedia Retrieval*, pages 87–95, 2019.

[65] Da-Chun Wu and Wen-Hsiang Tsai. A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters*, 24:1613–1626, 2003.

[66] Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2591–2601, 2020.

[67] Kuo-Chen Wu and Chung-Ming Wang. Steganography using reversible texture synthesis. *IEEE Transactions on Image Processing*, 24(1):130–139, 2015.

[68] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[69] Chaoning Zhang, Chenguo Lin, Philipp Benz, Kejiang Chen, Weiming Zhang, and In So Kweon. A brief survey on deep learning based data hiding, steganography and watermarking. *arXiv preprint arXiv:2103.01607*, 2021.

[70] Weiming Zhang, Xiaocheng Hu, Xiaolong Li, and Nenghai Yu. Recursive histogram modification: Establishing equivalency between reversible data hiding and lossless data compression. *IEEE Transactions on Image Processing*, 22(7):2775–2785, 2013.

[71] Weiming Zhang, Xiaocheng Hu, Xiaolong Li, and Nenghai Yu. Optimal transition probability of reversible data hiding for general distortion metrics and its applications. *IEEE Transactions on Image Processing*, 24(1):294–304, 2014.

[72] Xinpeng Zhang. Behavior steganography in social network. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 21–23. 2017.

[73] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018.

[74] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding data with deep networks. In *European Conference on Computer Vision*, pages 657–672, 2018.

Checklist

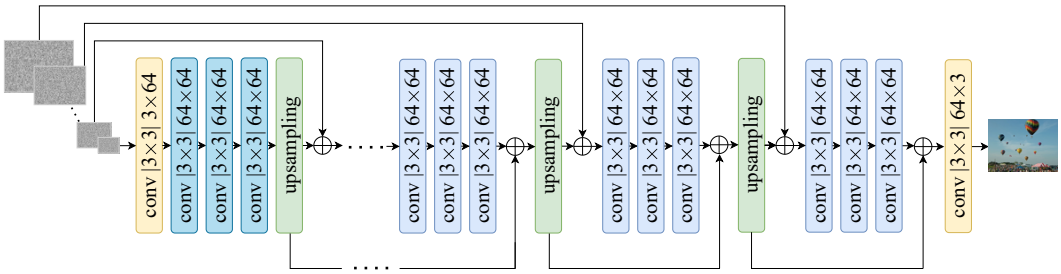
1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 4.2, we describe the possibility of secret image leakage.
   (c) Did you discuss any potential negative societal impacts of your work? [No] This is a new method for data hiding. The societal impacts of the proposed method are the same as those of the existing data hiding methods in general.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Because it would be too computational expensive.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] In the appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Specifications and Training Details

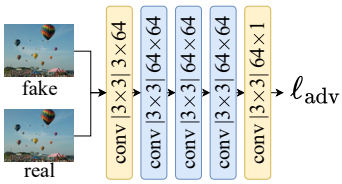
A.1 Model Architecture

The architecture of the SinGAN used in our paper follows that in [27]. As shown in Fig. 6, the generator at the n-th scale consists of a front-end convolution, $N - n + 1$ convolution blocks each with three convolution layers. Each convolution block, except for the last one, is followed by an upsampling layer with two residual connections, one for residual learning and one for incorporating a noise map from a higher scale (with a smaller spatial size). The last convolution block is responsible for the n-th scale image reconstruction with one residual connection. One back-end convolution with $\tanh()$ nonlinear activation is used to produce a scaled version of the RGB image as output. All convolution layers (except for the back-end one) have 64 filters with a filter size of 3×3, followed by batch normalization and leaky ReLU activation with the negative slope of 0.05. The weights of the front-end convolution, and the first $N - n$ convolution blocks, and the back-end convolution are inherited from those of the trained generator at the $n + 1$-th scale.

All generators share one discriminator, which is composed of five convolution layers all with a filter size 3×3. The first four convolution layers have 64 filters followed by leaky ReLU activation with the negative slope of 0.05, while the last convolution layer has a single convolution filter.



(a) n-th scale generator.



(b) Discriminator (shared).

Figure 6: Architecture of the SinGAN used in our paper.

A.2 Optimization

We adopt the improved techniques for training SinGANs as recommended in [27]. The λ in Eq. (4) is set to 10. The trade-off parameter in WGAN-GP [25] is set to 0.1 for gradient penalty. For the generator at the n-th scale, the newest three convolution blocks along with the front-end and back-end convolutions are jointly trained (or fine-tuned), while holding the older convolution blocks fixed (if any). This training strategy seems effective in preventing mode collapse. Adam[37] is adopted as the stochastic optimizer with an initial learning rate of 0.0005 and a decay factor of 0.1 after finishing 80% of iterations, and we set the maximum number of training iterations to $2,000$. The training time is approximately 20 minutes for one image pair (each with size $244 \times 164 \times 3$) on an NVIDIA GeForce RTX3080 GPU.

B Test Image Pairs for Quantitative Experiments

The 200 test image pairs used in our quantitative experiments are randomly drawn from five popular datasets (COCO[42], DIV2K[6], LSUN bedroom[68], ImageNet[17], Places[73]). Specifically, we

sample 80 images from each dataset to obtain a total of 400 images, which are randomly partitioned into 200 cover images and 200 secret images, as shown in Figs. 7 and 8 respectively, with co-located images forming one pair.

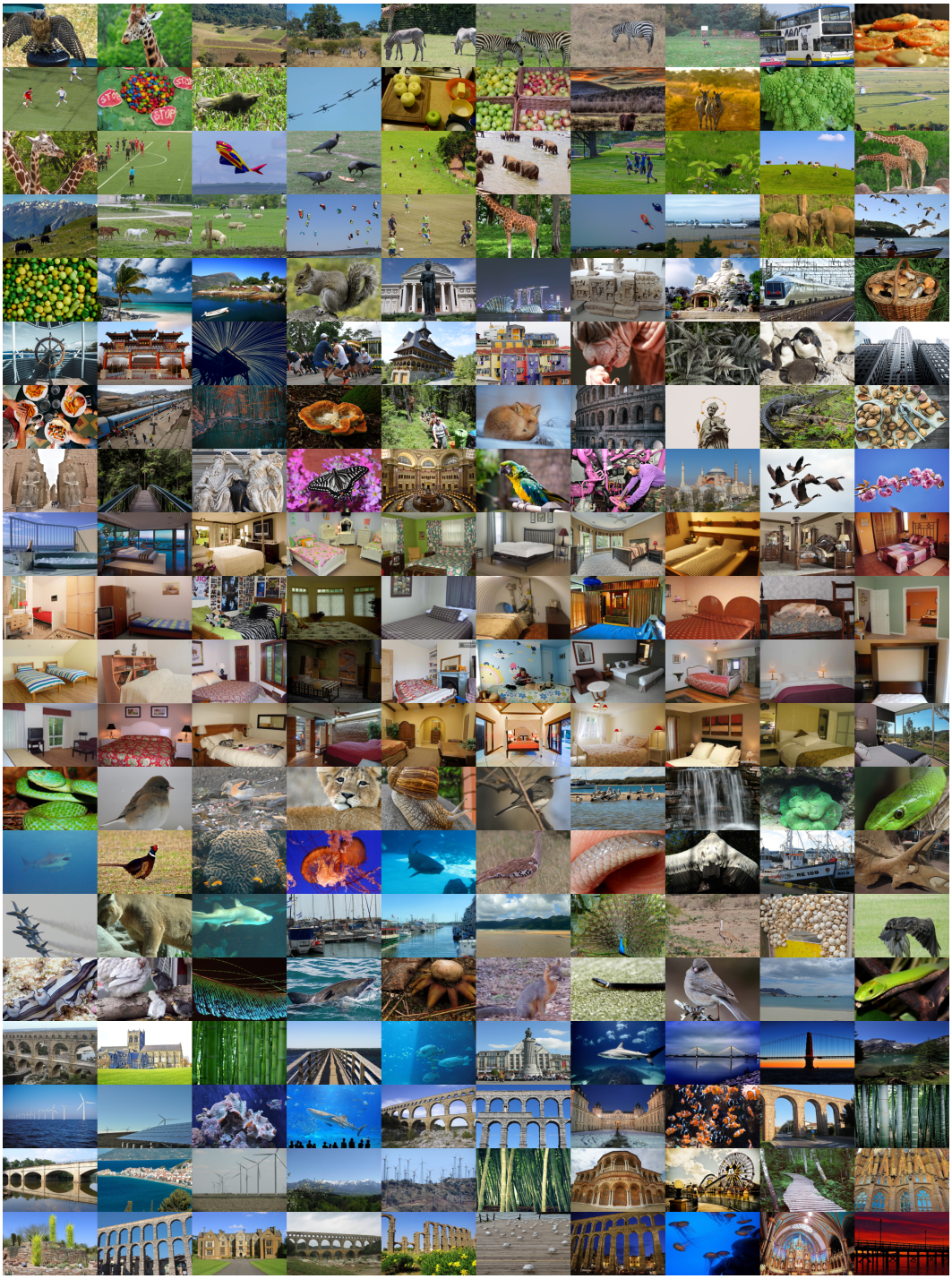


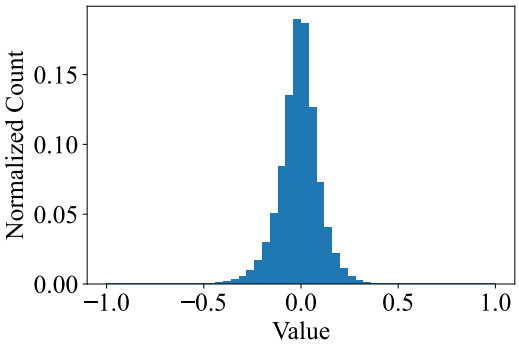
Figure 7: 200 cover images used for quantitative comparison.

Figure 8: 200 secret images used for quantitative comparison. The two co-located cover and secret images form one pair.

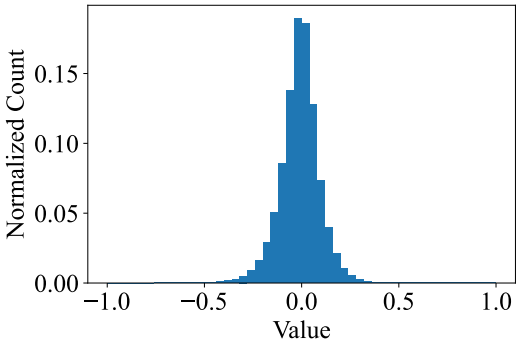
C Histograms of Weight Distribution

C.1 Total Weight Distribution

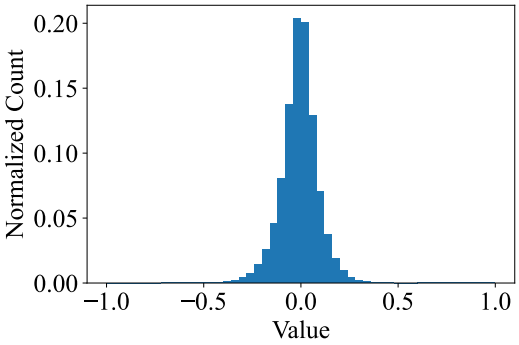
Fig 9 shows the histograms of all weights from 200 original and stego SinGAN generators. Careful visual inspection shows that the empirical weight distributions of the original and stego SinGAN generators are identical even when we hide up to four images.



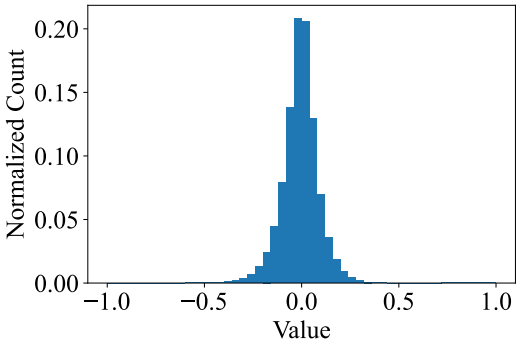
(a) No secret image (original).



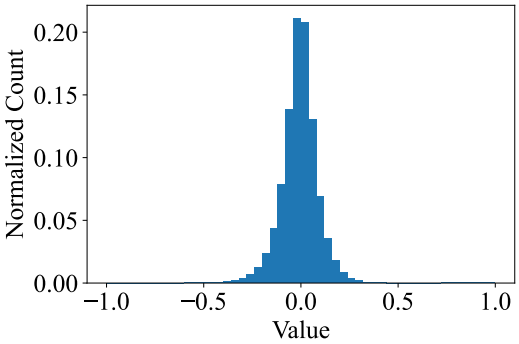
(b) One secret image.



(c) Two secret images.



(d) Three secret images.



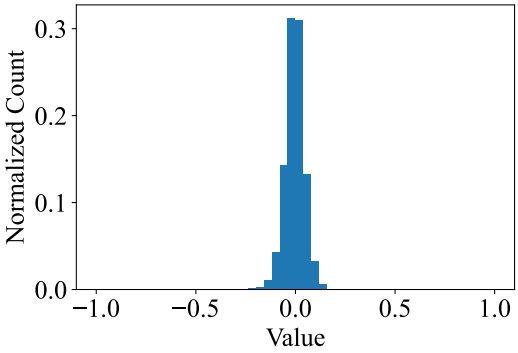
(e) Four secret images.

Figure 9: Visual comparison of histograms of the total weights.

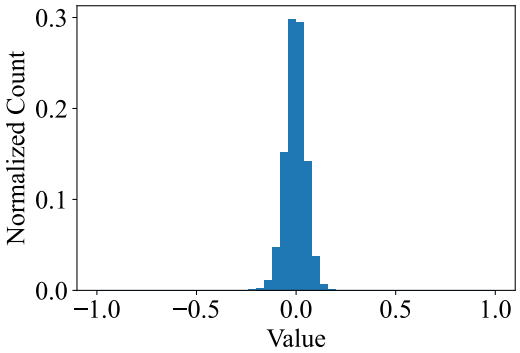
C.2 Per-Stage Weight Distribution

In addition to total weight distribution, the comparison of per-stage weight distribution is also provided. Here, one "stage" refers to a convolution block of the SinGAN generator, consisting of three convolution layers, as shown in Fig 6. Per-stage comparison gives us a finer view of the weight

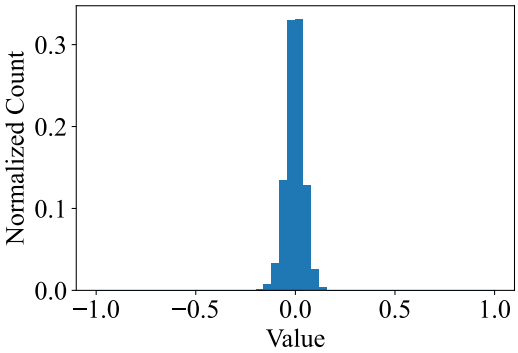
distributions of the original and stego SinGAN generators. As our final generator has six stages (*i.e.*, six convolution blocks), we show six sets of per-stage weight distribution comparison results in Figs. 10 to 15. Here, the calculation of normalized count is with respect to one stage of weights across 200 SinGANs. Again, no visually noticeable differences can be observed.



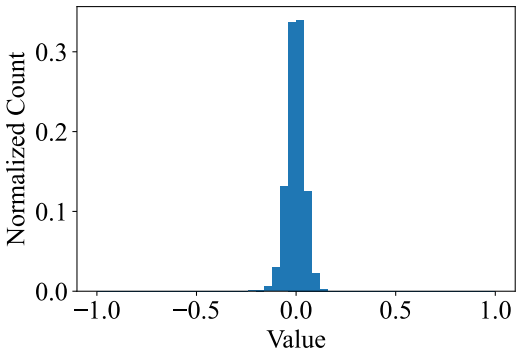
(a) No secret image (original).



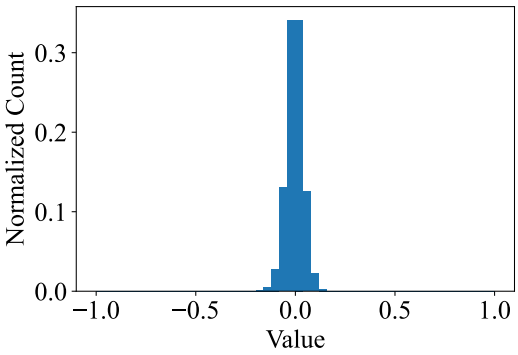
(b) One secret image.



(c) Two secret images.

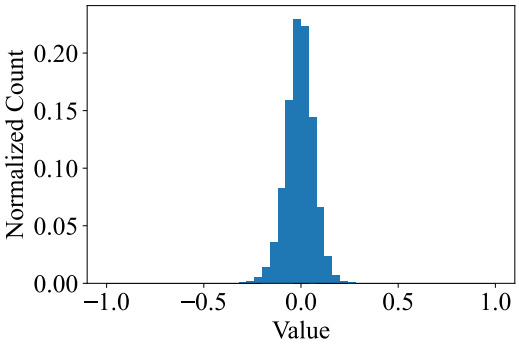


(d) Three secret images.

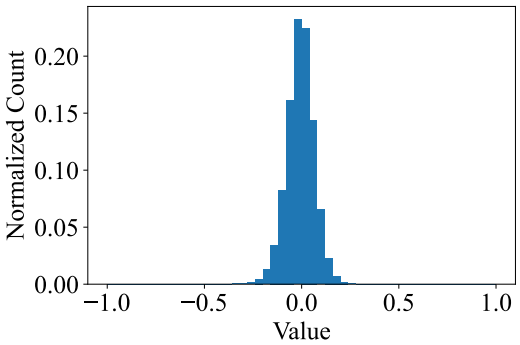


(e) Four secret images.

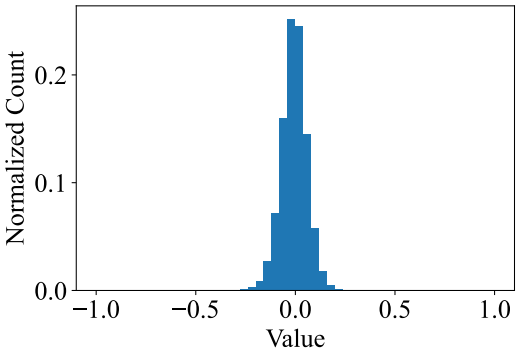
Figure 10: Visual comparison of histograms of the first-stage weights.



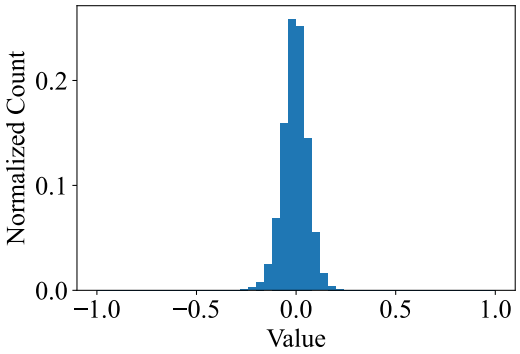
(a) No secret image (original).



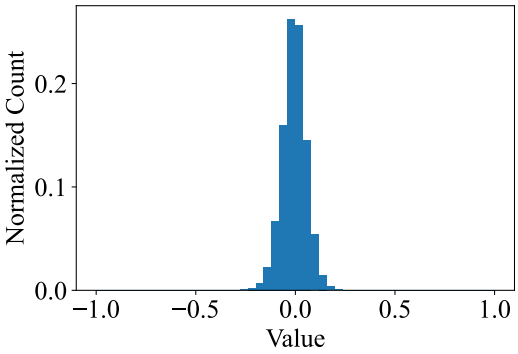
(b) One secret image.



(c) Two secret images.

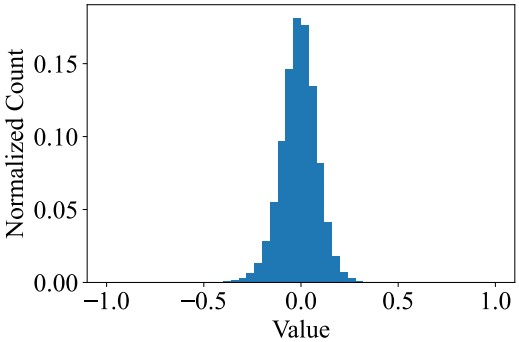


(d) Three secret images.

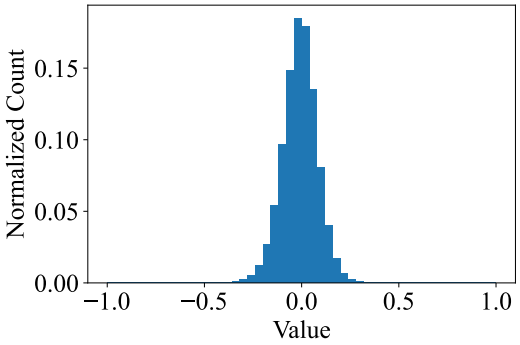


(e) Four secret images.

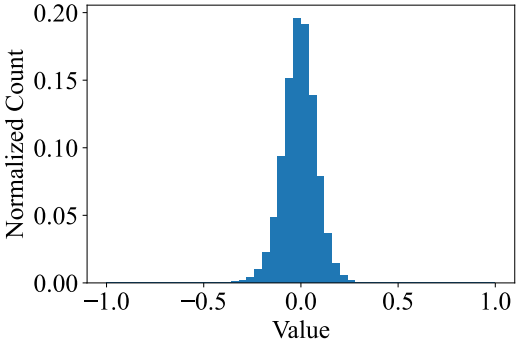
Figure 11: Visual comparison of histograms of the second-stage weights.



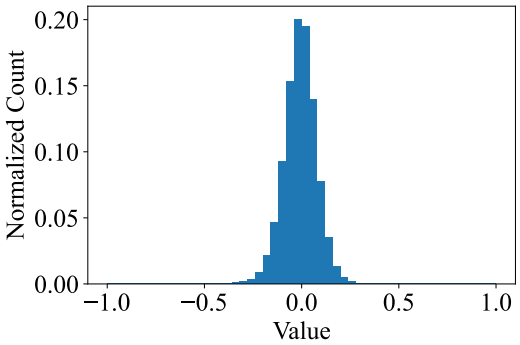
(a) No secret image (original).



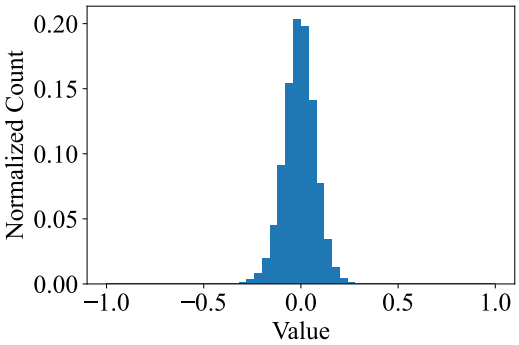
(b) One secret image.



(c) Two secret images.

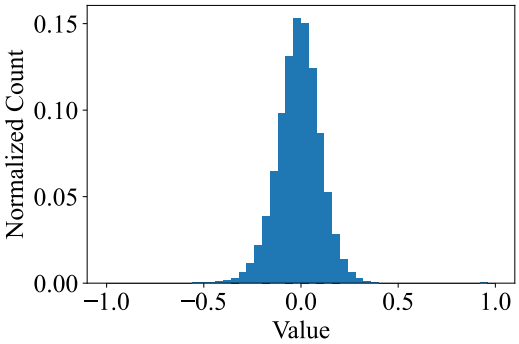


(d) Three secret images.

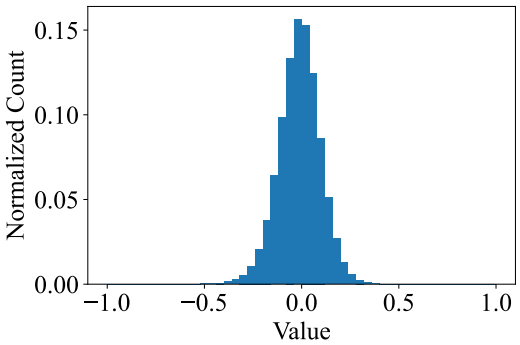


(e) Four secret images.

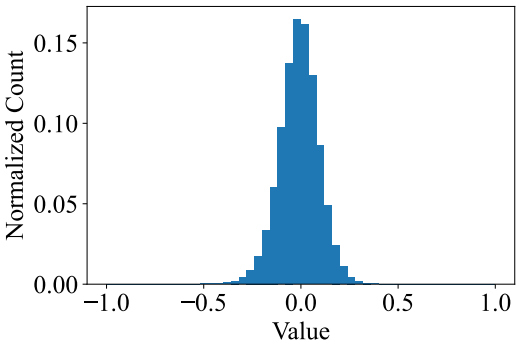
Figure 12: Visual comparison of histograms of the third-stage weights.



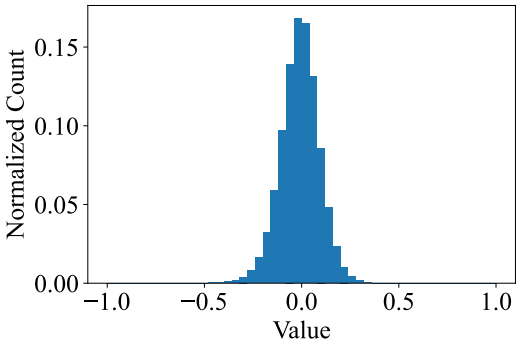
(a) No secret image (original).



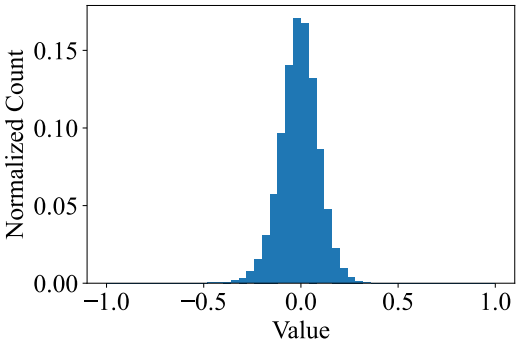
(b) One secret image.



(c) Two secret images.

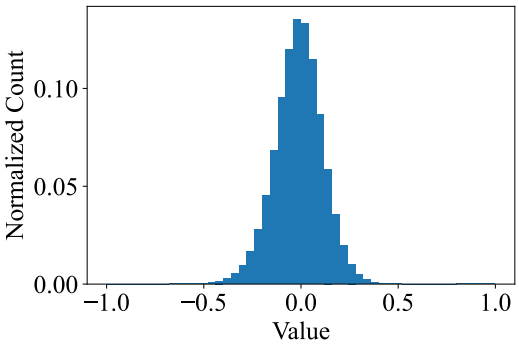


(d) Three secret images.

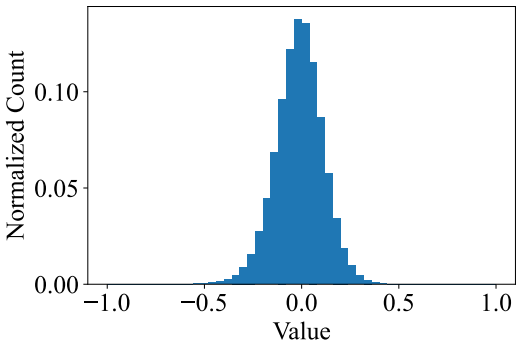


(e) Four secret images.

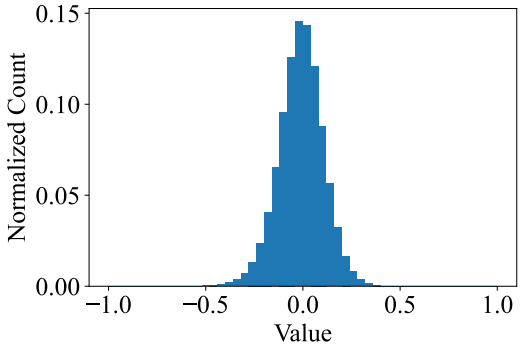
Figure 13: Visual comparison of histograms of the fourth-stage weights.



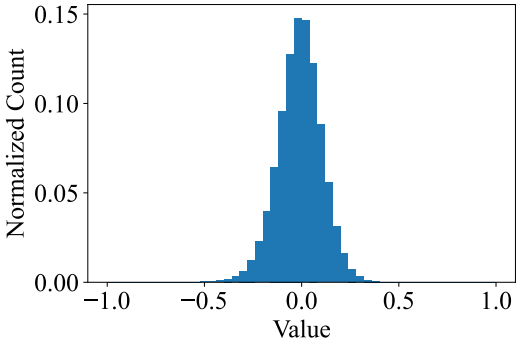
(a) No secret image (original).



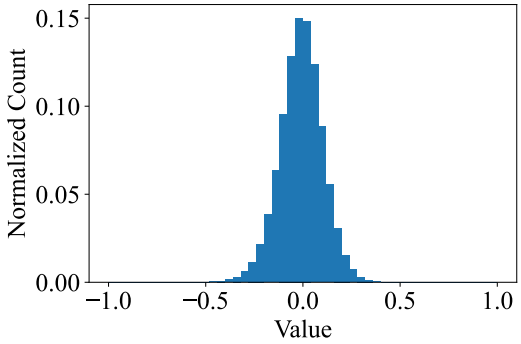
(b) One secret image.



(c) Two secret images.

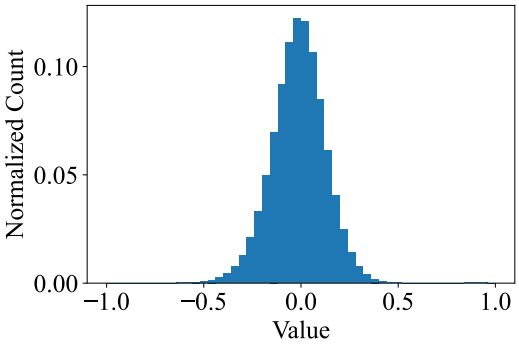


(d) Three secret images.

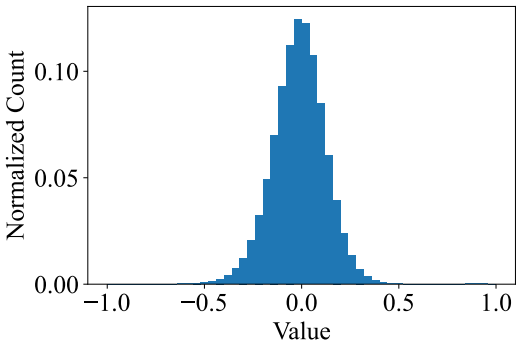


(e) Four secret images.

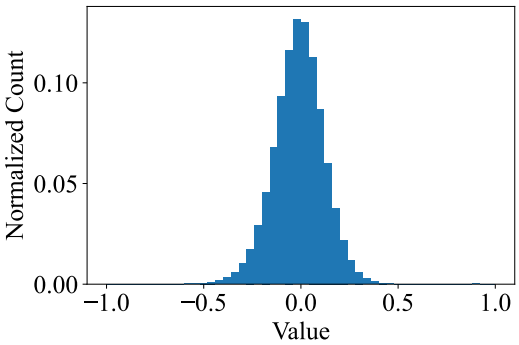
Figure 14: Visual comparison of histograms of the fifth-stage weights.



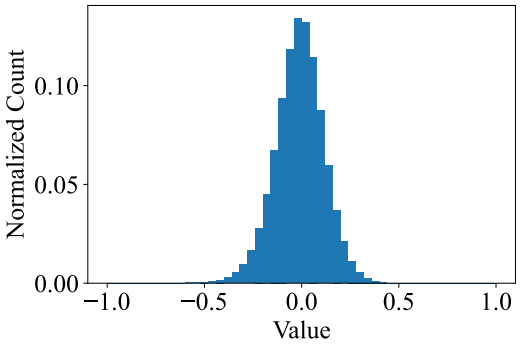
(a) No secret image (original).



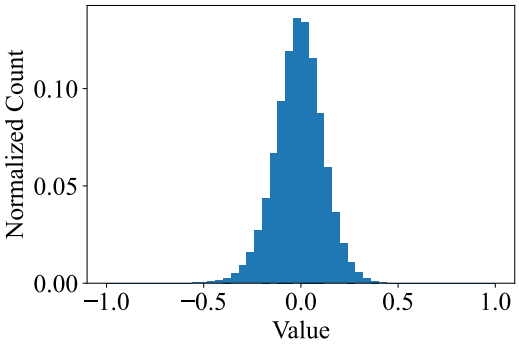
(b) One secret image.



(c) Two secret images.



(d) Three secret images.



(e) Four secret images.

Figure 15: Visual comparison of histograms of the sixth-stage weights.