FIXED NEURAL NETWORK STEGANOGRAPHY: TRAIN THE IMAGES, NOT THE NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent attempts at image steganography make use of advances in deep learning to train an encoder-decoder network pair to hide and retrieve secret messages in images. These methods are able to hide large amounts of data, but also incur high decoding error rates (around 20%). In this paper, we propose a novel algorithm for steganography that takes advantage of the fact that neural networks are sensitive to tiny perturbations. Our method, Fixed Neural Network Steganography (FNNS), achieves 0% error reliably for hiding up to 3 bits per pixel (bpp) of secret information in images and yields significantly lower error rates when compared to prior state of the art methods for hiding more than 3 bpp. FNNS also successfully evades existing statistical steganalysis systems and can be modified to evade neural steganalysis systems as well. Recovering every bit correctly, up to 3 bpp, enables novel applications, e.g. those requiring encryption. We introduce one specific use case for facilitating anonymized and safe image sharing.

1 INTRODUCTION

Image steganography aims to hide a secret digital *message* within a *cover* image (Morkel et al., 2005) — ideally, through minimal alterations, such that only intended recipients are aware of the hidden secret. Steganography has been widely used in applications such as watermarking (Wolfgang & Delp, 1996), copyright certification (Lu, 2004) or private information storage (Srinivasan et al., 2004). Classic steganography tools use image pixel statistics to hide information in images (Pevnỳ et al., 2010). Secret messages hidden with these methods can be recovered with 0% error, but to evade detection by steganalysis tools, they can only hide up to 0.4 bits per pixel (bpp) of information (Pevnỳ et al., 2010; Holub & Fridrich, 2012; Holub et al., 2014). Encouraged by data-driven deep learning techniques, recent works propose training deep encoder-decoder networks to hide and recover a larger number of bits per pixel in image (Zhang et al., 2019a; Zhu et al., 2018; Baluja, 2017; Wu et al., 2018; Hayes & Danezis, 2017). These methods do indeed achieve higher bpp rates, but they also result in high error rates for the retrieved message (Zhang et al., 2019a).

Many applications have stricter requirements for the steganography algorithm. In some scenarios, the hidden message has no redundancy, and there is zero tolerance for even a single incorrectly recovered bit. For example, if the secret message is encrypted, it will be a random bit string that must be recovered 100% intact for successful decryption. In this paper, we propose a novel approach for image steganography that aims to simultaneously achieve high steganographic capacity and reliably low error rates. Notably, we achieve **0.0% error** when encoding up to 3 bpp. We make **no assumptions** on the secret message and allow it to be any arbitrary bit string. We show that our method can be used with randomly initialized neural networks or in conjunction with pre-trained networks.

Unlike previous steganography methods that train deep networks to hide and recover messages in a specific dataset (Zhang et al., 2019a; Zhu et al., 2018; Baluja, 2017), our method is based upon a very different approach, which originated in the context of *adversarial attacks* on neural networks (Szegedy et al., 2013). Adversarial attacks are based on the key insight that deep neural networks are highly sensitive to small changes to the input. An adversary can therefore manipulate an image with imperceptible perturbations to influence the prediction of a neural network that uses this image as input. The last eight years have witnessed an outpouring of analysis (Kurakin et al., 2016b; Carlini & Wagner, 2016; Xu et al., 2017; Shafahi et al., 2018; Meng & Chen, 2017; Li & Li, 2017; Lu et al., 2017a; Graese et al., 2016; Dziugaite et al., 2016; Lu et al., 2017b; Gu & Rigazio,

2015; Kurakin et al., 2016a; Miyato et al., 2015; Nokland, 2015; Cisse et al., 2017; Hu et al., 2019; Guo et al., 2020; 2017) and methods (Liu et al., 2016; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Tramèr et al., 2017; Papernot et al., 2017; 2016; Biggio et al., 2013) to understand and create adversarial perturbations. Notably, it is fair to say that vulnerability to adversarial attacks is generally considered inevitable in most settings (Shafahi et al., 2018) and frustratingly hard to defend against (Carlini & Wagner, 2017; Dziugaite et al., 2016), especially when the target network architecture is known to the adversary (which is generally referred to as *white-box* setting).

Adversarial attacks are typically considered a nuisance, or a limitation of machine learning. However, in this paper we utilize their persistence and reliability as a desired feature for what we refer to as *Fixed Neural Network Steganography (FNNS)*. In a nutshell, FNNS is based on the following procedure (see Figure 1): We initialize a neural network (decoder) that takes as input an image and produces sufficient binary outputs. Given a secret message and a cover image, the sender (Alice) perturbs the original image in a fashion similar to adversarial perturbations (Madry et al., 2017). However, instead of targeting a single prediction bit (e.g. the classification of an image), the sender manipulates thousands or even millions of output bits simultaneously. The intended recipient (Bob) can use the same decoder network and recover the hidden message.

We show that FNNS reliably yields 0% error rate for up to 3 bpp and lower error rates than current state-of-the art methods for higher bit rates on multiple datasets. FNNS can also be used in conjunction with existing trained encoder-decoder methods (like SteganoGAN (Zhang et al., 2019a)) to further reduce the error rates obtained by the trained methods. Additionally, we show that FNNS evades existing statistical steganalysis methods (Boehm, 2014; Dumitrescu et al., 2002a;b; Böhme & Westfeld, 2004; Zhang & Ping, 2003) and can be made resistant to JPEG compression (Wallace, 1992) for low bpp. Finally, we introduce an example application of error-free steganography for anonymized image sharing: We replace faces in images with GAN generated substitutes that contain the original faces encrypted and hidden through FNNS — ensuring that only intended recipients (with the secret key) can recover the original images.

2 Related Work

Statistical image steganography methods (Pevnỳ et al., 2010) typically pre-date the use of neural networks. Least-Significant Bit (LSB) methods modify the lowest-order bits of each image pixel to encode a secret message (Van Schyndel et al., 1994; Wolfgang & Delp, 1996; Katzenbeisser & Petitcolas, 2000). Although compellingly simple and lossless, these methods are easily detectable (Dumitrescu et al., 2002a;b; Böhme & Westfeld, 2004; Zhang & Ping, 2003) and often lack robustness (Qin et al., 2010). Many statistical image steganography methods were proposed to evade detection by stegoanalysis algorithms. Highly Undetectable Steganography (HUGO) (Pevnỳ et al., 2010) use hand crafted features to measure distortion caused by modifying pixels. Pixels that cause the least amount of distortion are then modified. Wavelet Obtained Weights (WOW) (Holub & Fridrich, 2012) use directional high-pass filters to find regions of the cover image with high texture and penalize changes in low textured regions. S-UNIWARD (Holub et al., 2014) is similar to WOW but is designed to work with non-spatial domains (for e.g. frequency domain). The main limitation of statistical methods is that the number of bits they encode is relatively low (≤ 0.4 bpp).

Deep learning image steganography methods have recently achieved impressive results in terms of bpp payload (Zhang et al., 2020; Baluja, 2017; Rahim et al., 2018; Zhang et al., 2019b). In general, they mostly share a similar pipeline that can be trained end-to-end: An encoder network takes as input a cover image and a message that should be concealed within it. From these inputs it generates a steganographic image that has hidden information but is visually similar to the cover image. A subsequent decoder network recovers the hidden message from the steganographic image. Multiple loss functions ensure that 1) the generated image is close to the original one; 2) the decoder's output matches the secret message. Zhu et al. (2018)'s HiddenNet pioneered such an encoder-decoder pipeline, and their HiddenNet can hide up to 0.2 bpp with error rate 10^{-5}. SteganoGAN (Zhang et al., 2019a) introduces an additional critic that ensures the produced image looks realistic, like a natural image, which enables much higher bit rates. The authors show experiments of hiding up to 6 bpp with an error rate of 5-30% (depending on how many bits are hidden). In a similar vein, AdvSGAN (Li et al., 2021) achieves up to 1.0 bpp by learning an image steganography scheme that plays an adversarial game between a restricted neural coder and a critic. Deep Steganography (Baluja,

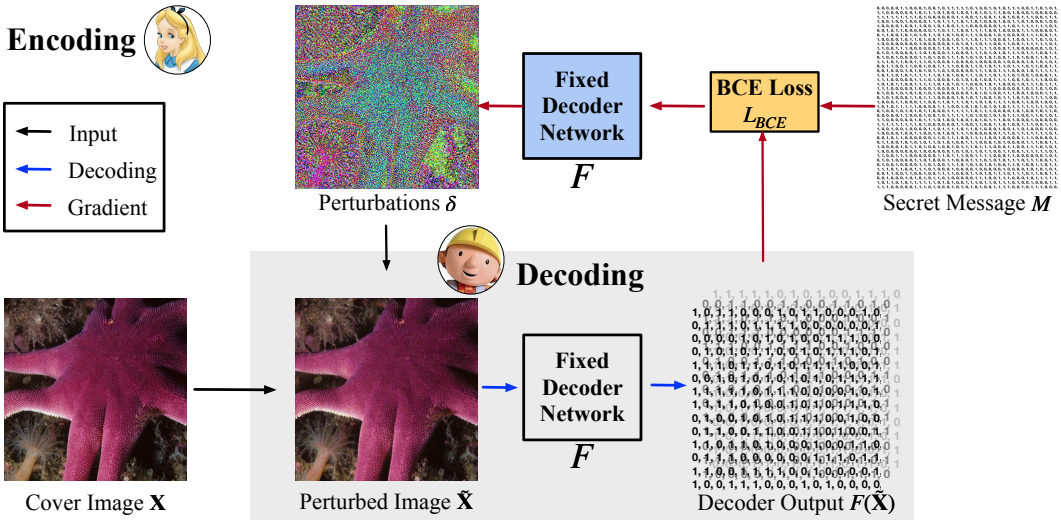

Figure 1: FNNS workflow: Alice (sender) encodes the message $\boldsymbol{M}$ into image $\tilde{\boldsymbol{X}}$ such that $F(\tilde{\boldsymbol{X}}) = \boldsymbol{M}$; Bob (receiver) decodes the message with the same decoder F. Alice generates perturbed image $\tilde{\boldsymbol{X}}$ by optimizing the difference between decoder output $F(\tilde{\boldsymbol{X}})$ and secret message $\boldsymbol{M}$. After FNNS, the perturbed image $\tilde{\boldsymbol{X}}$ is visually indistinguishable from the cover image $\boldsymbol{X}$.

2017), ISGAN (Dong et al., 2018), Attention Based Data Hiding (Yu, 2020) and End-to-end CNN for Image Steganography (Rahim et al., 2018) use similar encoder-decoder architectures to hide and recover structured images instead of random bits. These methods assume the secret message is an image, which allows them to learn image priors that aid in hiding the secret image. Despite handling a large number of bits, these end-to-end neural approaches also share a set of disadvantages: 1) the error rate for the recovered messages is very high, 2) they assume access to hundreds or even thousands of training images from the target domain to train encoder and decoder pairs, and 3) there is little recourse if the model produces an image with high error rate or strong distortions.

Imperceptible image perturbations. Adversarial examples (Goodfellow et al., 2014) are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake. Many approaches (Szegedy et al., 2013; Goodfellow et al., 2014; Guo et al., 2019; Xu et al., 2020; De Palma et al., 2021; Yuan et al., 2019) try to construct adversarial examples by perturbing image pixels. Moosavi-Dezfooli et al. (2017) propose a systematic algorithm for computing universal perturbations that many deep neural networks are highly vulnerable to. Su et al. (2019) propose a differential evolution method to generate low dimensional one-pixel adversarial perturbations that change the output of a classification network. Athalye et al. (2018b) create adversarial examples that are robust to affine image transformations, noise, and other distortions. Projected gradient descent (PGD) (Madry et al., 2017) is one of the most widely used algorithms to generate adversarial examples by adding small perturbations to the input. This method iteratively updates the input with gradient descent until a desired output is obtained. The input is projected, or more precisely clipped to be within $[-\epsilon, \epsilon]$ at the end of every step. This precaution ensures that the perturbations stay reasonably small for all pixels and remain imperceptible. Ghamizi et al. (2019) propose performing stenography by finding perturbations using PGD with a classification network. However, the amount of information they are able to hide is low and they need multiple images to hide long messages.

3 Fixed Neural Network Steganography

Setting. Let $\boldsymbol{X} \in [0,1]^{H \times W \times 3}$ be an RGB color image with height H and width W. Further, let $\boldsymbol{M} \in \{0,1\}^{H \times W \times D}$ be a message that we are trying to conceal in $\boldsymbol{X}$, where D specifies the number of bits we need to hide per pixel.[1] We assume there are two parties involved – the **sender**, Alice,

[1]For convenience we assume the message is of length $H \times W \times D$. In practice, if the message length is not a multiple of $H \times W$ we can simply ignore the unused output bits during optimization.

Algorithm 1 Adversarial Attack for Message hiding

Inputs: decoder network F, cover image $\boldsymbol{X}$, secret message $\boldsymbol{M}$
Hyper-parameters: learning rate $\alpha > 0$, perturbation bound $\epsilon > 0$, optimization steps $n > 0$, max L-BFGS iterations $k > 0$
$\tilde{\boldsymbol{X}} \leftarrow \boldsymbol{X}$
for n iterations **do**
  $\tilde{\mathbf{X}} = LBFGS(F(\tilde{\boldsymbol{X}}), \boldsymbol{M}, L_{\text{BCE}}, k)$ ▷ Take k steps to optimize $L_{\text{BCE}}(F(\tilde{\boldsymbol{X}}), \boldsymbol{M})$.
  $\boldsymbol{\delta} \leftarrow clip_{-\epsilon}^{\epsilon}\{\tilde{\boldsymbol{X}} - \boldsymbol{X}\}$ ▷ Clip pixel value changes exceeding $\pm\epsilon$.
  $\tilde{\boldsymbol{X}} \leftarrow clip_0^1\{\boldsymbol{X} + \boldsymbol{\delta}\}$ ▷ Clip pixel values to $[0, 1]$.
end for
return $\tilde{\boldsymbol{X}}$

who hides M in $\boldsymbol{X}$, creating $\tilde{\boldsymbol{X}}$; and the **receiver**, Bob, who extracts $\boldsymbol{M}$ out of $\tilde{\boldsymbol{X}}$. Given a decoder network $F : [0, 1]^{H \times W \times 3} \rightarrow [0, 1]^{H \times W \times D}$, our goal is to generate a perturbed image $\tilde{\boldsymbol{X}}$, which is close to $\boldsymbol{X}$ (according to metrics specified in the subsequent section) such that $F(\tilde{\boldsymbol{X}}) = \boldsymbol{M}$.

Encoding and Decoding. In order to generate image $\tilde{\boldsymbol{X}}$, we use an approach similar to adversarial perturbations (Madry et al., 2017). Given the cover image $\boldsymbol{X}$ and the ground truth message $\boldsymbol{M}$, the sender (Alice) solves the following optimization problem over the perturbed image $\tilde{\boldsymbol{X}}$:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{X}}} \underbrace{\langle \boldsymbol{M}, \log F(\tilde{\boldsymbol{X}}) \rangle + \langle (1 - \boldsymbol{M}), \log(1 - F(\tilde{\boldsymbol{X}})) \rangle}_{L_{BCE}}, \\ &\text{s.t.} \|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_\infty \leq \epsilon \text{ and } 0 \leq \tilde{\boldsymbol{X}} \leq 1, \end{aligned} \tag{1}$$

where $\langle , \rangle$ denotes inner-product across all $H \times W \times D$ dimensions and the objective function is the binary cross entropy (BCE) loss. The first of the two linear constraints enforces that the maximum perturbation does not exceed a value of ϵ and stays imperceptible. The second constraint enforces that the perturbed image, $\tilde{\boldsymbol{X}}$, is a well-defined image, with pixel values ranging within $[0, 1]$.

Our optimization algorithm is outlined in Algorithm 1, where $clip_0^1(\boldsymbol{x}) = \max(\min(\boldsymbol{x}, 1), 0)$. We use the unconstrained L-BFGS (Fletcher, 2013) algorithm to optimize the objective with respect to $\tilde{\boldsymbol{X}}$. To ensure that we do not violate any of the constraints, we project the solution back into the feasible region after k steps. We assume that the recipient (Bob) has access to the same network F as the sender (Alice). He can then recover the concealed message $\boldsymbol{M}$ by computing $F(\tilde{\boldsymbol{X}})$.

Decoder weights and initialization. Our encoding procedure optimizes the image $\tilde{\boldsymbol{X}}$ directly and considers the weights of F fixed throughout. This gives us the freedom to explore multiple ways to obtain weights for decoder F, yielding three distinct variants of our method:
1) FNNS-R: F is a *random* network and $\tilde{\boldsymbol{X}}$ is initialized to be the cover image $\boldsymbol{X}$. When a random network is used, the sender and receiver only need to share the architecture hyper-parameters of the decoder network, and the random seed used to initialize its weights but the actual weights of the network do not have to be shared. Additionally, if the image quality of the perturbed image is low, a different random decoder network can easily be initialized with a new random seed and the optimization can be repeated.
2) FNNS-DE: Given a trained encoder, decoder pair (from any of the prior work mentioned in section 2), we can define F as such a decoder and initialize $\tilde{\boldsymbol{X}}$ as $Enc(\boldsymbol{X}, \boldsymbol{M})$, where Enc is the trained encoder that is paired with the decoder F. With this initialization, a part of the message $\boldsymbol{M}$ is already encoded into $\tilde{\boldsymbol{X}}$ such that $F(\tilde{\boldsymbol{X}}) \approx \boldsymbol{M}$. As a result, its optimization is much faster. However, the encoding step sometimes deteriorates image quality, and it is hard for the optimization algorithm to "recover" in terms of quality in such cases.
3) FNNS-D: F is a *trained* decoder and $\tilde{\boldsymbol{X}}$ is initialized to be the cover image $\boldsymbol{X}$. With a trained decoder, messages can be hidden in images using both perturbations and by training weights conducive to hiding information in the images. As a result, the output image quality is often better and the optimization does not suffer from getting stuck in bad local optima.

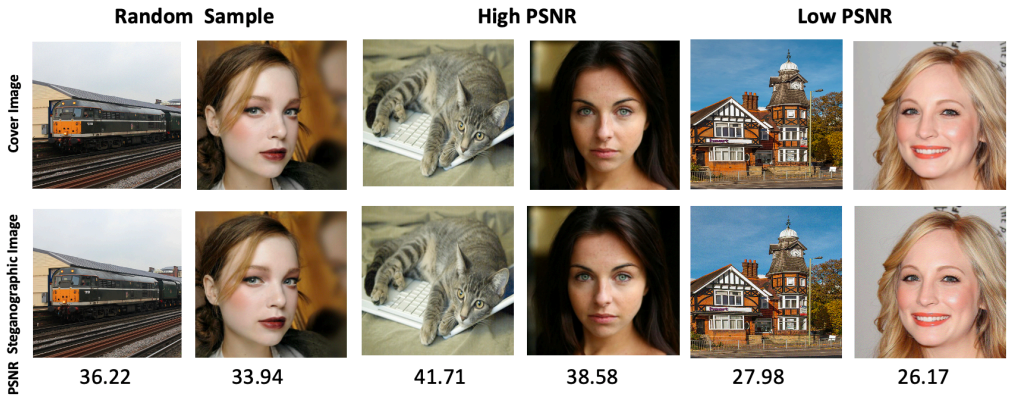

Figure 2: Examples of cover and steganographic image pairs from MS-COCO and CelebA. In each image FNNS-D is used to hide 4 bpp. (Zoom in for image details.)

Decoder architecture. Prior work on adversarial attacks has explored network designs that are especially *robust* to small perturbations (Xu et al., 2020). In contrast, for our decoder we want network architectures that are particularly *susceptible* in that regard. We develop our method based on the *basic* decoder of SteganoGAN (Zhang et al., 2019a), a 4-layer convolutional neural network that takes an $H \times W \times 3$ RGB image as input and outputs a bit string $\{0,1\}^{H \times W \times D}$ (after rounding). We choose SteganoGAN because it has been shown to achieve state-of-the-art performance for hiding arbitrary messages in images. For our experiments, we set the trained encoder and decoder for FNNS-D and FNNS-DE, to be the trained SteganoGAN encoder and decoder. For the random FNNS-R we empirically explore many variations with different depths, widths, normalization layers, and activation functions, and evaluate them w.r.t. bit error rates, PSNR, and SSIM.[2] Due to space constraints, we summarize our results in Appendix B. Throughout, we use 128 hidden channels in FNNS-R and 32 in FNNS-D and FNNS-DE. This is because for trained SteganoGAN models, increasing the number of hidden units leads to no significant improvements in accuracy or image quality (see Appendix C), but slows down the optimization process.

3.1 Evaluation Metrics

We use three popular metrics to evaluate our results. 1. The bit error rate, $\|M - \lfloor F(\tilde{\boldsymbol{X}}) \rceil\|_1$, where $\lfloor \cdot \rceil$ denotes rounding function, measures how many bits are incorrectly recovered. 2. Peak signal-to-noise ratio (PSNR) is a common metric used to measure image distortions between $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ and has been shown to be correlated with human evaluation scores (Isola et al., 2017). 3. Structural Similarity Index (SSIM) is another metric used to measure the similarity between two images. SSIM differs from PSNR in that it tries to capture the change in structural information as opposed to considering absolute errors. PSNR and SSIM between two images $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ with maximum possible pixel value $\max_{\boldsymbol{X}}$, averages $\mu, \tilde{\mu}$, standard deviation $\sigma, \tilde{\sigma}$ and co-variance $\sigma_{\boldsymbol{X}\tilde{\boldsymbol{X}}}$ is defined in Table 1.

$$\mathbf{MSE} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{i=1}^{W} \left[\boldsymbol{X}_{i,j} - \tilde{\boldsymbol{X}}_{i,j} \right]^2$$

$$\mathbf{PSNR} = 20 \log_{10}(max_{\boldsymbol{X}}) - 10 \log_{10}(\mathbf{MSE})$$

$$\mathbf{SSIM} = \frac{(2\mu_{\boldsymbol{X}}\mu_{\tilde{\boldsymbol{X}}} + c_1)(2\sigma_{\boldsymbol{X}\tilde{\boldsymbol{X}}} + c_2)}{(\mu_{\boldsymbol{X}}^2 + \mu_{\tilde{\boldsymbol{X}}}^2 + c_1)(\sigma_{\boldsymbol{X}}^2 + \sigma_{\tilde{\boldsymbol{X}}}^2 + c_2)}$$

Table 1: Metrics to evaluate image quality. Note that c_1, c_2 are small stabilization constants.

4 Steganography Results and Discussion

Experimental Setup We evaluate FNNS on three diverse datasets – a scenic image dataset Div2k (Agustsson & Timofte, 2017), a 2D object detection dataset COCO (Lin et al., 2014) and a hu-

[2]We have also explored the use of fully connected networks but found them to yield far higher error rates, while being slower to be optimized.

Dataset	Method	Error Rate (%) ↓				PSNR ↑				SSIM ↑			
		1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits
CelebA	SteganoGAN	3.94	7.36	8.84	10.00	25.98	25.53	25.70	25.08	0.85	0.86	0.85	0.82
	FNNS-R	0.14	1.80	5.28	15.17	39.79	35.12	33.40	32.53	0.96	0.89	0.84	0.79
	FNNS-D	**0.00**	**0.00**	**0.00**	3.17	36.06	34.43	30.05	33.92	0.87	0.86	0.71	0.84
	FNNS-DE	**0.00**	**0.00**	**0.00**	**2.58**	21.16	20.85	20.67	21.03	0.71	0.68	0.63	0.64
Div2k	SteganoGAN	5.12	8.31	13.74	22.85	21.33	21.06	21.42	21.84	0.76	0.76	0.77	0.78
	FNNS-R	0.02	0.18	3.29	10.88	35.31	30.73	28.99	28.60	0.93	0.85	0.78	0.76
	FNNS-D	**0.00**	**0.00**	**0.01**	5.45	29.30	26.25	22.90	25.74	0.82	0.73	0.53	0.65
	FNNS-DE	**0.00**	**0.00**	**0.01**	**1.75**	18.54	18.02	17.16	17.38	0.60	0.53	0.39	0.60
MS-COCO	SteganoGAN	3.40	6.29	11.13	15.70	25.32	24.27	25.01	24.94	0.84	0.82	0.82	0.82
	FNNS-R	0.04	0.32	2.16	10.38	34.68	30.79	29.32	28.22	0.91	0.84	0.79	0.74
	FNNS-D	**0.00**	**0.00**	**0.00**	13.65	37.94	34.51	27.77	34.78	0.95	0.90	0.72	0.89
	FNNS-DE	**0.00**	**0.00**	**0.00**	**1.74**	22.52	22.35	21.02	21.33	0.77	0.74	0.62	0.64

Table 2: Performance of FNNS and its variants. All values shown are averaged over 100 images. We bold the lowest error rate numbers up to statistical significance.

man face dataset CelebA (Liu et al., 2015). For each dataset, we use the provided test/validation images (if unavailable, we use the first 100 images in the dataset for validation). The hyper-parameters used for Algorithm 1 are as follows: perturbation bound $\epsilon = 0.3$, optimization steps $n = 100$, and L-BFGS iterations $k = 10$ with early stopping if the output has zero error. [3] In cases where the image quality of $\tilde{\boldsymbol{X}}$ is poor, we restart optimization with a different learning rate α. Concretely, we set the learning rate to 0.1 and change it to 0.05 or 0.5 if the output image gets a PSNR lower than 20. We train SteganoGAN models for only one epoch for FNNS-D and FNNS-DE, as we observe that a fully-trained (32 epochs) SteganoGAN decoder over-fits to its training objective such that it's hard to use it for FNNS. Appendix E shows the result of using 32 epochs SteganoGAN models.

Quantitative Comparison. We compare FNNS with SteganoGAN (the current state of the art method) in Table 2. [4] In addition (not shown in the table), we also compare with HUGO (Pevnỳ et al., 2010) and HiDDeN (Zhu et al., 2018). Both methods can achieve an error rate of 0% but only for 0.4 and 0.2 bpp respectively. In contrast, FNNS also achieves 0% error for significantly higher bit rates as shown in Table 2. FNNS-D achieves the best performance in terms of both low error rates and good image quality. FNNS-DE is also able to achieve error rates of 0%, but the corresponding image quality is worse when compared to FNNS-D. From Table 2, we also see that for Div2k it is hard to achieve 0% error with 3 bpp. We find that this is because Div2k is a small dataset with only 800 images and as a result there is not enough diversity to train a flexible SteganoGAN model. This is evidenced by the fact that 0% error for Div2k with 3 bpp can be achieved by using a model trained on MS-COCO.

Qualitative Comparison. Several qualitative examples are presented in Figure 2 (more images from all the method variants can be found in Appendix A). We see that even with 4 bpp, the cover and stenographic images look very similar visually. There are a few examples where the stenographic images look pixelated, especially when a random network is used. This is either because the image is over-optimized to get a low bit error rate or because of a bad random seed. To produce a better stenographic image, the random network can be re-initialized (if using a random network), the optimization can be stopped earlier or the hyper-parameters of FNNS can be changed slightly (in order to change the optimization problem).

Domain independence. The performance of trained encoder-decoder models like SteganoGAN degrade for out-of-domain cover images. For example, a SteganoGAN (Zhang et al., 2019a) model trained on Div2k images (which contains primarily landscapes) produces noticeable artefacts when tested on facial images from CelebA (Figure 3). The only recourse to correcting artifacts or quality would be to train another model with in-domain data. In contrast, FNNS-R is completely do-

Figure 3: Example of artifacts that arise when using an out-of-domain model.

[3] We continue optimizing for 25 steps after achieving 0 error in order to ensure that even after converting $\tilde{\boldsymbol{X}}$ into integer color values (integers between 0 to 255), $F(\tilde{\boldsymbol{X}}) = \boldsymbol{M}$.

[4] Results with higher bpp are shown in Appendix D.

main independent. Although FNNS-D uses a trained decoder, we find it to be just as robust against domain shift. We explain these findings by the fact that the secret message is hidden through an optimization procedure that is very robust to the exact filter values of the decoder network.

Optimization time. Most encoder-decoder deep stenography methods require many hours of training of hundreds of images. However, once trained, it takes less than a second for inference with the model with any given cover image and message. For FNNS-D and FNNS-DE, no network training is required; the optimization for encoding a message in an image at test time takes on average 30 seconds for 1-2bpp and 1min for 3-4bpp. For FNNS-R encoding a message takes about 3 min because optimizing the random network is a harder task.

JPEG Compression. A desirable property for any steganography system is robustness against lossy image compression. Unfortunately the goal of imperceptibility is inherently at odds with most image compression methods, which by design aim to remove imperceptible, and therefore unnecessary, information. JPEG (Wallace, 1992) is a lossy compression method for digital images that transforms an image into frequency space and removes high frequency components via quantization and rounding. Past work has shown that adversarial perturbations can be removed via JPEG compression (Dziugaite et al., 2016) and trained encoder-decoder steganography networks also fail to be resistant to JPEG compression. Modifications to the loss functions can improve resistance to compression, but significantly increase error rates. We can improve the robustness of FNNS to JPEG compression by adding a JPEG layer in our optimization pipeline, in which the back-propagated gradients are approximated with identity transformation (Athalye et al., 2018a). We evaluate on MS-COCO and show results in Table 3. Although we can successfully encode 0.1 bpp at only 0.06% error, it is fair to say that bit-rates beyond 0.5 bpp are currently still out of reach. Alternative approaches that we may explore in the future include adopting the Shadow Attack approach from Ghiasi et al. (2020) to find larger semantically meaningful perturbations and to incorporate global smoothness constraints.

bpp	Error Rate (%)	PSNR	SSIM
0.1	0.06	22.9	0.49
0.2	0.34	22.43	0.48
0.5	5.85	22.18	0.46
1.0	32.03	22.65	0.48

Table 3: Performance of JPEG-resistant FNNS on MS-COCO. FNNS can achieve low error rate with stable image quality when bpp is low.

5 STEGANALYSIS

Steganalysis tools are used to identify whether a message is hidden in an image. Broadly, these tools can be divided into two categories – statistical steganalysis and neural steganalysis. The former makes use of statistical methods and tools to detect steganography, while the latter trains a neural network to distinguish between natural images and steganographic images.

Statistical steganalysis methods were developed to detect LSB (least significant bit) steganography (Gupta et al., 2012) in lossless images such as PNG and BMP. Boehm (2014) presents StegExpose, a library combining many existing steganalysis techniques including Sample Pairs (Dumitrescu et al., 2002b), RS Analysis (Fridrich et al., 2001), Chi Squared Attack (Westfeld & Pfitzmann, 1999), and Primary Sets Dumitrescu et al. (2002a). We show the detection rate of StegExpose for the Div2K dataset in in Table 4. As seen in the table, some images are detected, but the detection rates are low in comparison with traditional methods that are detected with an accuracy of over 75% (Zhu et al., 2018). It is worth noting that even if the image is detected as stegnographic, messages cannot be accessed if they are encrypted.

Method	1 bpp	2 bpp	3 bpp	4 bpp
FNNS	0.08	0.18	0.22	0.31
FNNS-D	0.17	0.13	0.15	0.08
FNNS-DE	0.02	0.03	0.00	0.00

Table 4: Detection Rates obtained by using StegExpose on CelebA

Neural steganalysis makes use of recent advances in deep learning to detect images with hidden information (Ye et al., 2017; You et al., 2020; Lerch-Hostalot, 2021). Neural networks are quite powerful and are capable of accurately detecting steganographic images with bpp as low as 0.5. Assuming that we have white box access to the steganalysis network, we can include its classification score as an auxiliary detection loss in FNNS optimization, and find perturbations such that the neural model does not classify the image as steganographic. We evaluate this method's feasibility and compare results in Table 5. The detection rates are shown in the very right section of the table.

With 2 bpp, FNNS still achieves 0.0% error rates while only 5 out of 1000 images are detected. In

Dataset	Error Rate (%) ↓				PSNR ↑				Detection Rate (%) ↓			
	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits
CelebA	0.00	0.00	0.55	1.35	26.42	19.28	18.66	25.07	24	52	66	93
Div2k	0.00	0.02	0.21	3.95	26.15	19.12	19.23	21.14	37	69	84	99
MS-COCO	0.02	0.01	0.01	13.69	36.27	30.29	18.37	34.87	2	19	49	55

Table 5: Performance of FNNS-D with auxiliary detection loss. We trained a SiaStegNet(You et al. (2020)) on MS-COCO and found that the accuracy is close to 100% on all 3 datasets for steganalysis. We include its steganographic category output as an auxiliary loss in FNNS optimization. While the error rates can not be reduced to 0, they are still comparable to Table 2, and the detection rates are relatively low.

the future, we would like to further investigate the transferability of such steganalysis networks and more robust defenses against them.

6 Application - Face Anonymization

As shown in table 2, we can reliably obtain 0% error rates for hiding up to 3 bpp. The ability to recover the bit string with no errors allows us to encrypt the message, currently the only provably way to preclude unauthorized access of the hidden information. This property enables us to create a protocol where an encrypted message or image is sent from one party to another, and the file is itself an indistinct looking image — allowing users to use photo sharing webpages or social media as a medium of transmission. Below, we describe how FNNS can be used to create a protocol to share images that are anonymized to the public but not to the trusted recipients.

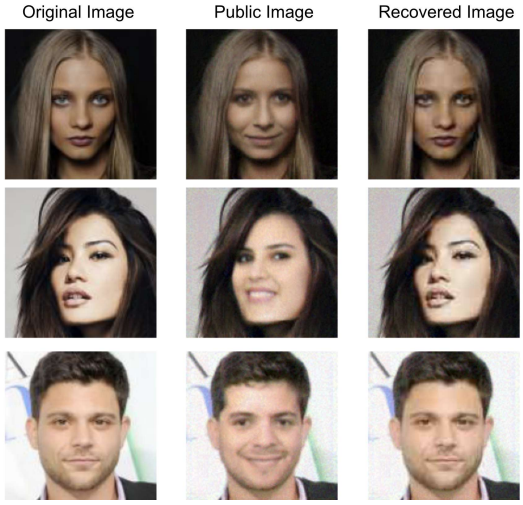

Figure 4: The images in the left column are the original images. The middle column is the result of replacing the face with a fake face with an encrypted form of the original face steganographically hidden through FNNS. The right column shows the image recovered by FNNS decoding the encrypted face, decrypting and re-insertion.

Motivation. Social media platforms have become a central part of our communication, with photo sharing as one of the center activities (e.g. Instagram, Facebook). There are however well documented dangers with such practices. If photos become publicly available, the owner loses control. Snapshots intended to amuse friends or grandparents can "go viral", with traumatic consequences for individuals portrait in the photos. Further, public photos are quickly indexed by image search engines (e.g. `https://clearview.ai/`) and can haunt individuals decades later. We use FNNS to facilitate a mechanism that allows users to share pictures with their friends on social media, but if these are leaked (e.g. through wrong privacy settings on social media pages, or careless behavior), the identity of the people in the images is preserved.

FNNS face anonymization. Current methods for face anonymization either significantly alter the image by using modifications (like blurring or masking) (Neustaedter et al., 2006; Boyle et al., 2000), or remove the sensitive face completely and replace it with a "fake" face (Hukkelås et al., 2019). The former approach can still leak sensitive information and cause the quality of the image to degrade, and the latter approach non-reversibly changes the image. We propose a novel approach for face anonymization using FNNS (see Figure 4). We make use of DeepPrivacy (Hukkelås et al., 2019) to detect faces in the **original image** (left) and replace them with plausible looking GAN-generated faces to obtain an anonymized **public image** (middle image). Note that the original face

Figure 5: Example of anonymizing multiple faces in an image. In this example each private face is hidden within the corresponding fake face that replaces it.

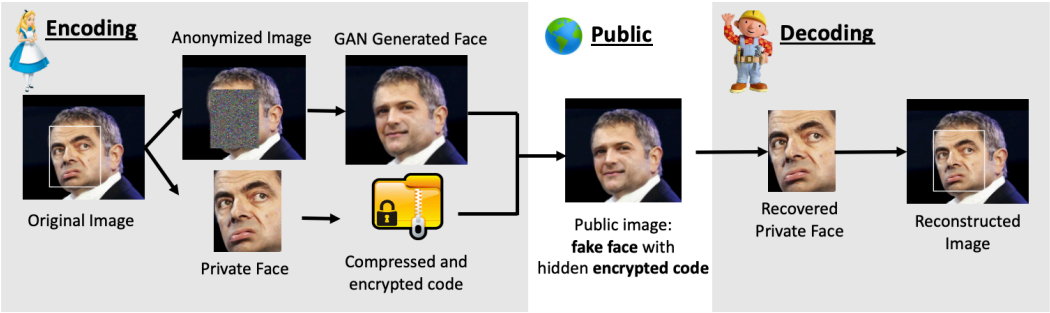

Figure 6: Pipeline for FNNS face anonymization.

is cut before a fake face is generated. As a result, this public image leaks no information about the sensitive face and can safely be shared.

In contrast to prior work, the perfect message reconstruction from FNNS allows us to hide each original private face inside its corresponding fake face in the public image. To ensure the private face is secure, we first compress it (e.g. using JPEG (Wallace, 1992)) and then encrypt it into a cryptographically secure bit string (e.g. using AES (Standard, 2001)). The intended recipient can decode this bit-string, using the decoder network and decrypt the image with a private key to obtain the **recovered image** (right image). Any third party will only observe the public image without any access to private information. See Figure 6 for a layout of the application pipeline. Because all information is hidden inside the faces in the public image, it is naturally compatible with existing image sharing pages, and the encryption / decryption portion can be realized easily in practice e.g. through a simple browser plugin. As each fake face contains all the information required to reconstruct the original private face, this approach can be applied to an arbitrary number of (non-overlapping) faces in an image. An example is presented in Figure 5.

7 Conclusion

In this paper, we propose a novel and effective algorithm for image steganography based on techniques known from the adversarial attacks literature. We show that our method can achieve superior performance while making visually unnoticeable changes to the input image. In contrast to prior work, we are able to achieve error rates of 0% for up to 3 bpp, which in turn enables new applications. In the future we plan to explore other variants and extension of FNNS. One useful variant would be to explore new spaces in which to conduct the optimization. Currently we use RGB space, but other color spaces like YCbCr or non-spatial spaces like the frequency space can be used as well with FNNS. Using different spaces could result in better image quality and could allow for easier encoding of constraints. For instance, to find perturbations robust to JPEG compression, FNNS can be used in only the low frequency region. In the future, we would also like to explore meta-learning algorithms for finding networks that are conducive to FNNS optimization.

Ethics Statement

Steganography is a tool and it's usage depends on the user. Steganography can potentially be used for activities with negative societal impact, such as hidden communication for criminal activities,

or espionage. But it can also be used for beneficial applications like preventing copyright infringements through watermarking, tracing illegal images on social media or anonymizing publicly shared images (as shown in the paper).

REPRODUCIBILITY STATEMENT

For our experiments, we use three publicly available datasets as mentioned in section 4. Our method is outlined in Algorithm 1 and we plan to release our code (with both scripts to obtain the results shown in our experiments and general purpose code to use our method) along with the camera ready version. We have also clearly described all the hyper-parameter values we used to obtain our results in section 4.

REFERENCES

Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 126–135, 2017.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018a.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pp. 284–293. PMLR, 2018b.

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

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proc. ECML*, pp. 387–402, 2013.

Benedikt Boehm. Stegexpose-a tool for detecting lsb steganography. *arXiv preprint arXiv:1410.6656*, 2014.

Rainer Böhme and Andreas Westfeld. Exploiting preserved statistics for steganalysis. In *International Workshop on Information Hiding*, pp. 82–96. Springer, 2004.

Michael Boyle, Christopher Edwards, and Saul Greenberg. The effects of filtered video on awareness and privacy. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 1–10, 2000.

Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, 2016.

Nicholas Carlini and David A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *CoRR*, abs/1704.08847, 2017.

Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Adversarial robustness guarantees for random deep neural networks. In *International Conference on Machine Learning*, pp. 2522–2534. PMLR, 2021.

Shiqi Dong, Ru Zhang, and Jianyi Liu. Invisible steganography via generative adversarial network. *arXiv preprint arXiv:1807.08571*, 4, 2018.

Sorina Dumitrescu, Xiaolin Wu, and Nasir Memon. On steganalysis of random lsb embedding in continuous-tone images. In *Proceedings. International Conference on Image Processing*, volume 3, pp. 641–644. IEEE, 2002a.

Sorina Dumitrescu, Xiaolin Wu, and Zhe Wang. Detection of lsb steganography via sample pair analysis. In *International workshop on information hiding*, pp. 355–372. Springer, 2002b.

Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel Roy. A study of the effect of JPG compression on adversarial images. *CoRR*, abs/1608.00853, 2016.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

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

Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. Adversarial embedding: A robust and elusive steganography and watermarking technique. *arXiv preprint arXiv:1912.01487*, 2019.

Amin Ghiasi, Ali Shafahi, and Tom Goldstein. Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. *arXiv preprint arXiv:2003.08937*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Abigail Graese, Andras Rozsa, and Terrance E. Boult. Assessing threat of adversarial examples on deep neural networks. In *15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, Anaheim, CA, USA, December 18-20, 2016*, pp. 69–74, 2016. doi: 10.1109/ICMLA.2016.0020. URL https://doi.org/10.1109/ICMLA.2016.0020.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR workshop*, 2015.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pp. 2484–2493. PMLR, 2019.

Chuan Guo, Jared S. Frank, and Kilian Q. Weinberger. Low frequency adversarial perturbation. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 1127–1137. PMLR, 22–25 Jul 2020. URL http://proceedings.mlr.press/v115/guo20a.html.

Shailender Gupta, Ankur Goyal, and Bharat Bhushan. Information hiding using least significant bit steganography and cryptography. *International Journal of Modern Education and Computer Science*, 4(6):27, 2012.

Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. *arXiv preprint arXiv:1703.00371*, 2017.

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

Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1–13, 2014.

Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/cbb6a3b884f4f88b3a8e3d44c636cbd8-Paper.pdf.

Håkon Hukkelås, Rudolf Mester, and Frank Lindseth. Deepprivacy: A generative adversarial network for face anonymization. In *International Symposium on Visual Computing*, pp. 565–578. Springer, 2019.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

S Katzenbeisser and FAP Petitcolas. Digital watermarking. *Artech House, London*, 2, 2000.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016a.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016b.

Daniel Lerch-Hostalot. Aletheia, April 2021. URL `https://doi.org/10.5281/zenodo.4655945`.

Boyi Li, Felix Wu, Kilian Q Weinberger, and Serge Belongie. Positional normalization. *arXiv preprint arXiv:1907.04312*, 2019.

Lin Li, Mingyu Fan, and Defu Liu. Advsgan: Adversarial image steganography with adversarial networks. *Multimedia Tools and Applications*, pp. 1–17, 2021.

Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 5775–5783, 2017. doi: 10.1109/ICCV.2017.615. URL `https://doi.org/10.1109/ICCV.2017.615`.

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*, pp. 740–755. Springer, 2014.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Chun-Shien Lu. *Multimedia security: steganography and digital watermarking techniques for protection of intellectual property: steganography and digital watermarking techniques for protection of intellectual property*. Igi Global, 2004.

Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 446–454, 2017a. doi: 10.1109/ICCV.2017.56. URL `https://doi.org/10.1109/ICCV.2017.56`.

Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *CoRR*, abs/1707.03501, 2017b.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 135–147, 2017. doi: 10.1145/3133956.3134057. URL `http://doi.acm.org/10.1145/3133956.3134057`.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *Proc. ICLR*, 2015.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. CVPR*, pp. 2574–2582, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, volume 1, 2005.

Carman Neustaedter, Saul Greenberg, and Michael Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(1):1–36, 2006.

Arild Nokland. Improving back-propagation by adding an adversarial gradient. *CoRR*, abs/1510.04189, 2015.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pp. 582–597, 2016.

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pp. 506–519, 2017.

Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *International Workshop on Information Hiding*, pp. 161–177. Springer, 2010.

Jiaohua Qin, Xuyu Xiang, and Meng Xian Wang. A review on detection of lsb matching steganography. *Information Technology Journal*, 9(8):1725–1738, 2010.

Rafia Rahim, Shahroz Nadeem, et al. End-to-end trained cnn encoder-decoder networks for image steganography. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018.

Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *CoRR*, abs/1809.02104, 2018. URL `http://arxiv.org/abs/1809.02104`.

Yeshwanth Srinivasan, Brian Nutter, Sunanda Mitra, Benny Phillips, and Daron Ferris. Secure transmission of medical records using high capacity steganography. In *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*, pp. 122–127. IEEE, 2004.

NIST-FIPS Standard. Announcing the advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197(1-51):3–3, 2001.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. A digital watermark. In *Proceedings of 1st international conference on image processing*, volume 2, pp. 86–90. IEEE, 1994.

Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

Andreas Westfeld and Andreas Pfitzmann. Attacks on steganographic systems. In *International workshop on information hiding*, pp. 61–76. Springer, 1999.

Raymond B Wolfgang and Edward J Delp. A watermark for digital images. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pp. 219–222. IEEE, 1996.

Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *arXiv preprint arXiv:2010.01279*, 2020.

Pin Wu, Yang Yang, and Xiaoqiang Li. Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet*, 10(6):54, 2018.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017.

Jian Ye, Jiangqun Ni, and Yang Yi. Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, 2017.

Weike You, Hong Zhang, and Xianfeng Zhao. A siamese cnn for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 16:291–306, 2020.

Chong Yu. Attention based data hiding with generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1120–1128, 2020.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12805–12812, 2020.

Kevin Alex Zhang, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Steganogan: High capacity image steganography with gans. *arXiv preprint arXiv:1901.03892*, 2019a. URL `https://arxiv.org/abs/1901.03892`.

Ru Zhang, Shiqi Dong, and Jianyi Liu. Invisible steganography via generative adversarial networks. *Multimedia tools and applications*, 78(7):8559–8575, 2019b.

Tao Zhang and Xijian Ping. Reliable detection of lsb steganography based on the difference image histogram. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 3, pp. III–545. IEEE, 2003.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 657–672, 2018.

A ADDITIONAL QUALITATIVE EXAMPLES

A.1 EXAMPLES FROM FNNS-R

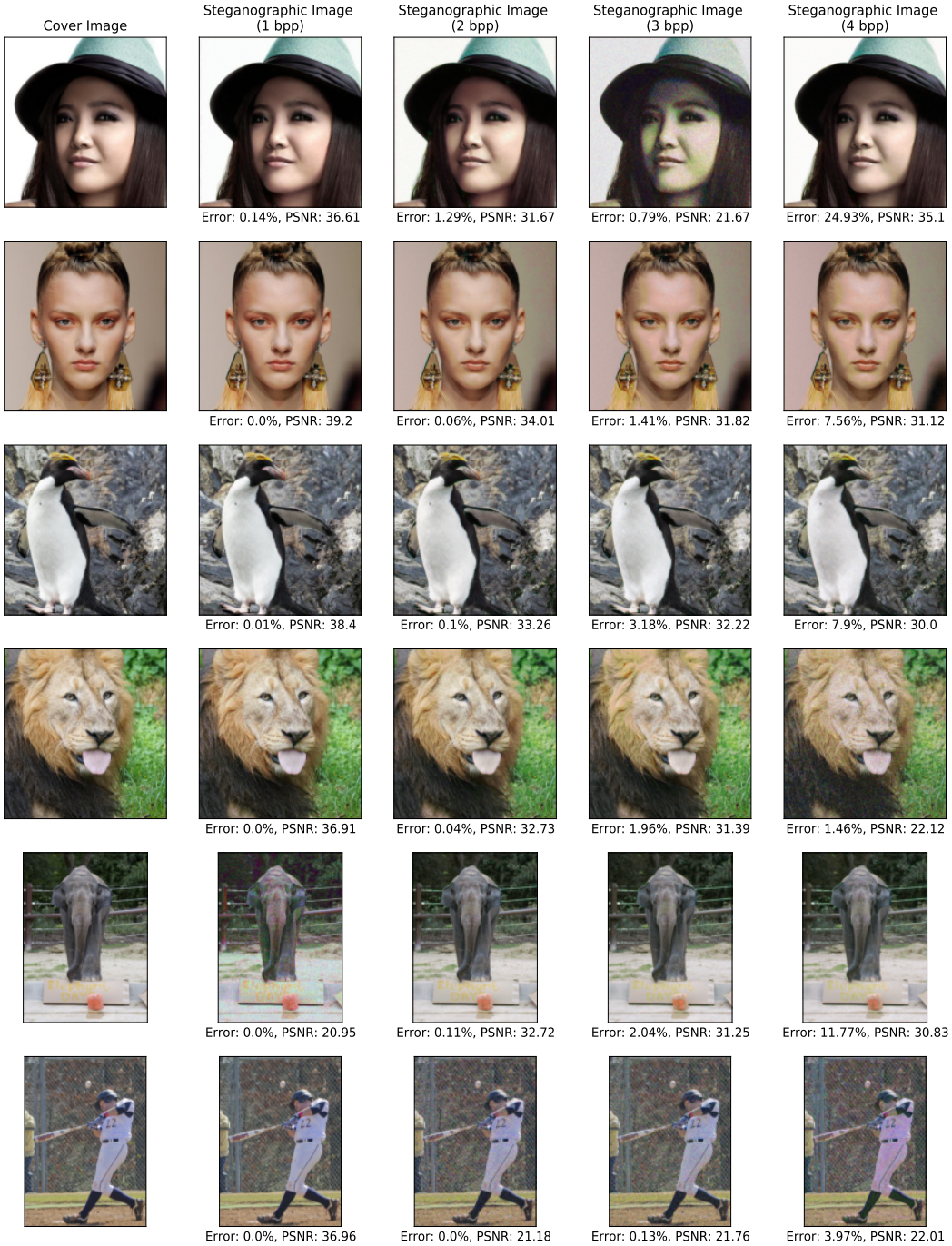

Figure 7: Examples of images with different amounts of hidden information. The first two images are from CelebA, the next two are from Div2k and the last two are from MS-COCO. PSNR values vary mostly due to the randomness of the networks. For images with low PSNR values the quality improves with a re-initialized network.

A.2 EXAMPLES FROM FNNS-D

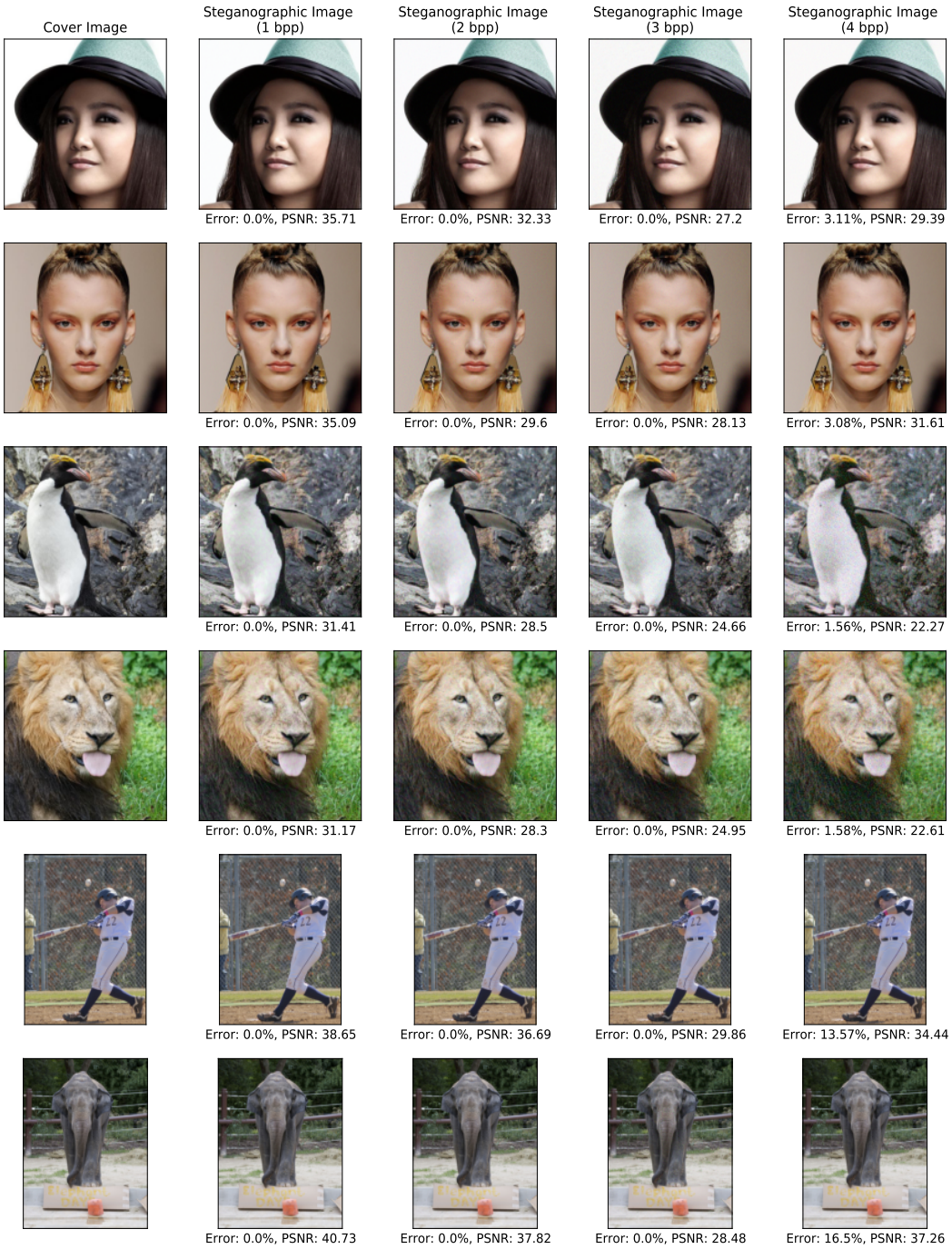

Figure 8: Examples of images with different amounts of hidden information. The first two images are from CelebA, the next two are from Div2k and the last two are from MS-COCO.

A.3 EXAMPLES FROM FNNS-DE

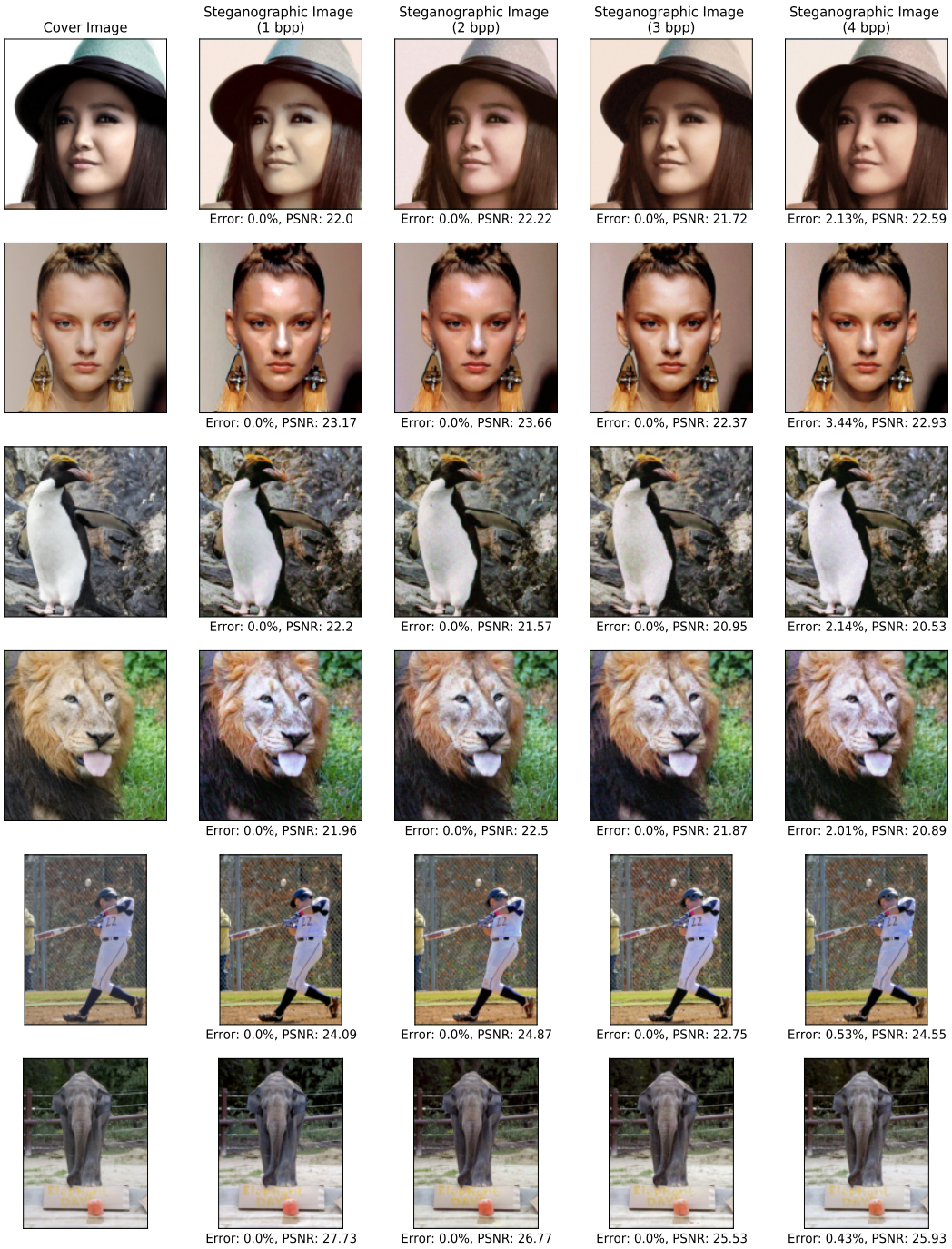

Figure 9: Examples of images with different amounts of hidden information. The first two images are from CelebA, the next two are from Div2k and the last two are from MS-COCO. As seen in row 1 or row 4, the color of the image changes sometimes. This is because the trained Steganogan network learns that information can be hidden by changing the color and maintaining the structure.

B RANDOM DECODER ARCHITECTURE

The decoder introduced in section 3 can be any network that takes an image as input and produces sufficiently many binary outputs, but some architectures are better suited for the task at hand. Unlike most prior work, that explores networks that are robust to adversarial attack Xu et al. (2020), we find ourselves looking for architectures that are susceptible to it. Consequently, in our search for a suitable decoder, we empirically explore many network architectures with different network depth, width, normalization and activation functions and evaluate them w.r.t. pixel error rates, PSNR, and SSIM. For the experiments in this sections, we test different architectures using 100 images from the training set of the Div2K Agustsson & Timofte (2017) dataset, with dimensions $512 \times 512 \times 3$ unless otherwise noted. Since, our input consists of images, we focus on convolutional networks (CNN) Goodfellow et al. (2016).[5]

Figure 10: Error rates (%) of 3 bits based on various widths and depths. We calculate the error rate on average over 100 images from Div2k. The lower the better.

Network depth and width. Figure 10 presents results of convolutional networks with varying height and width, and their respective error rates. Similar to the finding in Wu et al. (2020), we observe that wider networks are less robust and that increasing the width can lead to networks that are more sensitive to perturbations. However, we notice that the opposite is true for depth; the deeper the network, the less sensitive it is to perturbations. Although a width of 256 seems clearly superior, it is fair to say that the optimization is also slower. Finding a perturbed version of a $512 \times 512 \times 3$ image with 3 bpp takes 1 minute for network with a width of 32, but 6 minutes with width 256 (NVIDIA GTX 1080 GPU).

Normalization. As the weights of the decoder network are set randomly, the gradients can have high variance. In Table 6 experiment with tow types of normalization layers that do not require mini-batch statistics. We include Positional Normalization (PONO) Li et al. (2019) and Instance normalization (IN) Ulyanov et al. (2016) in our model. the results show that both normalization methods improve the error rates and both visual metrics, but IN performs the best in this setting. We hypothesize this is because normalization stabilizes the gradients and brings activations near the activation tipping points, which eases the optimization process. Unlike IN, PONO removes the structural statistics Li et al. (2019), which might be informative to create small perturbations.

Normalization	Error Rate(%) ↓	PSNR(dB)↑	SSIM↑
None	36.1 ± 9.5	15.8	0.35
PONO	15.4 ± 8.6	17.7	0.34
IN	2.5 ± 2.7	23.9	0.55

Table 6: Results with different normalizations. We calculate the average across 10 images.

Activation. One important design choice for neural networks are the activation functions. In Table 7 shows results for various popular choices. We observe that LeakyReLu results in better performance when compared to other activations like sigmoid or tanh. The PSNR and SSIM values for sigmoid or tanh are (trivially) high because the optimization fails and very few changes are made

Activation	Error rate (%)↓	PSNR(dB)↑	SSIM↑
None	1.8 ± 3.9	20.9	0.39
ReLU	7.1 ± 9.6	19.3	0.29
LeakyReLU	2.9 ± 4.3	24.7	0.57
Sigmoid	25.3 ± 4.1	39.5	0.96
Tanh	39.4 ± 1.6	45.5	0.95

Table 7: Error rates of different activation functions. We conceal 3 bits for each pixel. The model without any activation function performs best.

[5]We have also explored the use of fully connected networks but found them to yield far higher error rates, while being slower to optimize for.

to the input image. Omitting the activation layer, despite having low error rates, produces images of inferior quality. Adding the ReLU non-linearity makes the optimization easier and more stable.

The different architectures have different tradeoffs in terms of memory, time, and performance. Wider and deeper networks require more time and memory. For our experiments, we settled on four convolutional layers (three intermediate layers and one output layer) with 128 hidden units each, along with instance normalization and ReLU activation.

C STEGANOGAN HIDDEN LAYER SIZE

Table 8 shows difference in performance when using 32 hidden channels vs 128 hidden channels. As seen in the table the difference is quite small. However, using networks with a smaller hidden size speeds up both the training of the network and using it for FNNS. Hence, for FNNS we use networks with 32 hidden channels.

Method	error rate (%)				PSNR				SSIM			
	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits
SteganoGAN - 32 hidden units	5.12	8.31	15.74	22.85	23.53	23.66	22.86	22.87	0.83	0.82	0.8	0.82
SteganoGAN - 128 hidden units	5.97	10.28	18.11	24.93	20.98	21.52	21.54	21.75	0.76	0.76	0.77	0.78

Table 8: Performance of SteganoGAN with 32 hidden units and 128 hidden units.

D HIGHER BITS PER PIXEL

Dataset	Method	Error Rate (%) ↓		PSNR ↑		SSIM ↑	
		5 bit	6 bits	5 bits	6 bits	5 bit	6 bits
Celeba	SteganoGAN	32.15	31.16	19.51	21.82	0.74	0.79
	FNNS-R	14.14	16.27	18.68	17.86	0.18	0.16
	FNNS-D	15.3	18.41	12.94	12.99	0.07	0.07
	FNNS-DE	15.95	17.88	18.22	16.56	0.17	0.13
Div2k	SteganoGAN	31.44	35.35	20.05	20.34	0.79	0.8
	FNNS-R	13.01	15.98	16.71	16.46	0.25	0.26
	FNNS-D	18.12	19.67	12.34	12.3	0.13	0.14
	FNNS-DE	16.03	17.63	14.74	14.94	0.2	0.21

Table 9: Performance of FNNS and its variants with 5-6 bpp. All values shown are averaged over 100 images. SteganoGAN, as a generative model, maintains acceptable image quality while it has a high error rate. FNNS-based methods can achieve lower error rates with its optimization at the cost of poorer image quality. Since the SteganoGAN model cannot encode and decode 5-6 bpp well, FNNS works better with randomly initialized weights.

E OPTIMIZE FULLY TRAINED STEGANOGAN

For FNNS-D and FNNS-DE we used SteganoGAN models trained for 1 epoch as mentioned in section 4. This ensured that the model was flexible enough to achieve 0% error. Table 10 shows the result of using SteganoGAN models trained to completion for 32 epochs. As seen in the table, that the error rate decreases significantly when FNNSD is used without the image quality getting significantly worse. The learning rate was set to 0.5 and the number of optimization steps was set to 200 when using a SteganoGAN model trained for 32 epochs. All other hyper-parameters were the same as listed in section 4.

Dataset	Method	Error Rate (%) ↓				PSNR ↑				SSIM ↑			
		1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits	1 bit	2 bits	3 bits	4 bits
mscoco	SteganoGAN	2.42	4.02	7.76	10.56	27.49	27.04	26.64	26.73	0.88	0.87	0.84	0.85
	FNNS-D	0.00*	0.01	1.31	2.88	27.30	26.98	26.56	26.61	0.86	0.86	0.84	0.84
celeba	SteganoGAN	3.94	7.36	8.84	10.00	25.98	25.53	25.70	25.08	0.85	0.86	0.85	0.82
	FNNS-D	1.00	0.88	2.73	3.38	26.02	25.54	25.77	25.08	0.84	0.85	0.85	0.81
Div2k	SteganoGAN	5.12	8.31	13.74	22.85	21.33	21.06	21.42	21.84	0.76	0.76	0.77	0.78
	FNNS-D	0.00*	1.12	4.37	11.96	21.01	20.98	20.88	21.27	0.71	0.74	0.67	0.68

Table 10: Performance obtained when using a SteganoGAN trained for 32 epochs. In the table, * implies that the value is 0 rounded to two decimal places but it is not exactly 0.