Image-into-Image Steganography Using Deep Convolutional Network

Pin Wu[1][0000−0002−1411−1188], Yang Yang[1][0000−0002−6627−2987] Xiaoqiang Li[1,2][0000−0001−7243−2783]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
[2] Shanghai Institute for Advanced Communication & Data Science, Shanghai University, Shanghai, China
{adamcavendish,xqli}@shu.edu.cn

Abstract. Raising payload capacity in image steganography without losing too much safety is a challenging task. This paper combines recent deep convolutional neural network methods with image-into-image steganography. We show that with the proposed method, the capacity can go up to 23.57 bpp (bits per pixel) by changing only 0.76% of the cover image. We applied several traditional steganography analysis algorithms and found out that the proposed method is quite robust.

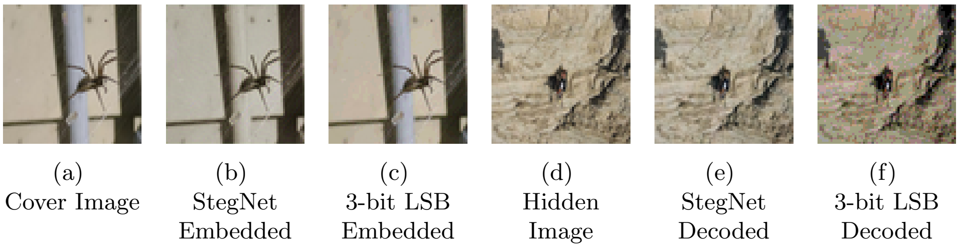
The source code is available at:
https://github.com/adamcavendish/Deep-Image-Steganography

Keywords: Convolutional Neural Network, Image Steganography, Steganography Capacity

1 Introduction

Image steganography, aiming at delivering a modified cover image to secretly transfer hidden information inside with little awareness of the third-party supervision, is a classical computer vision and cryptography problem. Traditional image steganography algorithms go to their great length to hide information into the cover image while little consideration is tilted to payload capacity, also known as the ratio between hidden and total information transferred. The payload capacity is one significant factor to steganography methods because if more information is to be hidden in the cover, the visual appearance of the cover is altered further and thus the risk of detection is higher.

To maximize the payload capacity while still resistible to simple alterations, pixel level steganography is majorly used, in which LSB (least significant bits) method [11], BPCS (Bit Plane Complexity Segmentation) [9], and their extensions are in dominant. LSB-based methods can achieve a payload capacity of up to 50%, or otherwise, a vague outline of the hidden image would be exposed. However, most of these methods are vulnerable to statistical analysis, and therefore it can be easily detected.



(a) Cover Image (b) StegNet Embedded (c) 3-bit LSB Embedded (d) Hidden Image (e) StegNet Decoded (f) 3-bit LSB Decoded

Fig. 1: StegNet and 3-bit LSB Comparison
Embedded-Cover-Difference = 0.76%, Hidden-Decoded-Difference = 1.8%, Payload Capacity = 23.57 bpp

Some traditional steganography methods with balanced attributes are hiding information in the JPEG DCT components. For instance, A. Almohammad's work [1] provides around 20% of payload capacity (based on the patterns) and still remains undetected through statistical analysis.

Most secure traditional image steganography methods recently have adopted several functions to evaluate the embedding localizations in the image, which enables content-adaptive steganography. HuGO [12], WOW and some other content-adaptive methods are quite robust against steganalysis, but they highly depend on the patterns of the cover image, and therefore the average payload capacity can be hard to calculate.

With our work, named after "StegNet", it is possible to raise payload capacity to an average of 98.2% or 23.57 bpp (bits per pixel), changing only around 0.76% of the cover image (See Figure 1), and still robust to statistical analysis.

The payload capacity (23.57 bpp) is calculated from Equation 1–2, and 0.76% is calculated from Equation 3.

$$\text{Decoded Rate} = 1 - \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}|H_{i,j} - D_{i,j}|}{m \times n}, \tag{1}$$

$$\text{Payload Capacity} = \text{Decoded Rate} * 8 * 3 \tag{2}$$

$$\text{Cover Changing Rate} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}|C_{i,j} - E_{i,j}|}{m \times n} \tag{3}$$

where C, H, E, D symbols stand for the cover image (C), the hidden image (H), the embedded image (E) and the decoded image (D) in correspondence, and "8, 3" stands for number of bits per channel and number of channels per pixel respectively.

This paper is organized as follows. Section 2 will describe some recent steganography related works based on deep learning. Section 3 will unveil the secret why neural network can achieve the amount of capacity encoding and decoding images. The architecture and experiments of our neural network is discussed in Section 4 and Section 5.

2 Related Work

Recently there're some works on applying neural networks for steganography, Volkhonskiy's [19] and Shi's [15] work focus on generating secure cover images for image steganography. Work [14] and work [13] applied deep neural networks for steganography analysis. Baluja [2] is working on the same field as StegNet, however, the hidden image is slightly visible on residual images of the generated embedded images, it requires more GPU memory than our work, and it takes longer time to embed.

3 Convolutional Neural Network for Image Steganography

3.1 High-order Transformation

In image steganography, we argue that we should not only focus on where to hide information, which most traditional methods work on, but we should also focus on how to hide it.

Most traditional steganography methods usually directly embed hidden information into parts of pixels or transformed correspondings. The transformation regularly occurs in *where to hide*, either actively applied in the steganography method or passively applied because of file format. As a result, the payload capacity is highly related and restricted to the area of texture-rich part of the image detected by the *handcoded* patterns.

DCT-based steganography is one of the most famous transform domain steganograph. We can consider the DCT process in JPEG lossy compression process as a kind of one-level high-order transformation which works at a block level, converting each 8×8 or 16×16 block of pixel information into its corresponding frequency-domain representation. Even hiding in DCT transformed frequency-domain data, traditional works [1,4] embed hidden information in mid frequency band via LSB-alike methods, which eventually cannot be eluded.

While in contrast, deep convolution neural network makes multi-level high-order transformations possible for image steganography [21], where high-level features use less information to represent complex visual patterns. If the model stores only these high-level features, the steganography capacity can be considerably boosted.

3.2 Trading Accuracy for Capacity

Traditional image steganography algorithms mostly embed hidden information as it is or after applying lossless transformations. After decoding, the hidden information is extracted as it is or after the corresponding detransformations are applied. Therefore, empirically speaking, it is just as file compression methods, where lossless compression algorithms usually cannot outperform lossy compression algorithms in capacity.

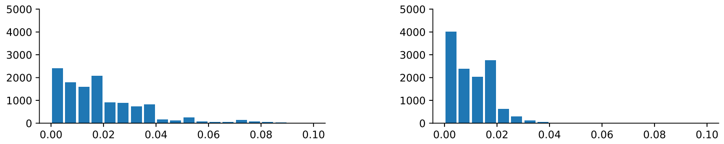
We need to think in a “lossy” way in order to embed almost equal amount of information into the cover. The model needs to learn to compress the cover image and the hidden image into an embedding of high-level features and converts them into an image that appears as similar as the cover image, which comes to the vital idea of trading accuracy for capacity.

Trading accuracy for capacity means that we do not limit our model in reconstructing at a pixel-level accuracy of the hidden image, but aiming at “recreating” a new image with most of the features in it with a panoramic view, i.e. the spider in the picture, the pipes’ position relatively correct, the outline of the mountain, etc.

In other words, the traditional approaches work in lossless ways, which after some preprocesses to the hidden image, the transformed data is crammed into the holes prepared in the cover image. However, StegNet approach decoded image has no pixelwise relationship with the hidden image at all, or strictly speaking, there is no reasonable transformation between each pair of corresponding pixels, but the decoded image as a whole can represent the original hidden image’s meaning through neural network’s reconstruction.

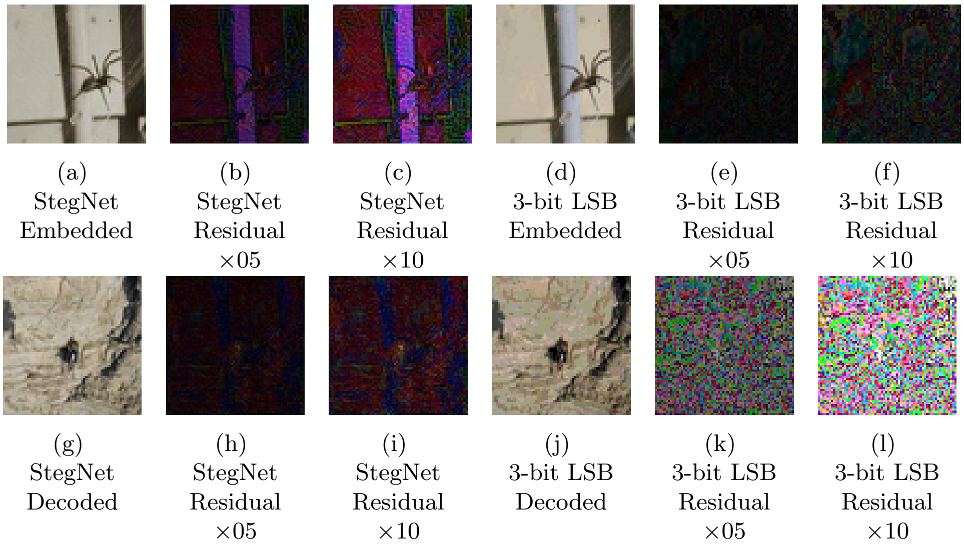
In the encoding process, the model needs to transform from low-level massive amount of pixelwise information into high-level limited sets of featurewise information with understanding of the image, and come up with a brand new image similar to the cover apparently but with hidden features embedded. In the decoding process, to the contrary, the model is shown only the embedded image, from which both cover and hidden high-level features are extracted, and the hidden image is rebuilt according to network’s own comprehension.

As shown in Figure 2 and StegNet part of Figure 3, StegNet is not applying LSB-like or simple merging methods to embed the hidden information into the cover. The residual image is neither simulating random noise (LSB-based approach, see 3-bit LSB part of Figure 3) nor combining recognizable hidden image inside. The embedded pattern is distributed across the whole image, and even magnified 5 to 10 times, the residual image is similar to the cover image visually which can help decreasing the abnormality exposed to human visual system and finally avoid to be detected.



(a) Between Cover and Embedded (b) Between Hidden and Decoded

Fig. 2: Residual image histograms



(a) StegNet Embedded

(b) StegNet Residual ×05

(c) StegNet Residual ×10

(d) 3-bit LSB Embedded

(e) 3-bit LSB Residual ×05

(f) 3-bit LSB Residual ×10

(g) StegNet Decoded

(h) StegNet Residual ×05

(i) StegNet Residual ×10

(j) 3-bit LSB Decoded

(k) 3-bit LSB Residual ×05

(l) 3-bit LSB Residual ×10

Fig. 3: StegNet and 3-bit LSB residual images
"×05" and "×10" are the pixelwise enhancement ratio

The residual image and the magnification operation is computed via

$$R(I_1, I_2) = \frac{|I_1 - I_2|}{\max |I_1 - I_2|}, \quad E(I, M) = \text{clip}(I \cdot M, 0, 1). \tag{4}$$

I_1, I_2 are either cover image and embedded image, or hidden image and decoded image, which are all effectively normalized to $[0, 1]$. Operation $clip(*, 0, 1)$ is to limit the pixelwise enhanced image in range of $[0, 1]$ and M is the magnification ratio, which 5 and 10 are chosen visualize the difference in this paper.

4 Architecture

4.1 Architecture Pipeline

The whole processing pipeline is shown in Figure 4, which consists of two almost identical neural network structure responsible for encoding and decoding. The identical structures can help the neural network to model similar high-level features of images in their latent space. The details of embedding and decoding structure are described in Figure 5. In the embedding procedure, the cover image and the hidden image are concatenated by channel while only the embedded image is shown to the network. Two parts of the network are both majorly made up of one lifting layer which lifts from image channels to a uniform of 32 channels, six 3×3 basic building blocks raising features into high-dimensional latent space and one reducing layer which transforms features back to image space.

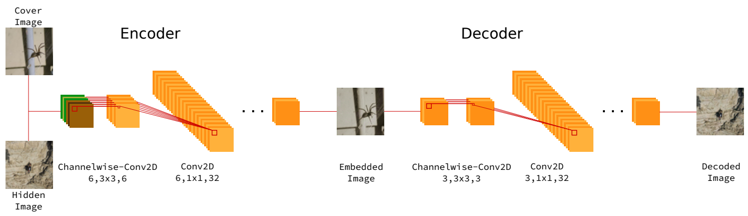
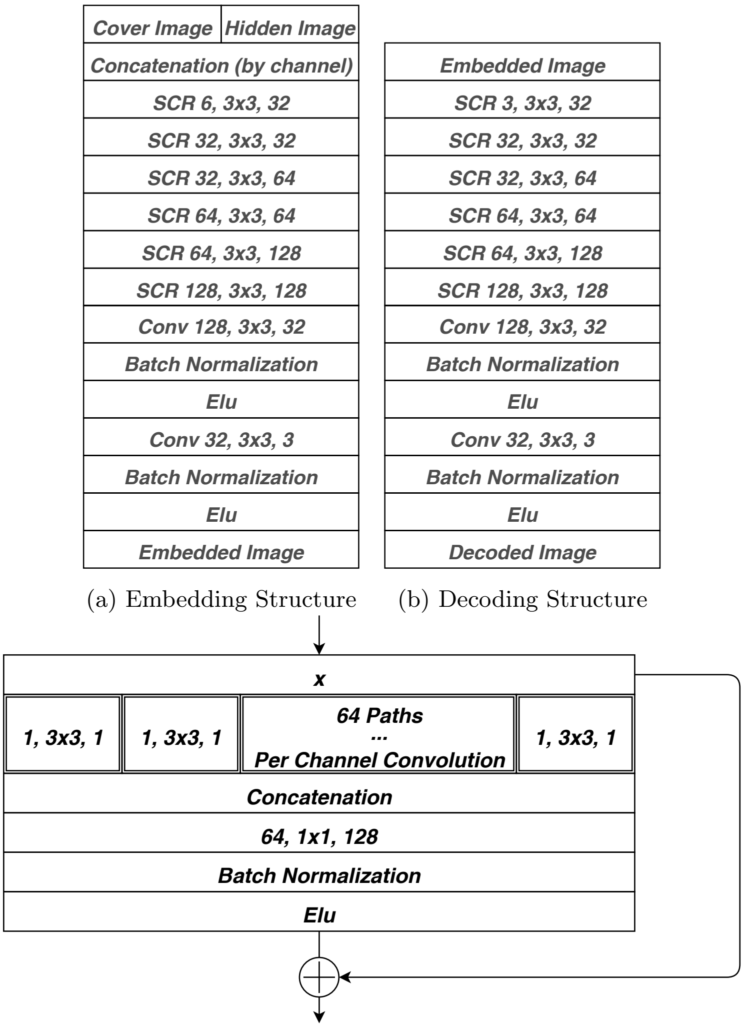


Fig. 4: StegNet Processing Pipeline



(a) Embedding Structure (b) Decoding Structure

(c) SCR Block

Fig. 5: StegNet Network Architecture

The basic building block named "Separable Convolution with Residual Block" (abbreviated as "SCR" in following context) has the architecture as Figure 5. Number of layers and channels are selected trading off between the use of GPU memory, the training time and the final effect. We adopt batch-normalization [8] and exponential linear unit (ELU) [6] for quicker convergence and better result.

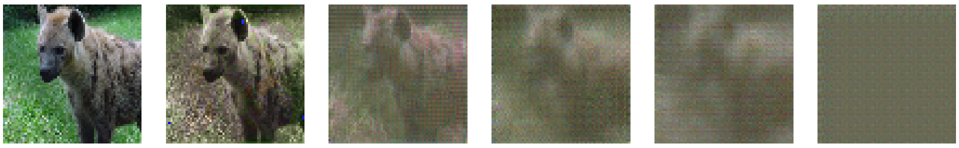
4.2 Separable Convolution with Residual Block

Our work adopt skip connections in Highway Network [16], ResNet [7] and ResNeXt [20], and separable convolution [5] together to form the basic building block "SCR".

The idea behind separable convolution [5] originated from Google's Inception models [17, 18], and the hypothesis behind is that "cross-channel correlations and spatial correlations are decoupled". Further, in Xception architecture [5], it makes an even stronger hypothesis that "cross-channel correlations and spatial correlations can be mapped completely separately". Together with skip-connections [7] the gradients are preserved in backpropagation process via skip-connections to frontier layers and as a result, ease the problem of vanishing gradients.

4.3 Stride or Pooling Impact

StegNet does not use strides like ResNet [7] because strides or poolings have negative impact on embedded and decoded images. We notice that if we apply strides or pooling layers in the steganography model, the spatial information, or the appearance of the image, is gradually lost with more strided layers inserted. See Figure 6-7.



(a) Cover Image

(b) Original StegNet

(c) Strided Layers ×1

(d) Strided Layers ×2

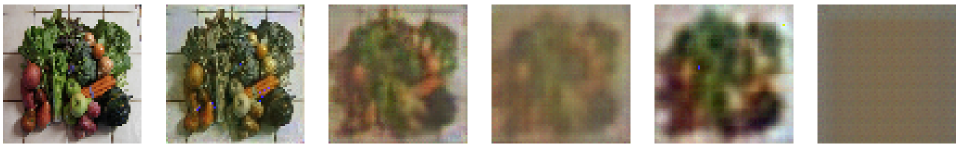
(e) Strided Layers ×3

(f) Global Average Pooling

Fig. 6: Strided Layers' Impact on Embedded Images

The spatial information can be completely destroyed to the utmost with global average pooling applied.

We suppose the impact made by the strided layers is because the noise included by padding and normalization might weight more when the image is scaled down, so it would be a lot harder for the network to reconstruct the original signal through dirty signals.



(a) Hidden Image

(b) Original StegNet

(c) Strided Layers ×1

(d) Strided Layers ×2

(e) Strided Layers ×3

(f) Global Average Pooling

Fig. 7: Strided Layers' Impact on Decoded Images

4.4 Training

Learning the end-to-end mapping function from cover and hidden image to embedded image and embedded image to decoded image requires the estimation of millions of parameters in the neural network. It is achieved via minimizing the weighted loss of L_1-loss between the cover and the embedded image, L_1-loss between the hidden and the decoded image, and their corresponding variance losses (variance should be computed across images' height, width and channel). (See equation 5,6,7)

$$E_i = F_{\mathrm{CE}}(C_i, H_i; \Theta_{\mathrm{CE}}) \qquad D_i = F_{\mathrm{ED}}(E_i; \Theta_{\mathrm{ED}}) \tag{5}$$

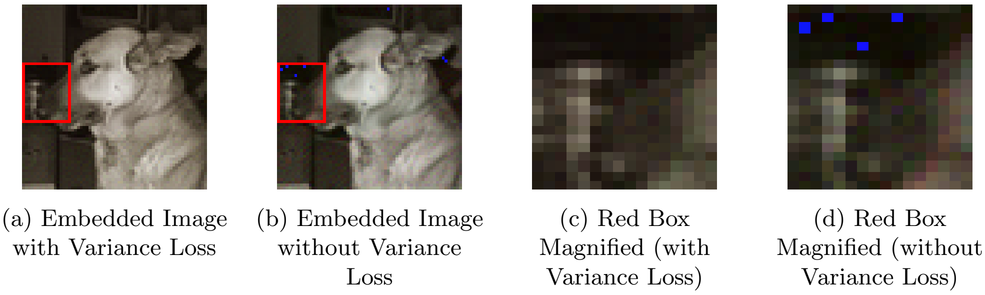
$$L_{\mathrm{CE}} = \frac{1}{n}\sum_{i=1}^{n} |E_i - C_i| \qquad L_{\mathrm{HD}} = \frac{1}{n}\sum_{i=1}^{n} |D_i - H_i| \tag{6}$$

$$\mathrm{Loss} = \frac{1}{4}(L_{\mathrm{CE}} + L_{\mathrm{HD}} + \mathrm{Var}(L_{\mathrm{CE}}) + \mathrm{Var}(L_{\mathrm{HD}})) \tag{7}$$

Here F_{CE} and F_{ED} represents the embedder network and decoder network and their corresponding parameters are noted down as Θ_{CE} and Θ_{ED}.

L_{CE} is used to minimize the difference between the embedded image and the cover image, while L_{HD} is for the hidden image and the decoded image. Choosing only to decode the hidden image while not both the cover and the hidden images is under the consideration that the embedded image should be a concentration of high-level features apparently similar to the cover image whose dimension is half the shape of those two images, and some trivial information has been lost. It would have pushed the neural network to balance the capacity in embedding the cover and the hidden if both images are extracted at the decoding process.

Furthermore, adopting variance losses helps to give a hint to the neural network that the loss should be distributed throughout the image, but not putting at some specific position (See Figure 8 for differences between. The embedded image without variance loss shows some obvious noise spikes (blue points) in the background and also some around the dog nose).



(a) Embedded Image with Variance Loss (b) Embedded Image without Variance Loss (c) Red Box Magnified (with Variance Loss) (d) Red Box Magnified (without Variance Loss)

Fig. 8: Variance Loss Effect on Embedding Results

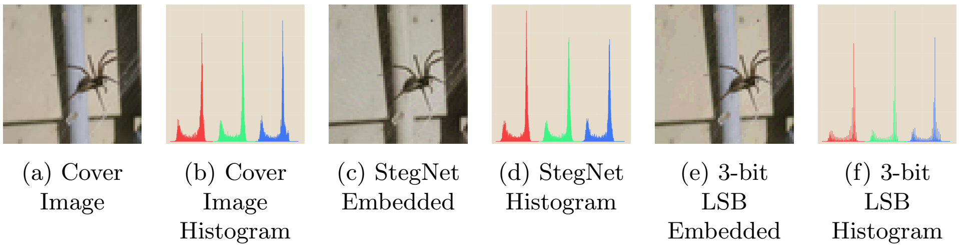
5 Experiments

5.1 Environment

Our work is trained on one NVidia GTX1080 GPU and we adopt a batch size of 64 using Adam optimizer [10] with learning rate at 10^{-5}. We use no image augmentation and restrict model's input image to 64 × 64 in height and width because of memory limit. We use 80% of the ImageNet dataset for training and the remaining for testing to verify the generalization ability of our model.

5.2 Statistical Analysis

The embedded images shown in Figure 1 are visually quite similar, however LSB method embedded image is very fragile to histogram analysis. Figure 9 is a comparison of histogram analysis between LSB method and our work. It shows a direct view of robustness of StegNet against statistical analysis, which the StegNet embedded's histogram and the cover image's histogram are much more matching.



(a) Cover Image (b) Cover Image Histogram (c) StegNet Embedded (d) StegNet Histogram (e) 3-bit LSB Embedded (f) 3-bit LSB Histogram

Fig. 9: Histogram Comparison between StegNet and Plain LSB

A more all-round test is conducted through StegExpose [3], which combines several decent algorithms to detect LSB based steganography. The detection threshold is its hyperparameter, which is used to balance true positive rate and false positive rate of the StegExpose's result. The test is performed with linear interpolation of detection threshold from 0.00 to 1.00 with 0.01 as step. Figure 10 is the ROC curve, where true positive stands for an embedded image correctly identified that there's hidden data inside while false positive means a clean figure falsely classified as an embedded image. The figure is plotted in red-dash-line-connected scatter data, showing that StegExpose can only work a little better than random guessing, the line in green. In other words, the proposed steganography method can better resist StegExpose attack.

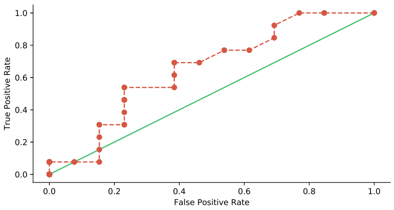


Fig. 10: ROC Curves: Detecting Steganography via StegExpose

6 Conclusion and Future Work

We have presented a novel deep learning approach for image steganography, and the proposed method has achieved superior performance to traditional methods. We tested the proposed method with several steganography analysis methods and proves its robustness.

In spite of the good capacity StegNet provides, there's still some noise generated at non-texture-rich areas in the generated images, i.e., plain white or plain black parts. Therefore, the variance loss adopted by StegNet might not be the optimal solution to loss distribution.

References

1. Almohammad, A., Hierons, R.M., Ghinea, G.: High Capacity Steganographic Method Based Upon JPEG. The Third International Conference on Availability, Reliability and Security (2008)
2. Baluja, S.: Hiding Images in Plain Sight: Deep Steganograph, p. 2069–2079. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/6802-hiding-images-in-plain-sight-deep-steganography.pdf
3. Boehm, B.: StegExpose - A Tool for Detecting LSB Steganography. arXiv e-prints (2014), arXiv: 1410.6656

4. Chang, C.C., Chen, T.S., Chung, L.Z.: A steganographic method based upon jpeg and quantization table modification. Information Sciences **141**(1–2), 123–138 (2002)
5. Chollet, F.: Xception: Deep Learning with Depthwise Separable Convolutions. arXiv e-prints (2016), arXiv:1610.02357
6. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). arXiv e-prints (2015), arXiv:1511.07289
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2016)
8. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv e-prints (2015), arXiv:1502.03167
9. Kawaguchi, E., Eason, R.: Principle and applications of BPCS-Steganography. Proc. SPIE 3528, Multimedia Systems and Applications, 464 (1998)
10. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv e-prints (2014), arXiv:1412.6980
11. Mielikainen, J.: Lsb matching revisited. IEEE signal processing letters **13**(5), 285–287 (2006)
12. Pevný, T., Filler, T., Bas, P.: Using High-Dimensional Image Models to Perform Highly Undetectable Steganography, pp. 161–177. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2010)
13. Sedighi, V., Fridrich, J.: Histogram layer, moving convolutional neural networks towards feature-based steganalysis. Electronic Imaging **2017**(7), 50–55 (2017)
14. Sharifzadeh, M., Agarwal, C., Aloraini, M., Schonfeld, D.: Convolutional neural network steganalysis's application to steganography. arXiv:1711.02581 [cs] (Nov 2017), http://arxiv.org/abs/1711.02581, arXiv: 1711.02581
15. Shi, H., Dong, J., Wang, W., Qian, Y., Zhang, X.: Ssgan: Secure steganography based on generative adversarial networks. arXiv:1707.01613 [cs] (Jul 2017), http://arxiv.org/abs/1707.01613, arXiv: 1707.01613
16. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway Networks. arXiv e-prints (2015), arXiv: 1505.00387
17. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv e-prints (2016), arXiv:1602.07261
18. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. arXiv e-prints (2014), arXiv:1409.4842
19. Volkhonskiy, D., Nazarov, I., Borisenko, B., Burnaev, E.: Steganographic generative adversarial networks. arXiv:1703.05502 [cs, stat] (Mar 2017), http://arxiv.org/abs/1703.05502, arXiv: 1703.05502
20. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated Residual Transformations for Deep Neural Networks (2017)
21. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks **8689**, 818–833 (Nov 2013). https://doi.org/10.1007/978-3-319-10590-1_53