Anti-forensics of JPEG Compression Using Generative Adversarial Networks

Yingmin Luo Hanqi Zi Qiong Zhang Xiangui Kang[1]

***Abstract*—JPEG compression is one of the most popular image compression methods. The manipulation history of JPEG compression provides evidence of operational information about the device or software utilized to generate an image. Besides, JPEG compression introduces complex compression artifacts, particularly the blocking artifacts, ringing effects and blurring, which lower the image visual quality. Therefore, erasing the traces left by JPEG compression is of great importance. To solve this problem, we present a JPEG compression anti-forensic method adopting the framework of generative adversarial networks (GANs). This architecture consists of a generator and a discriminator, where the generator can automatically learn how to hide the traces left by JPEG compression during the optimization process against the discriminator. Through extensive experiments, it is demonstrated that the anti-forensically modified images generated by our method can deceive the existing JPEG compression detectors and have very good visual quality.**

Keywords—JPEG compression, anti-forensics, generative adversarial networks

I. Introduction

Image anti-forensics is a newly developed technology which aims to eliminate or obscure the traces of digital image manipulations with corresponding post-processing operations. With such techniques, image manipulation detectors can be deceived and their forensic performance can be greatly reduced.

One type of image processing operation with particular forensic significance is JPEG compression, which is one of the most popular image compression formats today. Most digital images are subjected to JPEG compression, either by the camera used to capture, during image storage, or for the purposes of digital transmission over the Internet. Once the evidence of JPEG compression has been detected, the quantization table used during compression can be estimated [1]. Most digital cameras and image editing software use proprietary JPEG quantization tables when compressing an image, of which the origin can be identified by matching the quantization tables used to compress the raw image [2].

As we know, JPEG compression may leave blocking artifacts, ringing effects, blurring, quantization artifacts in DCT frequency domain and some other traces in compressed images. Based on these traces, a few JPEG compression forensic techniques have been proposed. Two JPEG compression detectors were proposed in [1] based on compressed images' quantization artifacts in the DCT frequency domain and blocking artifacts in the spatial domain respectively. Recently, [3] [4] introduced a new form of convolutional neural networks to detect JPEG compression and other image manipulations, which achieved state-of-the-art forensic performance. As for anti-forensics of JPEG compression, [5] proposed a method through introducing noise dither to each quantized DCT coefficient to remove the quantization artifacts. The same authors disguised both of the blocking artifacts and the quantization artifacts in their extension work [6].

Different from previous anti-forensics technique of manually hiding traces left by JPEG compression, we propose an anti-forensic framework of JPEG compression, using a generative adversarial network (GAN) [7]. GAN is comprised of two networks: a generator and a discriminator. Generative adversarial networks have been applied to different image-to-image translation problems, such as super resolution [8], style transfer [9] and others. In this work, we model JPEG compression anti-forensic problem as an image-to-image translation problem for the following two reasons. On one hand, the discriminator can be regarded as a JPEG compression detector, while the generator is an anti-forensic tool, from which we can transfer a JPEG compressed image into a modified one to deceive the forensic detector. On the other hand, the generator can automatically learn how to hide the traces left by JPEG compression during the optimization process. The contributions of this work are summarized as follows:

1. We train a network to generate anti-forensically modified images which have the ability to deceive a JPEG compression detector without degrading image quality.

2. Extensive experimental results show that our method has advantage over several traditional JPEG anti-forensic methods.

II. The Proposed GAN Architecture

Our proposed GAN model is made up of two deep convolutional neural networks: a generator and a discriminator. After iterative trainings, our framework can generate anti-forensically modified images with good visual quality and reasonable statistical characteristics.

The detailed GAN architectures are shown in Fig. 1, where the generator *G* receives a JPEG compressed image as initial input and attempts to erase the traces left by compression operation. The discriminator *D*, as the opponent of *G*, tries to

[1]This work was supported by NSFC (Grant Nos. U1536204, 61772571, 61702429), and the special funding for basic scientific research of Sun Yat-sen University (Grant No. 6177060230). Yingmin Luo, Hanqi Zi, QiongZhang and Xiangui Kang are with Guangdong Key Laboratory of Information security Technology, School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China. Correspondence author: Xiangui Kang, isskxg@mail.sysu.edu.cn.

detect whether an image is uncompressed or generated, which can measure the gap between an uncompressed image and a generated image.

A. Generator

As mentioned above, the generator *G* is used to generate anti-forensically modified images.

To generate modified images that have good visual quality and similar statistical characteristics to uncompressed images, we design a generator structure inspired by the generative adversarial network for image super resolution (SRGAN) [8].

As shown in Fig. 1. (a), JPEG compressed images are input into a series of groups. The first group includes a convolutional layer with 3×3 kernels and 64 feature maps, followed by a leaky rectified linear unit (LeakyReLU). LeakyReLU has been proved to be beneficial to GAN framework for the stability of the training process [10]. Then, we deploy 16 identical residual blocks. Each residual block is made up of same components: two convolutional layers with 3×3 kernels and 64 feature maps, followed by Batch-normalization (BN) layers and LeakyReLU as the activation function. Besides, we place skip connections on each block's output, which are added into the output of the second BN layer of next block. After 16 residual blocks, another three convolutional layers are deployed. It is worth noting that the size of feature maps keeps the same in each convolutional layer with the use of 1×1 stride. Specifically, the pixel values of an input image are scaled to $[-1,1]$ and we place a Hyperbolic Tangent (Tanh) function after the last convolutional layer.

B. Discriminator

The discriminator *D* is designed to distinguish between a generated image and an uncompressed image.

Here, we design a relatively complex architecture for *D*, which is competitive against the generator in the game.

Considering that the high frequency information of images is suppressed by JPEG compression and the low frequency part of JPEG compressed images is similar to that of original ones. Thus, we make the input of the discriminator contain high-frequency information to capture statistical differences between a generated and an uncompressed image. Specifically, we stack the input image with the filtering output of a high-pass filter repressing the image content and preserving the high-frequency part which has been used in image steganalysis [11]. The output of the high-pass filter can be expressed as follows:

$$\boldsymbol{r} = \boldsymbol{x} * \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix}, \tag{1}$$

where $\boldsymbol{x}$ represents the input image and $*$ means the convolution. Therefore, the input image and it's high-frequency residual construct the input data of the discriminator.

After the pre-processing of input data, we continuously apply three groups with the same function layers. Each group contains two consecutive convolutional layers with 3×3 kernels and 1×1 strides, and ends with a 4×4 Max-pooling layer using the stride of 4×4; each convolutional layer is followed by a BN layer and a RELU. We increase the number of convolutional kernels for each group by a factor of 2. Finally, the output of the previous group is fed into a 64 neurons fully-connected layer with ReLU as the activation, then followed by a one neuron fully-connected layer with Sigmoid function as the activation. The discriminator's architecture is shown in Fig. 1. (b).

III. Design of Loss Functions

We aim to anti-forensically modify a JPEG compressed image so that it has the ability to deceive a JPEG compression detector. To achieve this, we deploy the generative adversarial framework [7]. In principle, we are supposed to simultaneously train the two networks by optimizing the following minimax problem:

$$min_{\theta_G} max_{\theta_D} \mathbb{E}_{x}[log D_{\theta_D}(\boldsymbol{x})] + \mathbb{E}_{x'}\left[log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{x'})\right)\right)\right], \tag{2}$$

where $D_{\theta_D}(\cdot)$ and $G_{\theta_G}(\cdot)$ represent the outputs of the discriminator and the generator respectively, θ_D and θ_G are learnable parameters for the discriminator and generator respectively, while $\boldsymbol{x}$ and $\boldsymbol{x'}$ represent uncompressed images and JPEG compressed images respectively. However, the loss function of the generator should be changed to be more specific for our JPEG compression anti-forensic task.

A. Loss of Generator

Our ultimate goal is to train a steady and strong generator *G*, which can erase the JPEG compression traces, producing anti-forensically modified images in good visual sense and statistically close to uncompressed images. Catering to the anti-forensic task, the loss function of generator *G* can be modeled as follows:

$$L_G = \mathbb{E}_{x'}\left[\alpha L_G^{pixel} + \beta L_G^{vgg} + \gamma L_G^{adv}\right], \tag{3}$$

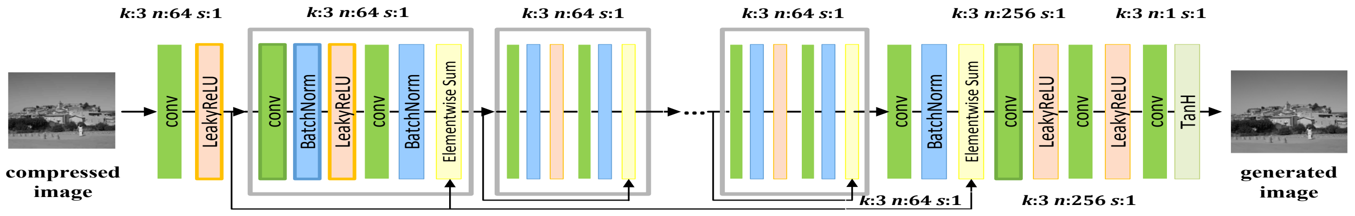
where L_G^{pixel}, L_G^{vgg}, and L_G^{adv} represent respectively pixel-wise l_1 loss, perceptual loss using the pre-trained VGG-16 network [12], and adversarial loss; α, β and γ respectively represent the pre-defined weights of each loss. The combination of these three loss terms determines the actual network's loss function, and the training process of the generator is to find θ_G to minimize L_G.

Pixel-wise loss. Given an uncompressed image $\boldsymbol{x}$ and its corresponding JPEG compressed image $\boldsymbol{x'}$, both with the size of $W \times H$, the pixel-wise loss L_G^{pixel} is calculated as follows:

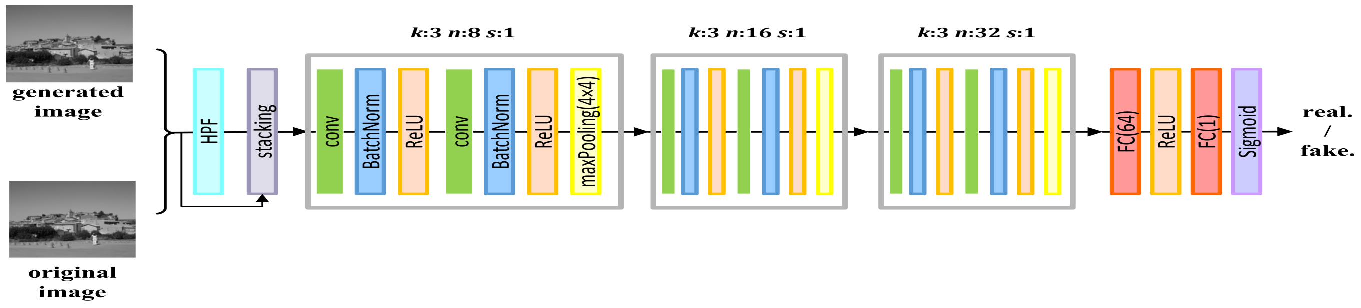
$$L_G^{pixel} = \frac{1}{N}\sum_{i=1}^{N}|\boldsymbol{x_i} - G(\boldsymbol{x'})_i|, \tag{4}$$

where the subscript i refers to the pixel index of an image, $N = W \times H$, and $G(\boldsymbol{x'})$ denotes a generated image.

Pixel-wise loss reflects the differences between pixels in the uncompressed image and the corresponding pixels in the generated image, which is one of the most common loss functions for image content [13]. It can directly monitor the visual quality, but it struggles to reflect lost high-frequency details such as textures and conduct the generator to erase JPEG compression's visual traces completely. In this case, we



(a)



(b)

Fig. 1. The proposed network architectures for JPEG compression anti-forensics. (a) generator. (b) discriminator. *k*, *n*, and *s* respectively denote the kernel size, the number of kernels and stride of each convolutional layer

deploy perceptual loss additionally.

Perceptual loss. The perceptual loss can represent image textures more effectively and help produce an image that is more visually realistic. It is a simple l_2 loss, but based on the differences of CNN feature maps of the generated image and the corresponding uncompressed image [14] :

$$L_G^{vgg} = \frac{1}{N}\sum_{i=1}^{N} \left\| \emptyset(\boldsymbol{x})_i - \emptyset\left(G(\boldsymbol{x}')\right)_i \right\|_2^2, \tag{5}$$

where $\emptyset(\cdot)$ represents the feature map obtained by the 12th convolution (after activation) within the VGG-16 network, pre-trained on ImageNet [15], while i and N respectively refer to the element index and the total number of elements in the feature maps. Perceptual loss is a good complement to pixel-wise loss, both of them help produce images having good visual quality. To produce images having similar statistical characteristics to uncompressed images, we should deploy adversarial loss.

Adversarial loss. The discriminator *D* is assumed to be a detector of JPEG compression. From the perspective of the generator, we wish the images generated by *G* can deceive the discriminator *D*. Thus, we have the adversarial loss L_G^D as:

$$L_G^{adv} = log\left(1 - D\left(G(\boldsymbol{x}')\right)\right), \tag{6}$$

where $D(\cdot)$ represents the output of D.

B. Loss of Discriminator

The discriminator is an opponent of the generator, the generator tries to create images to deceive the discriminator, while the discriminator tries to distinguish a generated image from an uncompressed one. Through continuous iterative trainings, *G* and *D* contend against with each other and directly obtain the feedback from the rival, updating their own parameters to achieve better performance. In our proposed framework, *D* is a CNN-based JPEG compression detector, designed to be a binary classifier to detect whether an image is uncompressed or generated. Thus, we apply the traditional loss function [7] to the discriminator:

$$L_D = \mathbb{E}_x\left[-logD_{\theta_D}(\boldsymbol{x})\right] + \mathbb{E}_{x'}\left[-log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{x}')\right)\right)\right] . \tag{7}$$

The training process of the discriminator is to find θ_D to minimize L_D.

IV. Experimental Results

A. Experimental Setups

We implemented the proposed networks using TensorFlow framework [16] and trained them on a machine equipped with GPU of NVIDIA GTX TITAN XP. In all experiments, we deployed two image datasets, BossBase V1.01 [17] and BOWS2-Original [18]. Each dataset contains 10,000 uncompressed grayscale images with size of 512×512. We first mixed two datasets and then randomly divided it into two sets, the training set and the testing set. The training set contains 16,000 images and the testing set contains 4,000 images. To increase the number of training samples, we cropped five non-overlapped parts with size of 128×128 from each image in the training set. Therefore, we had 80,000 training image samples and 4,000 testing image samples in total. We used all training samples and it's compressed ones for the training process, and compressed all testing samples for the following experiments.

Although we use images with the size of 128×128 to train our networks, the generator can accept images with any size. Without loss of generality, we conducted our experiments on testing images with four resolutions of 64×64, 128×128, 256×256 and 512×512, which are cropped from the central part of images in the testing set. Additionally, we conducted our experiments on JPEG compressed images obtained with three quality factors of 25, 50 and 75 to prove the effectiveness of our method. It's worth noting that, as the VGG-16 network was trained on colored images, we extended every grayscale image to three channels by simple duplication and then preprocessed it as that in [12] before feeding it into the VGG-16 network.

During the training process of our networks, we set the batch size to 16 images for training the generator. Before training the actual GAN, we first trained the generator with learning rate of 5.0×10^{-4}, and weight terms of $\alpha = 1.0$, $\beta = 0.0$ and $\gamma = 0.0$ for two epochs. Next, the generator was trained with the weight terms $\alpha = 1.0$, $\beta = 1.0 \times 10^{-6}$ and $\gamma = 1.0 \times 10^{-2}$ for 50 epochs. On the other hand, we trained the discriminator with 32 images as a batch consisting of 16 generated images and their corresponding uncompressed images. Specifically, we train the generator for one iteration after each training iteration of the discriminator. The leak slope of LeakyReLUs was set to 0.2, which were used in the generator. We used Adam [19] as optimizers with $\beta_1 = 0.9$, $\beta_1 = 0.999$ and $\varepsilon = 10^{-8}$ for both of the generator and the discriminator respectively. We set the learning rates as 5.0×10^{-4} and 5.0×10^{-6} respectively for the generator and the discriminator for the first 30 epochs and divided them by 10 for the last 20 epochs.

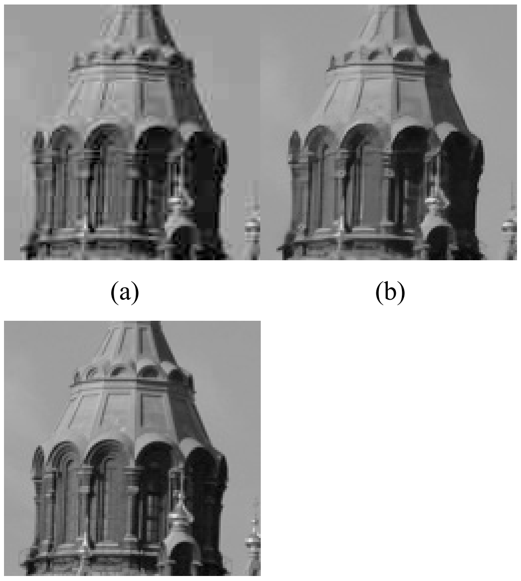


Fig. 2. (a) JPEG compressed image. (b) Generated image. (c) Original uncompressed image.

B. *Visual Quality of Generated Images*

We first conducted experiments to evaluate the visual quality of images generated by our proposed network. Peak-signal-to-noise-ration (PSNR) and structural similarity (SSIM) [20] are two popular metrics for evaluating an image's visual quality. So, we use these two metrics to evaluate the visual quality of generated images with original uncompressed ones as references. For each resolution case, we cropped the central part from 4,000 images in the testing set and computed their average PSNR and SSIM values with their corresponding uncompressed ones as references. At the same time, we compared our method with two popular JPEG compression anti-forensic techniques [5][6].

From Fig. 2, we can see that, the generated image has clearer textures and sharper edges compared with the compressed one. The blocking artifacts and ringing effects from the compressed one are greatly reduced, which makes the generated image visually closer to the uncompressed one.

Table I shows that the generated images from our proposed network have the highest PSNR and SSIM values in most cases. It is evident that our method obtains the best performance of PSNR and SSIM under any QF value of JPEG compression. Specifically, the resolution of 64×64 is so small that a few images stay the same after compression, thus a few compressed images' PSNR values become infinite. However, it is illustrated that our anti-forensic method does not sacrifice but even enhance visual quality of images, whereas the methods proposed in [5][6] will lower the image visual quality actually.

TABLE I. THE AVERAGE PSNR AND SSIM VALUES

Size	Method	QF=25		QF=50		QF=75	
		PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM
64×64	$\mathcal{J}$	***INF***	0.884	***INF***	0.928	***INF***	0.956
	$\mathcal{F}^1$	30.537	0.800	31.548	0.861	32.489	0.910
	$\mathcal{F}^2$	30.424	0.814	31.674	0.852	32.356	0.870
	$\mathcal{F}$	33.559	**0.897**	36.000	**0.938**	38.615	**0.963**
128×128	$\mathcal{J}$	32.444	0.886	34.912	0.929	37.547	0.957
	$\mathcal{F}^1$	29.977	0.764	32.226	0.835	34.646	0.892
	$\mathcal{F}^2$	30.242	0.810	31.453	0.850	32.118	0.870
	$\mathcal{F}$	**33.508**	**0.900**	**35.868**	**0.939**	**38.548**	**0.964**
256×256	$\mathcal{J}$	32.610	0.890	35.074	0.932	37.717	0.959
	$\mathcal{F}^1$	29.606	0.732	31.788	0.807	34.258	0.875
	$\mathcal{F}^2$	30.273	0.805	31.468	0.850	32.138	0.872
	$\mathcal{F}$	**33.605**	**0.902**	**35.919**	**0.940**	**38.633**	**0.964**
512×512	$\mathcal{J}$	33.259	0.898	35.730	0.937	38.359	0.962
	$\mathcal{F}^1$	29.661	0.702	31.825	0.783	34.366	0.860
	$\mathcal{F}^2$	30.772	0.809	32.007	0.854	32.714	0.879
	$\mathcal{F}$	**34.090**	**0.908**	**36.280**	**0.943**	**38.975**	**0.965**

$\mathcal{J}$ means JPEG compression, $\mathcal{F}^1$, $\mathcal{F}^2$ and $\mathcal{F}$ represents the method proposed in [5][6] and this paper respectively. *INF* means the infinite number that computers cannot express.

C. *Performance Evaluation for Anti-forensics*

The purpose of a JPEG compression anti-forensic technique is to deceive a JPEG compression detector. CNN-based methods [3][4] can achieve state-of-the-art performance in image forensics. Thus, we adopt the CNN architecture proposed in [4] to train several JPEG compression detectors using all 16,000 images in the training set created in Section

IV.A and it's compressed ones. For each resolution case, we trained a network and then we computed the detection rate on all 4,000 images in the testing set created in IV.A. We used all testing images to create it's compressed ones and three anti-forensically modified ones, and the detection rate accounts for how many JPEG compressed images or anti-forensically modified images can be detected as JPEG compressed by the detector. Table II shows the detection rate.

Seen from this table, the images modified by our method can effectively decrease the detection rate of the JPEG compression detector. The anti-forensic performance of our method is comparable with the method proposed in [6], and is much better than the method proposed in [5]. Nevertheless, our method has the merit that it can achieve the anti-forensic performance without sacrificing image visual quality, while the methods proposed in [5][6] will decrease the visual quality of images.

TABLE II. THE DETECTION RATE OF THE CNN-BASED JPEG COMPRESSION DETECTOR

Size	Method	QF=25	QF=50	QF=75
		ACC (%)	ACC (%)	ACC (%)
64 × 64	$\mathcal{J}$	100.00	100.00	99.95
	$\mathcal{F}^1$	100.00	99.45	89.43
	$\mathcal{F}^2$	**0.05**	**0**	**0**
	$\mathcal{F}$	14.15	6.55	2.08
128 × 128	$\mathcal{J}$	99.98	99.95	98.15
	$\mathcal{F}^1$	94.73	58.00	20.75
	$\mathcal{F}^2$	**0**	**0**	**0**
	$\mathcal{F}$	12.50	9.05	3.25
256 × 256	$\mathcal{J}$	99.98	99.93	98.95
	$\mathcal{F}^1$	91.76	39.65	9.25
	$\mathcal{F}^2$	**0.18**	**0.13**	**0.025**
	$\mathcal{F}$	16.50	12.73	7.23
512 × 512	$\mathcal{J}$	99.98	99.90	97.33
	$\mathcal{F}^1$	77.08	19.80	4.65
	$\mathcal{F}^2$	**0**	**0**	**0**
	$\mathcal{F}$	4.86	2.95	0.43

$\mathcal{J}$ means JPEG compression, $\mathcal{F}^1$, $\mathcal{F}^2$ and $\mathcal{F}$ represents the method proposed in [5][6] and this paper respectively.

CONCLUSION

In this paper, we have proposed a JPEG compression anti-forensic method based on a generative adversarial network. Unlike traditional JPEG compression anti-forensic methods, we achieve the goal of better anti-forensic performance in a data-driven way. Experimental results show that our method can not only deceive a JPEG compression detector, but also enhance image visual quality. In the future, we will engage in improving the anti-forensic performance of our presented method.

REFERENCES

[1] Z. Fan and R. de Queiroz, "Identification of bitmap compression history: JPEG detection and quantizer estimation," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 230–235, Feb. 2003.

[2] H. Farid, "Digital image ballistics from JPEG quantization," Tech. Rep. TR2006-583, Dept. of Computer Science, Dartmouth College, 2006.

[3] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proc. of the 4th ACM Workshop on Information Hiding and Multimedia Security*, ACM, 2016, pp. 5-10.

[4] B. Bayar and M. C. Stamm, "A Generic Approach Towards Image Manipulation Parameter Estimation Using Convolutional Neural Networks," in *Proc. of the 5th ACM Workshop on Information Hiding and Multimedia Security*, ACM, 2017, pp. 147-157.

[5] M. Stamm, S. Tjoa, W. S. Lin, and K. J. Ray Liu, "Antiforensics of JPEG compression," in *Proc. of the IEEE Int. Conf. Acoust., Speech, and Signal Process.*, 2010, pp. 1694–1697.

[6] M. Stamm and K. Liu, "Anti-forensics of digital image compression", *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1050-1065, 2011.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Proc. of the Advances in Neural Information Processing System (NIPS)*, 2014, pp. 2672-2680.

[8] C. Ledig, L. Theis, F. Husz´ar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv*:1609.04802, 2016.

[9] P. Isola, J. Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[10] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to train a GAN? tips and tricks to make GANs work." accessed 25 July 2017. [Online]. Available: https://github.com/soumith/ganhack

[11] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in *Proc. of the Media Watermarking, Security, and Forensics*, International Society for Optics and Photonics, 2015, vol. 9409, pp. 94090J.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.

[14] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for realtime style transfer and super-resolution," in *Proc. of the European Conference on Computer Vision*, Springer, 2016, pp. 694–711.

[15] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and Li. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of the Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2009, pp. 248-255.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: a system for large-scale machine learning," In *OSDI*, 2016, vol. 16, pp. 265-283.

[17] P. Bas, T. Filter, T. Pevny, "Break our steganographic system: The ins and outs of organizing BOSS," in *Proceedings of the 13th International Conference on Information Hiding (IH)*, 2011, pp. 59-70.

[18] The 2nd BOWS Contest (Break Our Watermarking System) was orgainise within the activity of the Water Marking Virtual Laboratory (Wavila) of the European Network of Excellence ECRYPT (http://www.ecrypt.ed.org) between the 17th of July 2007 and 17th of April, 2009. Available at:http://bows2.ec-lille.fr/.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM", in *Proc. of the International Conference on Pattern Recognition (ICPR)*, IEEE, 2010, pp. 2366-2369.