 CMC, vol.56, no.2, pp.313-324, 2018

Embedding Image Through Generated Intermediate Medium Using Deep Convolutional Generative Adversarial Network

Chuanlong Li[1, 2, *], Yumeng Jiang[3] and Marta Cheslyar[1]

Abstract: Deep neural network has proven to be very effective in computer vision fields. Deep convolutional network can learn the most suitable features of certain images without specific measure functions and outperform lots of traditional image processing methods. Generative adversarial network (GAN) is becoming one of the highlights among these deep neural networks. GAN is capable of generating realistic images which are imperceptible to the human vision system so that the generated images can be directly used as intermediate medium for many tasks. One promising application of using GAN generated images would be image concealing which requires the embedded image looks like not being tampered to human vision system and also undetectable to most analyzers. Texture synthesizing has drawn lots of attention in computer vision field and is used for image concealing in steganography and watermark. The traditional methods which use synthesized textures for information hiding mainly select features and mathematic functions by human metrics and usually have a low embedding rate. This paper takes advantage of the generative network and proposes an approach for synthesizing complex texture-like image of arbitrary size using a modified deep convolutional generative adversarial network (DCGAN), and then demonstrates the feasibility of embedding another image inside the generated texture while the difference between the two images is nearly invisible to the human eyes.

Keywords: GAN, CNN, texture synthesis, steganography, watermark, image concealing, information hiding.

1 Introduction

Deep learning is great for computer vision related tasks because its high complexity and self-learning scheme. Computer itself can detect and find their own objective features by consistent computing while the image features which used to solve a certain task are hard to define by humans sometimes. Concealing image inside another image is an appealing research field in computer vision and information hiding which can be applied on

[1] School of Computer and Software, Nanjing University of Information Science and Technology, Ning Liu Road, No. 219, Nanjing, 210044, China.

[2] Jiangsu Engineering Centre of Network Monitoring, Ning Liu Road, No. 219, Nanjing, 210044, China.

[3] School of Informatics, University of Edinburgh, Old College, South Bridge, Edinburgh, EH8 9YL, UK.

[*] Corresponding Author: Chuanlong Li. Email: nuist_darko@outlook.com.

artworks which need strict protection of the original art piece. However, hiding image in another image is difficult because the hidden capacity is too much to be easily discovered directly by human eyes. Texture synthesizing has drawn lots of attention in computer vision related field and information hiding such as steganography and watermark [Jetchev, Bergmann and Vollgraf (2016); Yuan, Xia and Sun (2017)]. Both steganography and watermark are the kind of technology which embed secret messages into another carrier medium [Wang, Lian and Shi (2017); Xia, Wang, Zhang et al. (2016); Zhou, Wu, Yang et al. (2017)]. The carrier used to hide message is typically an image which is best known to be a complex texture. The stego image is known as the one combines the carrier image and secret image but looks identical to the original image so as not to be easily suspected by the middle attacker. Embedding a secret message into another cover image can easily change the appearance of the original cover image, not to mention embedding a full-size image into another. Thus, when embedding secret into another cover, we need to find the suitable place of the cover image to embed information in, typically means the complex details of an image [Cheng and Wang (2006); Xiong, Xu and Shi (2018)]. And the reason why texture image is getting more attention is because their consistent appearance is less detectable to human vision system and steganographic detection systems [Zhou, Chen, Zhang et al. (2018)].

There are researchers that using computer synthesized images to hide secret message [Efros and Freeman (2001); Gatys, Ecker and Bethge (2015); Jarušek, Volna and Kotyrba (2015); Wu and Wang (2015)]. They usually hide the secret in areas selected by human metrics and suffer from low hidden capacity [Li, Wei, Ferreira et al. (2018); Qian, Dong, Wang et al. (2015); Wang, Lian and Shi (2017)]. Deep convolutional neural network is more suitable for computers to extract image features themselves [Dong, Loy, He et al. (2016)] and generative neural network [Goodfellow, Pouget-Abadie, Mirza et al. (2014); Ledig, Theis, Huszár et al. (2017); Radford, Metz and Chintala (2015)] can be used to generate image which is almost impossible for humans to recognize whether it is a real image. And by defining a small moving patch inside the image, the generative network can generate similar texture-like image as the original one. The idea used in this paper comes from two simple ideas. First is that the cover image used to hide another image is a complex texture-like image completely generated through a network model which is hard to retrieve by a middle attacker. Second is because the network is completely data-driven, the network itself can learn the best possible ways to embed an image inside another complex texture image.

The remainder of this paper is organized as follows. Section II discusses some related works. The proposed architecture and demonstrations are described in section III. Some experiments and results are shown in Section IV. Section V concludes this paper and puts forward some future work.

2 Related works

Baluja [Baluja (2017)] discovers a method of using deep neural networks to hide image inside another image. In their study, they conceal the secret image inside another image called cover image and then obtain an image named container image. The image transition in the process is mainly through three simple deep neural networks. The first

network is called Prep Network which is used to preprocess the hidden image. The second network is called Hiding Network which is also the main network among the entire architecture. This network takes the output of the Hiding Network and a new image which serves as cover image as input, and creates another image contains the secret image which looks alike to the input cover image. The third network called Reveal Network which is used for revealing the secret image from the container image. The author also points out that the whole network system is more like an auto encoder which is most commonly used in image compression. The three parts are trained as a single network with two different losses. The whole network works by learning how to compress the secret image information into the least noticeable parts of the input cover image. The dataset they use is from tiny-imagenet and only this one dataset for both cover images and secret images.

The method proposed by Baluja does not include a complete steganographic system and may also not be resistant to a steganalysis. Hayes et al. [Hayes and Danezis (2017)] presents a new method using generative neural network to generate stego images. They train the discriminator as a corresponding steganalysis in order to decide whether a secret message is contained within an image. By implementing consistent training procedure, the generative network keeps generating stego images so as to fool the discriminator into un-altered images while the discriminator keeps distinguishing whether the image is an embedded secret image or just an ordinary image. Both the generative network and the discriminator network can keep improving their own capability until they reach a certain balance in the competition. At last the generative network would generate real enough images that the discriminator network cannot decide whether it is real or embedded with secret.

Texture synthesis in computer vision is quite popular these days. But using synthesized texture in steganography is quite a few. Wu et al. [Wu and Wang (2015)] propose a patch-based method which can generate steganography using reversible texture synthesis. Their approach can synthesize arbitrary size of texture images and the hidden capacity can be increased proportional to the size of the texture image. The original source texture can also be recovered due to the reversible capacity. Also, the stego texture image is comprised of a source texture instead of modifying the original image contents can provide better visual result than pixel-based algorithms [Efros and Leung (1999); Liang, Liu, Xu et al. (2001)]. However, the existing texture related steganography method usually has low embedding capacity and the features they use to synthesize the texture image are selected in a manually designed manner.

3 Proposed method

The whole network architecture has two specific designs which correspond to two different application scenarios. The first architecture is designed by separating the texture image generation and the secret image embedding process. The texture-like image is generated by a deep convolutional generative neural network and then served as the input image of the concealing network for image hiding. The architecture of the concealing network is adopted from the one proposed by Google research team but with different network designs. The second method integrates the concealing network with the deep convolutional generative network. In this way, the whole network can generate a texture

image while embedding another image in process. Moreover, the first architecture is easier to train and can be used afterwards while the training process for the second architecture is difficult because the two networks have different data input and thus cannot be used in a more common scenario. The trained model of the first architecture can be used multiple times without re-training while the second is more suitable for one-time end-to-end scenario.

The general framework is shown in Fig. 1. The generative network first generates a texture like image served as the input of the concealing network. The generated image and the GAN model should be inaccessible to a third party in practice. Thus, the middle attacker cannot obtain the image used as cover and therefore increase the difficulty for them to distinguish whether an image contains information. Also, because the cover image is purely generated, it is hard for an attacker to restore the embedded image without the model.

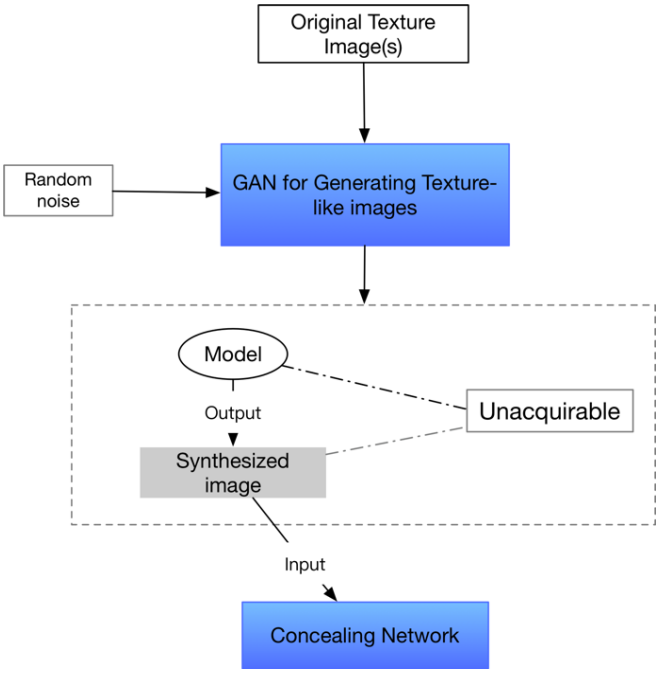

Figure 1: General layer architecture for the concealing network

3.1 Texture generation

Generative adversarial network works by having the generator generates realistic image which is then used to deceive the discriminator. The generator receives a random noise at first and then tries to capture the distribution of the sample images as the discriminator focuses on differentiating the generated image from the real image. The traditional GANs are often unstable to train and can result in unexpected generative outputs. Deep convolutional generative adversarial network improves the performance of traditional GAN by combining Convolutional neural network with traditional GAN concept. The loss function of the proposed GAN network is adopted from the SGAN with minor

modification as shown in formula (1).

$$\min_G \max_D V(D,G) = \frac{1}{lm}\sum_{\alpha=1}^{l}\sum_{\beta=1}^{m}\mathbb{E}_{Z\sim pz(Z)}\left[\log\left(1-D_{\alpha\beta}(G(Z))\right)\right] + \frac{1}{lm}\sum_{\alpha=1}^{l}\sum_{\beta=1}^{m}\mathbb{E}_{X'\sim pdata(X)}\left[\log D_{\alpha\beta}(X')\right] \quad (1)$$

Here, l and m are the spatial dimensions, $1<\alpha<l$, $1<\beta<m$, $X^{'}$ is a random selected rectangular patch of original texture. The discriminator calculates the probabilities of $l\times m$ size field which can be used to determine whether an input is real or generated. P_{data} is the true data distribution and $X^{'}$ has the same size as the generated texture images.

Also, batch normalization has proven to be very efficient for the training process, it can decrease the side effect because of the bad initialization and help the stochastic gradient descents more smoothly and effectively. Batch normalization is used on all layers in generator network except the output layer and so is the discriminator network but also exclude the input layer.

The GAN architecture is shown in Fig. 2. The input to the generative network is a N dimensional noise and then maps to a small patch selected from the original texture image. Then the input tensor flows into several transposed convolution layers with upscaling. All the layers above are applied batch normalization and activated using *ReLU* function. The last layer of the generative network is activated using *Tanh* function without batch normalization. The discriminator used in this GAN architecture is similar as the common GANs. It down samples the first layer and then convolves the next fewer layers with batch normalization and *LReLU* activation function except the output layer. The output layer of the discriminator is activated using sigmoid function and applies no batch normalization. Fully-connected layers are all abandoned in the network architecture for more flexibility and less parameters, but no effect on the outcome since we mainly use the convolutional layers for feature extraction.

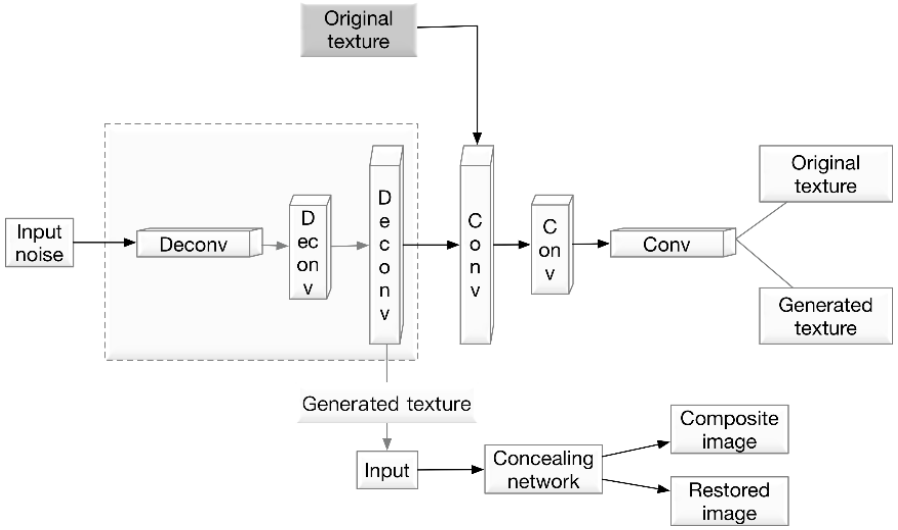

Figure 2: Texture generation network

3.2 Concealing process

The network used for image concealing consists of two parts. The first part is used to conceal the image inside another texture image, the second is used to restore the concealed image as closely as possible. The general network structure is shown in Fig. 3.

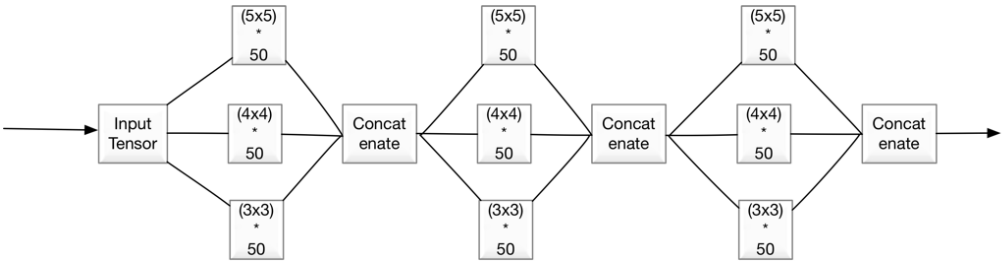

Figure 3: General layer architecture for the concealing network

The feature of texture images is relative simple in this application scenario, for only a small fraction of the details are all we need. Thus, in order to achieve better results, the concealing network only contains convolutional layers and no other layers like pooling or batch normalization layer. The loss function used for the concealing process includes two: SSIM and MSE. SSIM is short for the Structual Similarity index and is great for measuring the similarity between two images. Using SSIM as a loss function to train the concealing network would improve great performance theoriatically. However, it is hard to train the model in practice because of the initial difference between two images are too much. Thus, mean squared error is preffered in this architecture to achieve the optimal performance while reserving good results. The formula of MSE and SSIM are shown below as formula (2) and formula (3) respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2}$$

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{3}$$

Here, μ_x and μ_x are the average of x and y respectively, σ_x and σ_y are the variance of x and y respectively, σ_{xy} is the covariance of x and y. c_1 and c_2 are two variables to stabilize the division with weak denominator. L is the dynamic range of the pixel-values, and k_1=0.01 and k_2=0.03 by default.

4 Experiment

Two different datasets are used for the experiment. The first dataset is extracted from the DTD dataset [Cimpoi, Maji, Kokkinos et al. (2014)] and reformatted into different categories for better training results. DTD dataset includes multiple kinds of texture images which is great for our texture generation process. The concealing network uses

4135 reformatted training images from DTD dataset and 4135 images from the COCO2017 training dataset [Lin, Maire, Belongie et al. (2014)] to train a model for image concealing. The original texture for image synthesize is picked from the DTD dataset with random choice. The texture generation network is trained using one Nvidia GTX1080Ti GPU and the concealing network is trained using four Nvidia GTX1080Ti GPUs. The concealing network is trained for 230 epochs by minimizing the mean square error of pixels between the original image and the concealed image despite SSIM is more suitable and the training epoch of the generation network varies depending on the texture images. Several different network architectures are tested with different kernel size and different number of network layers. All the code is written in Keras with TensorFlow backend.

4.1 Generated texture-like image

The first texture generation experiment uses only one original texture image and then synthesize another texture-like image which resembles the style of the original texture. Some generated examples are shown in Fig. 3 and Fig. 4. In Fig. 3, the first row represents the original images, the second row represents the generated textures. Note that the first two images on the second row are both generated using the same original texture, but with different convolutional layers and training epochs. When the original texture is sparse and complex, the filter number and patch dimension should also be increased accordingly. By modifying the spatial dimension of the random noise, the network can generate image of different size. In Fig. 4, the textures at last row are generated using multiple source textures. The network learns the common features and styles of the original source textures and generates the synthesized texture.

Figure 4: Original source textures and synthesized texture-like images

Figure 5: Multiple source textures synthesize one texture-like image

4.2 Image concealing

Texture image shares little feature similarities with the image to be hidden, thus it is difficult to train the concealing network to reach a good result. The network which achieves the best performance in the experiment uses kernel of 3×3, 4×4 and 5×5 size and 50 kernels of each size in each layer. The restored image may lose some of the details or suffer from color distortion when the two images differ too much. For the main purpose of the architecture is to conceal image with the minus notice, we implement the conceal network to try to minimize the pixel difference between the texture image and the embedded image, while sacrifice some clarity of the restored image as shown in Fig. 6 (From left to right: Secret image, embedded texture-like image, restored-secret, residual for texture image, residual for secret image). We also did experiments regarding the second framework, but the results turn out to be not very ideal. The restored image often suffers severe color distortion when the two images differ too much. One reason maybe because the training images are too few for the network to learn enough feature expressions. The generated texture which embeds secret image is also sent to a steganalysis tool for anomaly detection. We refer to the embedded image as stego image under this circumstance and get a detection rate at 0.02%. Despite the steganalysis tool mainly focuses on the lowest bit anomaly detection, we still can argue that the generated texture and the secret merge well with each other across the whole image area. The image pixel is rescaled between 0 and 255 during pixel error calculation for better clarity. The results of the detection and pixel difference are shown in Tab. 1.

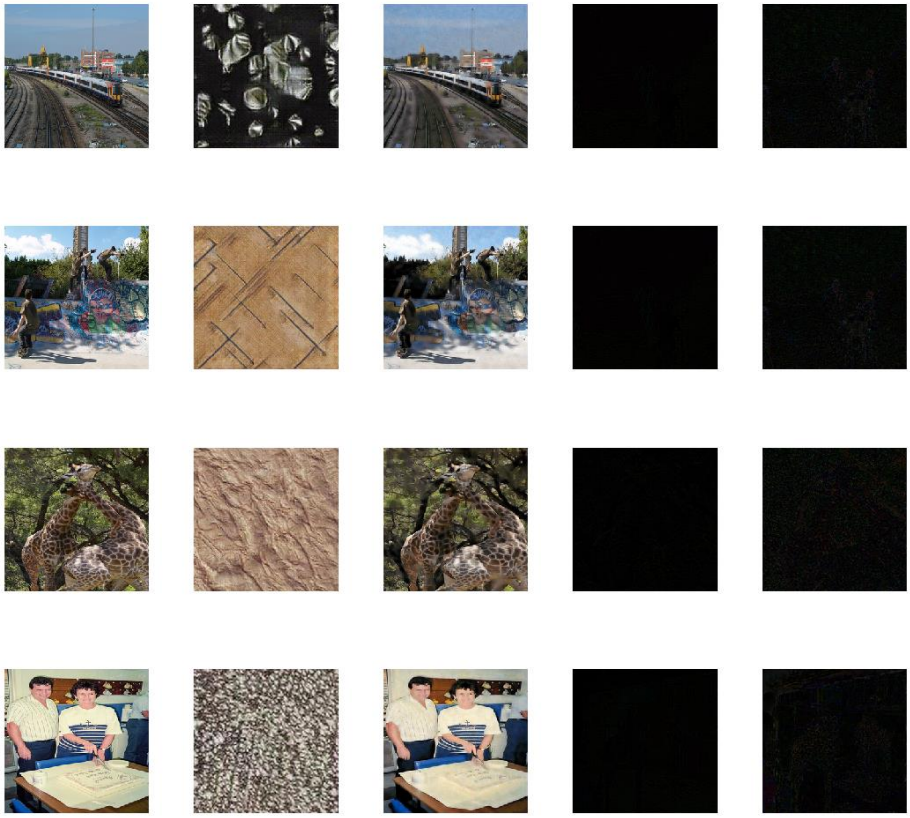

Figure 6: Generated texture image concealing results

Table 1: Embedding result and detection rate

	MSE	Detection Rate
Embedded Texture	3.54	0.02
Restored Secret	11.46	-

5 Conclusion

Currently the generative network is specialized for texture image generation which is simpler than nature image generation. And the texture image generated by the GAN network has to have the same width and height as the secret image to the concealing network. Although this seems less important because the GAN network can generate arbitrary size of image, one can see that allow arbitrary size of secret image embedded in a cover image is much more sophisticated and can reach less detective rate. Also, the current network does not use any sophisticated architecture and it is possible to improve

the performance by altering the network structure. The feature extraction process can be better achieved by using more complex and sophisticated network designs such as inception and residual network. The image embedding process can be integrated into the texture image generation process which can achieve more natural looking images and less detection rate.

Acknowledgement: This work is supported by the National Key R&D Program of China under grant 2018YFB1003205; by the National Natural Science Foundation of China under grant U1536206, U1405254, 61772283, 61602253, 61672294, 61502242; by the Jiangsu Basic Research Programs-Natural Science Foundation under grant numbers BK20150925 and BK20151530; by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund; by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET) fund, China.

References

Baluja, S. (2017): Hiding images in plain sight: Deep steganography. *Advances in Neural Information Processing Systems*, pp. 2066-2076.

Cheng, Y. M.; Wang, C. M. (2006): A high-capacity steganographic approach for 3D polygonal meshes. *The Visual Computer*, vol. 22, no. 9, pp. 845-855.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; Vedaldi, A. (2014): Describing textures in the wild. *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 3606-3613.

Dong, C.; Loy, C. C.; He, K.; Tang, X. (2016): Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295-307.

Efros, A. A.; Freeman, W. T. (2001): Image quilting for texture synthesis and transfer. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 341-346.

Efros, A. A.; Leung, T. K. (1999): Texture synthesis by non-parametric sampling. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1033-1038.

Gatys, L.; Ecker, A. S.; Bethge, M. (2015): Texture synthesis using convolutional neural networks. *Advances in Neural Information Processing Systems*, vol. 70, no. 1, pp. 262-270.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D. et al. (2014): Generative adversarial nets. *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672-2680.

Hayes, J.; Danezis, G. (2017): Generating steganographic images via adversarial training. *Advances in Neural Information Processing Systems*, vol. 30, pp. 1951-1960.

Jarušek, R.; Volna, E.; Kotyrba, M. (2015): Neural network approach to image steganography techniques. *Advances in Intelligent Systems and Computer*, vol. 378, pp. 317-327.

Jetchev, N.; Bergmann, U.; Vollgraf, R. (2016): Texture synthesis with spatial generative adversarial networks. *Computer Vision and Pattern Recognition*.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A. et al. (2017): Photo-realistic single image super-resolution using a generative adversarial network. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 105-114.

Li, B.; Wei, W.; Ferreira, A.; Tan, S. (2018): ReST-Net: Diverse activation modules and parallel subnets-based CNN for spatial image steganalysis. *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 650-654.

Li, F.; Wu, K.; Zhang, X.; Lei, J.; Wen, M. (2018): Multi-source stego detection with low-dimensional textural feature and clustering ensembles. *Symmetry*, vol. 10, no. 5, pp. 1-23.

Liang, L.; Liu, C.; Xu, Y. Q.; Guo, B.; Shum, H. Y. (2001): Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, vol. 20, no. 3, pp. 127-150.

Lin, T. Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P. et al. (2014): Microsoft COCO: Common objects in context. *European Conference on Computer Vision*, vol. 8693, pp. 740-755.

Qian, Y.; Dong, J.; Wang, W.; Tan, T. (2015): Deep learning for steganalysis via convolutional neural networks. *Media Watermarking, Security and Forensics*, vol. 9409, pp. 94090J-94090J-10.

Radford, A.; Metz, L.; Chintala, S. (2015): Unsupervised representation learning with deep convolutional generative adversarial networks. *Computer Science*.

Wang, J.; Lian, S.; Shi, Y. Q. (2017): Hybrid multiplicative multi-watermarking in DWT domain. *Multidimensional Systems and Signal Processing*, vol. 28, no. 2, pp. 617-636.

Wu, K. C.; Wang, C. M. (2015): Steganography using reversible texture synthesis. *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 130-139.

Xia, Z.; Wang, X.; Zhang, L.; Qin, Z.; Sun, X. et al. (2016): A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Transactions on Information Forensics and Security*, vol. 11, pp. 2594-2608.

Xiong, L.; Xu, Z.; Shi, Y. Q. (2018): An integer wavelet transform based scheme for reversible data hiding in encrypted images. *Multidimensional Systems and Signal Processing*, vol. 29, no. 3, pp. 1191-1202.

Yuan, C.; Xia, Z.; Sun, X. (2017): Coverless image steganography based on SIFT and BOF. *Journal of Internet Technology*, vol. 18, no. 2, pp. 435-442.

Zhou, H.; Chen, K.; Zhang, W.; Qian, Z.; Yu, N. (2018): Targeted attack and security enhancement on texture synthesis based steganography. *Journal of Visual Communication and Image Representation*, vol. 54, pp. 100-107.

Zhou, Z.; Wu, Q. J.; Yang, C. N.; Sun, X.; Pan, Z. (2017): Coverless image steganography using histograms of oriented gradients-based hashing algorithm. *Journal of Internet Technology*, vol. 18, no. 5, pp. 1177-1184.