Neural Network Steganography Using Extractor Matching

Yunfei Xie and Zichi Wang

December 3, 2023

Neural Network Steganography Using Extractor Matching

Yunfei Xie[0009-0002-6435-7976], and Zichi Wang[0000-0003-0283-5338]*

Shanghai University, Shanghai, 200444, China
{xieyunfei, wangzichi}@shu.edu.cn

Abstract. Neural networks have been applied in various fields, including steganography (called neural network steganography). The network used for secret data extraction is called the extractor. This paper proposes a neural network steganography scheme using extractor matching. In our scheme, the extractor is a publicly available normal network possessed by the receiver, which is used for conventional intelligent tasks. Sender connects extractor to another neural network (called cover network), and then trains the connected network to guarantee correctly data extraction without decreasing the performance of the original task of cover network. During the process of training, the parameters of extractor remain unchanged. Specifically, these network parameters are obtained using an extraction key. The receiver can correctly extract secret data with the help of correct extraction key, while an incorrect key will fail to extract secret data. The feasibility of our scheme is demonstrated in experiments.

Keywords: Steganography, Neural networks, Extractor matching.

1 Introduction

As neural network continues to evolve, their capabilities have become more powerful and diverse. Neural networks are now extensively utilized in various fields, demonstrating their wide-ranging applications. Neural networks not only exhibit excellent performance in many traditional tasks, e.g., target track [1], image recognition [2], natural language processing [3], but also demonstrate strong performance in steganography.

Many researchers, like the authors in [4], use neural networks to achieve image steganography. With the rapid advancement of neural network, it has become increasingly common to directly implement steganography within neural networks. Wu et al. in [5] proposed a digital neural network watermarking framework, which can both perform the original task of the network and embed data within it. The authors in [6] used a backdooring technique to train a model that deliberately outputs specific data with a trigger. The specific data have no noticeable impact on the primary task for which the model is designed.

In neural network steganography, the sender aims to embed secret data into neural network model with tiny impact on the original model. The receiver can extract secret data using an extraction network, which is designed for data extraction specially. The extraction network is typically transmitted to receiver with the stego network, as shown in Fig. 1. The problem with this framework is the risk of interception by a third party during the transmission of extraction network.

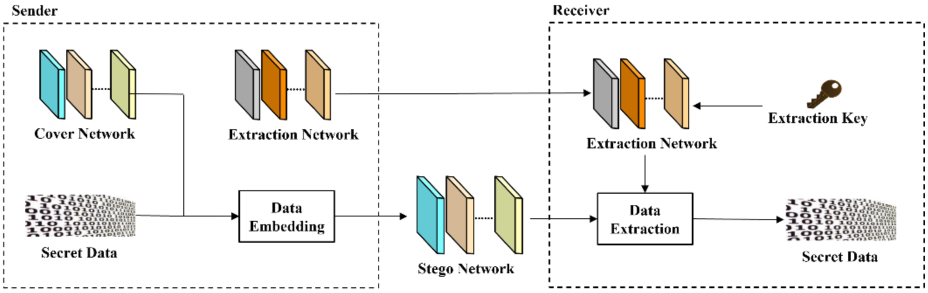


Fig. 1. Neural network steganography.

To mitigate this risk, this paper proposes a novel neural network steganography scheme using extractor matching. Extractor is a publicly available normal neural network possessed by receiver. The sender performs data embedding based on extractor in order to make sure the secret data must use extractor for extraction. Matching training can achieve it. Extraction key is agreed upon in advance by the sender and receiver. This means that extractor and extraction key using for data extraction are available to the receiver before the secret data is transmitted.

By doing so, the sender only needs to transmit the stego network because receiver already has extractor and extraction key. In this way, even if extraction key is accessed by a third-party during transmission, he cannot extract the secret data because secret data has not been transmitted at that time. If both extraction key and stego network are obtained by the third party, he cannot know which one is extractor because there are a huge number of publicly available normal networks on the Internet, and therefore he cannot get the secret data. Similarly, even if the third parties have extractor and stego network, they cannot extract valid secret data without extraction key agreed upon by the sender and receiver in advance. But receiver can correctly extract secret data by extractor with the help of correct extraction key. Wang et al. in [7] proposed a similar functionality, which involves extracting additional data without transmitting corresponding key. In this way, the security of steganography can be enhanced. Neural network steganography using extractor matching is designed precisely to avoid the issue of data leakage caused by extraction network.

For certain scenarios, e.g., a general steganographic framework for neural networks to achieve covert communication proposed in [8], a multi-source data hiding scheme which multiple senders can simultaneously transmit different secret data to a receiver in [9]. Utilizing steganography using extractor matching can omit the step of transmitting extracting network to receiver. The non-specific transmission of extraction network greatly reduces the risk of data leakage, as it is very difficult for third parties to find extractor in the massive public normal network.

In this paper, ResNet [10] and AlexNet [11] are used as an example. Neither network is necessary. Different networks need to design different embedding and extraction schemes. However, the use of a normal neural network is necessary because a publicly available neural network with normal functionality is less detectable by third parties than a special extractor or cover network, which is used for communication. In this paper, AlexNet serves as the extractor owned by receiver and ResNet acts as the cover of secret data, as shown in Fig. 2.

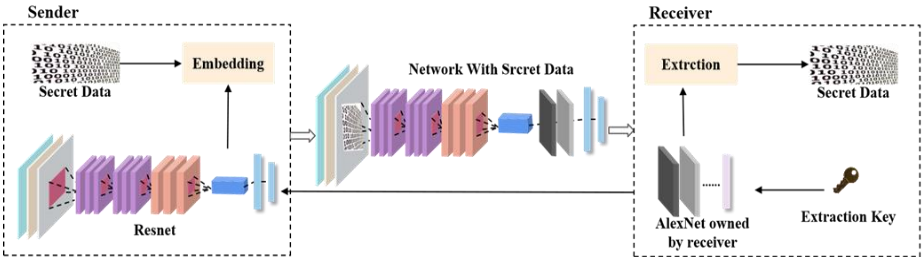


Fig. 2. Steganography scheme using extractor matching

During the data embedding process, the method described in [12] was employed, which involves embedding during the training process. Because AlexNet is a public network, the sender can get it directly without the receiver specifically transmitting it. Adding the weights of AlexNet to the loss function calculation of ResNet links the two networks. The loss function of ResNet is modified to two parts: extraction loss and original network loss. AlexNet is a pre-trained network. The weights of AlexNet are used in training to calculate the extraction loss with the weights of ResNet, and its weights do not change throughout the training process, which ensures that the receiver can use the same network to correctly extract secret data. Extraction key is used to obtain the parameters of AlexNet. The weights of ResNet are constantly changed during training to make the loss small enough. This approach helps to maintain the neural network's original detection accuracy without significant degradation. Receiver can extract the secret data from the stego network by AlexNet. Extraction key is agreed upon in advance by the sender and receiver, and it is only used to help extractor for extraction. The innovation and contributions of this paper can be summarized as follows:

1. This paper proposes a novel neural network steganography method using extractor matching. It involves incorporating extractor into cover network to participate in data embedding and model training, thereby achieving personalized neural network steganography tailored to the receiver.
2. This paper addresses the transmission security issue of transmission. By considering a public normal network possessed by the receiver as extractor, a customized neural network steganography scheme is proposed. This scheme achieves the correctly data extraction without specially transmitting extractor by the sender.

3. The proposed method in this paper does not significantly affect the detection accuracy of cover network by embedding secret data during the training process. Satisfactory results are shown in the experiments section.

2 Method

ResNet and AlexNet are both classic models of Convolutional Neural Networks (CNNs). In this paper, these two networks are used as examples to verify the feasibility of our scheme. The reason for using ResNet is that, in steganography, it is required that the cover network be common and that there be a large number of such networks. Such a network is secure in transmission, and ResNet is one such commonly available network. AlexNet is used because of its simple structure, which allows the receiver to perform as few operations as possible during the data extraction process. The requirement for the extractor is to make it easier for the receiver to use it to extract secret data.

In this paper, the proposed steganographic scheme is implemented on the MNIST dataset. Initially, the training of AlexNet is performed, simulating a scenario where the receiver possesses a normal network which is capable of recognizing handwritten digit images.

2.1 Data embedding

This paper aims to embed data into the weights of ResNet during training process. The embedding function is

$$\boldsymbol{V_M} = f(\boldsymbol{W}, \boldsymbol{K}), \tag{1}$$

where $\boldsymbol{W}$ is the weights of cover network (ResNet in this paper) using for data embedding, and $\boldsymbol{K}$ is the weights of extractor (AlexNet in this paper). Since the sizes of two networks' weights are different, it needs to be standardized during the embedding process so that their sizes are the same with secret data. The secret data is a string of binary sequences that can be thought of as a two-dimensional tensor. In this paper, tensor is standardized by increasing the dimension, retaining the axis while translating the input tensor, and modifying the number of neurons. The output $\boldsymbol{V_M} = [V_m(1), V_m(2), \dots, V_m(t)]^T \in [0,1]^t$, "$t$" represents the length of secret data. It is also the number of neurons in the last dimension after standardization.

Let secret data $\boldsymbol{M} = [m(1), m(2), \dots, m(t)]^T \in (0,1)^t$. The value of $\boldsymbol{V_M}$ should close to $\boldsymbol{M}$ as much as possible. This makes the embedded data easy to extract in subsequent operations. To achieve the function, the total loss of ResNet in (2) needs to be divided into two parts. One is the extraction loss of secret data, and the other part is the original loss of ResNet.

$$Loss_R = \alpha \cdot Loss_M + Loss_r \tag{2}$$

$Loss_r$ is the original loss of ResNet using for ensuring satisfactory detection accuracy. $Loss_M$ is the mean square error of the value of $\boldsymbol{V_M}$ and the real embedded data, which is defined as

$$Loss_M = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{M_i} - \boldsymbol{V_{M_i}})^2 \tag{3}$$

n is the number of samples in the dataset. The extraction error and extraction loss are not calculated in the same way. However, only if the extraction loss is small enough during training can sender guarantee that receiver will extract secret data correctly. Extraction error is the basis for judging whether secret data is extracted correctly. Extraction loss needs to use the weights of AlexNet to participate in the calculation. Although the weights of AlexNet do not change during training, it does affect the training results of ResNet. Modifications under the guidance of loss function connect the two networks.

In (2), α is a very significant parameter which balances extraction error and detection accuracy by controlling extraction loss and original loss of ResNet during training. As the value of α increases, the extraction error diminishes while the detection precision decreases. The discussion about the value of α will be addressed in Section 3.1.

2.2 Decoding Network

In the above subsection, it was stated that the secret data is embedded into the weights of ResNet. The data is embedded in the second fully connected layer of ResNet. This result was obtained through multiple experimental attempts, and the use of this layer has less impact on the original task. Based on the network structure of AlexNet in Fig. 3, this paper selects the second and third Conv (convolutional) layers, as well as the last FC (fully connected) layer for joint training. Again, this selection is the result of experimentation. The experimental result is mentioned in Section 3.2.

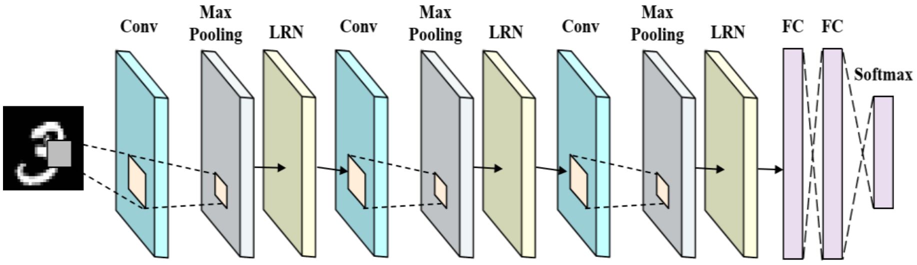


Fig. 3. Architecture of AlexNet for MNIST.

For the receiver, it only needs to use the pre-agreed extraction key to get the weights of extractor, and the perform some simple calculations on the weights of cover network and extractor. Mathematically, receiver only needs to get the value of $\boldsymbol{V_m}$ and round it to get the secret data. The value of $\boldsymbol{V_M}$ can be obtained by computing the Hadamard product between $\boldsymbol{W}_{dr}$ and $\boldsymbol{K}_{dr}$, as shown in Fig.4. $\boldsymbol{W}_{dr}$ and $\boldsymbol{K}_{dr}$ are the parameters $\boldsymbol{W}$ and $\boldsymbol{K}$ after standardization referred to in Section 2.1 and normalization referred to below. $\boldsymbol{W}_{dr}$ and $\boldsymbol{K}_{dr}$ come in different sizes with $\boldsymbol{W}$ and $\boldsymbol{K}$.

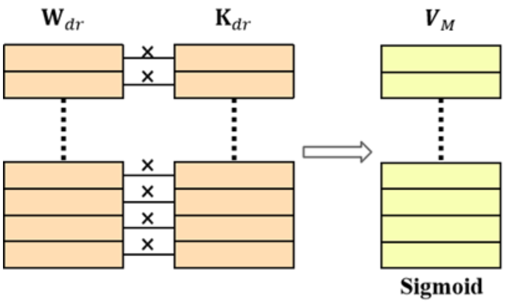


Fig. 4. Decoding network.

In Fig. 4, the value of $\boldsymbol{W}_{dr}$ and $\boldsymbol{K}_{dr}$ is any real number, but the value of $\boldsymbol{V}_{\boldsymbol{M}}$ is a real number between 0 and 1. To guarantee the value of $\boldsymbol{V}_{\boldsymbol{M}}\epsilon(0,1)^t$, it is also necessary to normalize the Hadama product of $\boldsymbol{W}_{dr}$ and $\boldsymbol{K}_{dr}$ using the sigmoid function. Cause the output range of sigmoid is 0 to 1, after processing the Hadama product with sigmoid, the value of $\boldsymbol{V}_{\boldsymbol{M}}$ is maintained between 0 and 1. Hadamard product is defined as

$$\mathbf{A} \odot \mathbf{B} = a_{ij} \times b_{ij} \tag{4}$$

A and **B** are both m×n-order matrices. $\boldsymbol{A} = [a_{ij}]_{m\times n}$ and $\mathbf{B} = [b_{ij}]_{m\times n}$. Since the value of $\boldsymbol{V}_{\boldsymbol{M}}$ is any real number between 0 and 1, and the value of $\boldsymbol{M}$ is 0 or 1, and the value of $\boldsymbol{V}_{\boldsymbol{M}}$ is designed as close to $\boldsymbol{M}$ as possible by continuously adjusting the weights of ResNet during training process, the extraction data can be recovered by rounding the values of $\boldsymbol{V}_{\boldsymbol{M}}$ in (5). Thus, the extraction error can be calculated using (6). In (6), $e = 0$ means that the secret data are extracted correctly.

$$\boldsymbol{M}_r = round(\boldsymbol{W}_{dr} \odot \boldsymbol{K}_{dr}) \tag{5}$$

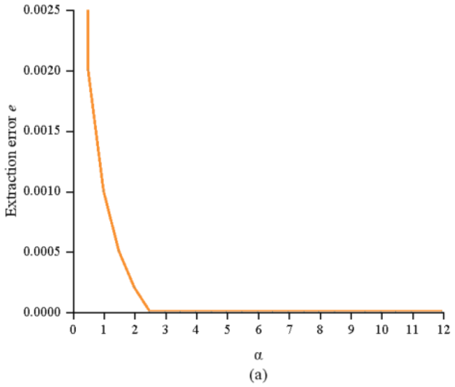
$$e = \frac{1}{nt}\sum_{i=1}^{n}\sum_{j=1}^{t}|\boldsymbol{M}_i(j) - \boldsymbol{M}_{r_i}(j)| \tag{6}$$

3 Experimental Results

To verify the feasibility of the proposed scheme, this section presents experimental results and analysis. All the experiments are implemented by TensorFlow and trained under the environment of Python 3.6 on a Windows 10 system with an NVIDIA GeForce GTX 1660 SUPER GPU with 14 GB of memory. The Adam optimizer [13] is used for optimization.

3.1 Parameter Determination

In Section 2.1, it was mentioned that α can balance the relationship between extraction error and detection accuracy by controlling extraction loss and original loss of ResNet, ensuring both are satisfactory. This section will verify this point through experiments. In the experiment, a batch size of 20 and a capacity of 2000 bits were set. The second and third Conv layers, as well as the last FC layer were used for joint training. The detection accuracy of pre-trained AlexNet is hovered around 0.92. The extraction error and detection accuracy with different α are shown in Fig. 5.



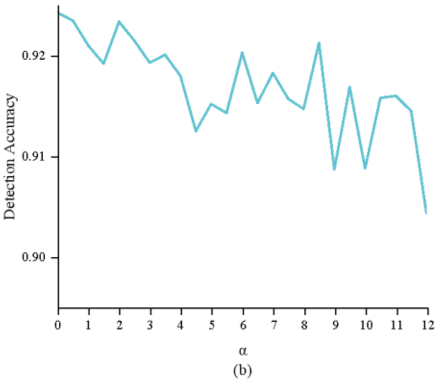


Fig. 5. Extraction error and detection accuracy with different α, (a) extraction error; (b) detection accuracy

In Fig. 5(a), when $\alpha = 0$, the extraction error $e = 0.4803$, which is significantly higher compared to $\alpha \neq 0$. This shows that it is indeed possible to control the magnitude of to alter the network's emphasis of training.

The experimental results indicate that increasing reduces the extraction error while simultaneously decreasing the detection accuracy, consistent with the description in Section 2.1 and confirming the rationality of the design.

According to the Fig. 5, when $\alpha = 2$, the detection accuracy starts to noticeably decline, whereas at $\alpha = 2$, the extraction error is relatively small. This suggests that the value of α should be taken as small or equal to 2 as possible. Considering these factors, it can be concluded that $\alpha = 2$ satisfies the requirements for both detection accuracy and extraction error.

3.2 Determination of Extractor Layers

Layers of AlexNet using for joint training affects the efficiency of embedding and extracting. To obtain a good solution, the following experiments were conducted to decide which layer to use. The experiments were conducted with a batch size of 20, a capacity of 100 bits, and the parameter α in (2) being 2. The extraction error, detection accuracy and training time with different layers or keys are shown in Table I.

Table 1. Extraction error, detection accuracy and training time with different layers or keys

Layers or keys of Extractor	Extraction Error *e*	Detection Accuracy	Training Time(s)
None	/	0.9371	7.898
Conv1, Conv2, Conv3 FC1, FC2, FC3	0	0.9395	86.807
Conv1, Conv2, Conv3 FC2, FC3	0	0.9259	26.516
Conv2, Conv3 FC2, FC3	0	0.9217	26.572
Conv2, Conv3 FC3	0	0.9349	11.770
Conv2, Conv3	0	0.8940	11.696
Random Key I	0.171	0.9242	8.020
Random Key II	0.183	0.9234	8.130

In Table I, “None” represents the original ResNet without being connected AlexNet for data embedding. “Random Key” refers to using a randomly generated extraction key to get the weights of AlexNet. This is to verify whether an erroneous key can assist the extractor in data extraction.

Compared with the original ResNet, it can be observed that embedding secret data using AlexNet has an impact on the training time of ResNet. The magnitude of this impact depends on which and how many layers of AlexNet are used. The more layers are used, the longer training time will be. The training time with the FC layer evidently increasing when compared to the results of the experiments without FC layer. This is because when FC layers are connected to ResNet, they establish more neural connections than Conv layers, resulting in greater computational requirements. The experimental results also showed that if only Conv layers are used, the network would sacrifice detection accuracy to ensure the correct extraction of secret data. This elucidates the rationale behind selecting the second and third Conv layers, along with the final FC layer in Section 2.2. By doing so, the training time is effectively reduced without compromising the extraction error of secret data and preserving the network's detection accuracy. The last two sets of experiments also demonstrate that only with the use of correct extraction key can the secret data be successfully recovered, thereby confirming the security.

3.3 Embedding Capacity and Security

Embedding capacity is an important evaluation metric in steganography. This subsection will experimentally explore the embedding capacity of the proposed steganography scheme.

The length of secret data is set as {500, 1000, ..., 6000}. The maximum capacity of the network is determined by extraction error and detection accuracy. When both values of extraction error and detection accuracy fall within the acceptable range, the maximum length of secret data is maximum embedding capacity. When the extraction error $e = 0$, it indicates that the secret data can be embedded and extracted correctly. The batch size is set to 20, and the parameter α in (2) being 2. The second and third Conv layers, as well as the last FC layer were used for training. The extraction error and detection accuracy with different capacity are shown in Fig. 6.

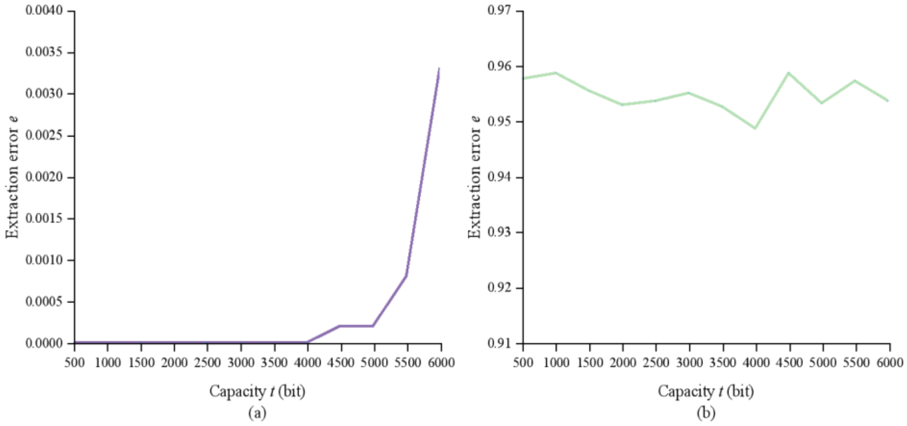


Fig. 6. Extraction error and detection accuracy with different capacity, (a) extraction error; (b) detection accuracy

In Fig. 6(a), when capacity $t > 4000$, the extraction error $e \neq 0$. This can be considered as the embedding capacity of proposed scheme being 4000 bits, which is satisfactory for steganography.

In terms of the security of the proposed method, this paper proves that it is secure enough by comparing the histograms of the parameter distribution of ResNet before and after data embedding, as shown in Fig. 7. The histogram shows that the similarity of the parameter distribution before and after data embedding is very high, and it is difficult to distinguish between stego and normal networks, so the security is also guaranteed.

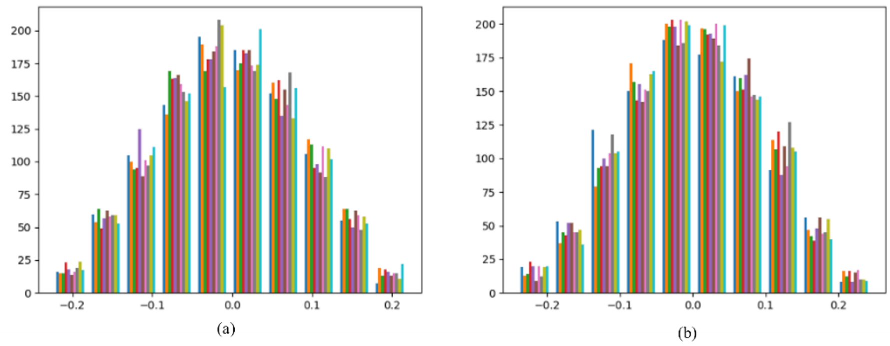


Fig. 7. Histogram of parameter distribution, (a) before data embedding, (b) after data embedding

The experiments in this section demonstrated the feasibility of the scheme and its successful embedding performance. However, further exploration is needed in the future to achieve better embedding performance. For example, how to achieve a larger embedding capacity with no impact on the detection accuracy of cover network.

4 Conclusion

This paper proposes a neural network steganography scheme using extractor matching. The sender and receiver agree in advance on an extraction key, and then sender connects a public normal network called extractor to the other network called cover network for data embedding. Extraction key is used to obtain the parameters of extractor. The proposed scheme does not modify the parameters of extractor during data embedding. Extractor is a publicly available normal network possessed by the receiver. Therefore, there is no need for specially transmitting the extractor by sender, and receiver can have the extractor and extraction key before transmission of stego network, making the steganography more secure. Receiver can obtain the secret data via extractor with the help of correct extraction key, while other keys fail to extract secret data. In this paper, AlexNet serves as extractor owned by receiver and ResNet acts as cover network. The detection accuracy of the original cover network is not affected while ensuring the correctly data extraction. This is achieved by embedding data during the training process instead of modifying parameters after training.

For further study, it is hoped to develop more neural network for steganography using extractor matching. Expand the choices for data embedding beyond just the weight of cover network, making the data embedding more covert. Additionally, a general framework of steganography using extractor matching can be considered.

Acknowledgment

This work was supported in part by the Natural Science Foundation of China under Grants 62376148 and 62002214, and supported in part by the Chenguang Program of Shanghai Education Development Foundation and Shanghai Municipal Education Commission under Grant 22CGA46.

References

1. Omid Elhaki, Khoshnam Shojaei, “Neural network-based target tracking control of underactuated autonomous underwater vehicles with a prescribed performance,” in Ocean Engineering, Volume 167,2018, Pages 239-256.
2. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2016, pp. 770–778.
3. A. Gelbukh, “Natural language processing,” Fifth International Conference on Hybrid Intelligent Systems (HIS'05), Rio de Janeiro, Brazil, 2005, pp. 1 pp.-.
4. A. G. Devi, A. Thota, G. Nithya, S. Majji, A. Gopatoti and L. Dhavamani, “Advancement of Digital Image Steganography using Deep Convolutional Neural Networks,” 2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC), Bengaluru, India, 2022, pp. 250-254.
5. H. Wu, G. Liu, Y. Yao, and X. Zhang, “Watermarking neural networks with watermarked images,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 7, pp. 2591–2601, 2021.
6. Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning your weakness into a strength: Watermarking deep neural networks by backdooring,” in Proc. 27th {USENIX} Security Symp. ({USENIX} Security 18), 2018, pp. 1615–1631.
7. Z. Wang, G. Feng, H. Wu and X. Zhang, “Data Hiding in Neural Networks for Multiple Receivers [Research Frontier],” in IEEE Computational Intelligence Magazine, vol. 16, no. 4, pp. 70-84, Nov. 2021.
8. Z. Yang, Z. Wang, and X. Zhang, “A general steganographic framework for neural network models,” Information Sciences, vol. 643, pp. 119250, 2023.
9. Z. Yang, Z. Wang, X. Zhang and Z. Tang, “Multi-source Data Hiding in Neural Networks,” 2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP), Shanghai, China, 2022, pp. 1-6.
10. K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,”2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778.
11. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. “ImageNet classification with deep convolutional neural networks.” Commun. ACM 60, 6 (June 2017), 84–90.
12. Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, “Embedding watermarks into deep neural networks,” in Proc. 2017 ACM Int. Conf. Multimedia Retrieval, pp. 269–277.
13. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.