Audio Steganalysis With Convolutional Neural Network

Bolin Chen
Sun Yat-sen University
School of Data and Computer Science
Guangzhou, China
chenbl8@mail2.sysu.edu.cn

Weiqi Luo*
Sun Yat-sen University
School of Data and Computer Science
Guangzhou, China
luoweiqi@mail.sysu.edu.cn

Haodong Li
Sun Yat-sen University
School of Data and Computer Science
Guangzhou, China
lihaod@mail2.sysu.edu.cn

ABSTRACT

In recent years, deep learning has achieved breakthrough results in various areas, such as computer vision, audio recognition, and natural language processing. However, just several related works have been investigated for digital multimedia forensics and steganalysis. In this paper, we design a novel CNN (convolutional neural networks) to detect audio steganography in the time domain. Unlike most existing CNN based methods which try to capture media contents, we carefully design the network layers to suppress audio content and adaptively capture the minor modifications introduced by ±1 LSB based steganography. Besides, we use a mix of convolutional layer and max pooling to perform subsampling to achieve good abstraction and prevent over-fitting. In our experiments, we compared our network with six similar network architectures and two traditional methods using handcrafted features. Extensive experimental results evaluated on 40,000 speech audio clips have shown the effectiveness of the proposed convolutional network.

CCS CONCEPTS

•**Security and privacy** →*Authentication;* •**Computing methodologies** →*Learning latent representations;*

KEYWORDS

Audio Steganalysis, Deep Learning, Convolutional Neural Network

ACM Reference format:
Bolin Chen, Weiqi Luo, and Haodong Li. 2017. Audio Steganalysis With Convolutional Neural Network. In *Proceedings of IH&MMSec '17, June 20–22, 2017, Philadelphia, PA, USA, ,* 6 pages.
DOI: http://dx.doi.org/10.1145/3082031.3083234

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IH&MMSec '17, June 20–22, 2017, Philadelphia, PA, USA
© 2017 ACM. 978-1-4503-5061-7/17/06...$15.00
DOI: http://dx.doi.org/10.1145/3082031.3083234

1 INTRODUCTION

Steganography is the art of hiding secret messages into digital covers such as images, audio and video. On the contrary, steganalysis aims to expose the hidden secret messages with steganography. In the past decade, many steganalytic methods have been reported. However, most existing steganalytic methods are mainly dependent on the handcrafted features, which means that these methods have to carefully analyze the hiding property of the targeted steganography and design the special features for steganalysis. In this paper, we focus on audio steganalysis with deep learning, which is a popular machine learning technique based on learning representations of data.

Up to now, many statistical features have been investigated for audio steganalysis. For instance, in [8], the authors tried to build a linear basis to captures certain statistical properties of audio signal. In [10], the authors introduced the Mel-frequency based feature for audio steganalysis. In [13], the authors analyzed the statistics of the high-frequency spectrum and the Mel-cepstrum coefficients of the second-order derivative, and then in [14], they employed the Mel-cepstrum coefficients and Markov transition features from the second-order derivative of the audio signal. Besides, there are some effective features to detect audio steganography in the frequency domain, such as AMR based steganalysis [18] and MP3 based steganalysis [17]. The above methods are based on handcrafted features, and the performances with existing features are still far from satisfactory for the audio steganalysis in the time domain.

Unlike traditional methods, deep learning can effectively replace handcrafted features via feature learning and hierarchical feature extraction. Recently, various deep learning architectures such as Deep Belief Network (DBN)[4], Stacked Auto Encoder (SAE) [5], Convolutional Neural Network (CNN) [11] and Recurrent Neural Network (RNN) [6] have been proposed, and achieve state-of-the-art results in many areas, such as computer vision, audio recognition, and natural language processing. However, just several deep learning based methods have been proposed for digital forensics and steganalysis such as [1, 2, 15, 16, 19, 25].

In this paper, we proposed a new Convolutional Neural Network (CNN) to detect ±1 LSB audio steganography in the time domain. Although this audio steganography in the time domain is a little bit old, to our best knowledge, the detection accuracy with existing steganalytic methods is still far from satisfactory until now. Since the modification introduced by the ±1 LSB steganography is minor, the original

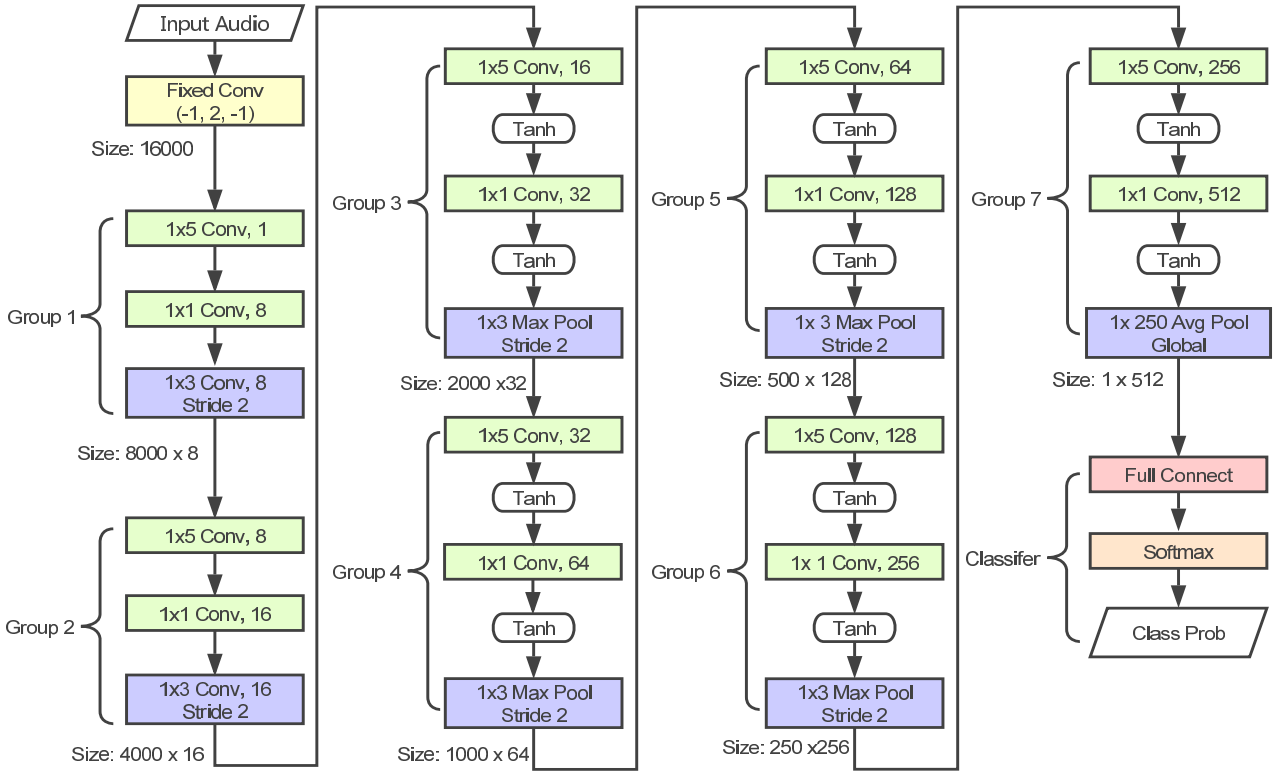


Figure 1: The proposed CNN Architecture. The parameters inside the boxes represent kernel size, layer type, and the number of channels, respectively. For example, "1×1 conv, 8" means a convolutional layer with 1x1 kernel size and 8 channels. The size of output data is showed following the subsampling layers (i.e., the purple boxes).

contents would be well preserved after data hiding. Thus, those typical network architectures that try to capture media contents would not be suitable for this steganalysis problem. In the proposed network, we first obtain the residual of an audio clip with a fixed convolutional layer. Then, seven groups of layers are applied for transforming the input data into a 512-D feature. Finally, a fully connected layer and a softmax layer are served as a classifier to output the class probabilities. In order to reduce the danger of overfitting and enhance the robustness of the proposed model, we have introduced several modifications into the groups of layers. For example, using 1×1 convolutional layers to reduce the number of parameters, performing different types of subsampling in different groups, omitting the activation function in the first two groups. The extensive results show that the proposed network outperforms its variants, and achieve significant improvement compared to the conventional steganalytic methods based on handcrafted features.

The rest of this paper is organized as follows. Section 2 describes the details of the proposed CNN architecture. Section 3 shows the experimental results. Finally, the concluding remarks and future works are given in Section 4.

2 THE PROPOSED CNN ARCHITECTURE

In this section, the overall architecture of the proposed CNN is first introduced, and then detailed analyses on different components of the proposed architecture are presented in the subsequent subsections.

2.1 Overall Architecture

The architecture of the proposed CNN is illustrated in Fig. 1. A convolutional layer with a fixed kernel $(-1, 2, -1)$ is placed at the beginning of the network, and then seven groups of layers (i.e., Group 1 to Group 7) are stacked one after another. Each group consecutively consists of a 1×5 convolutional layer, a 1×1 convolutional layer, and a subsampling layer. Among them, the 1×5 convolutional layer changes neither the number of channel nor the spatial size of the input data, while the 1×1 convolutional layer doubles the channel and the subsampling layer reduces the spatial size of the input data by half. Having been processed by the layer groups, the original data with size 16000 (see the experimental setups in Section 3.1) is finally transformed to a 512-D feature. This feature is then fed into a fully connected layer and a

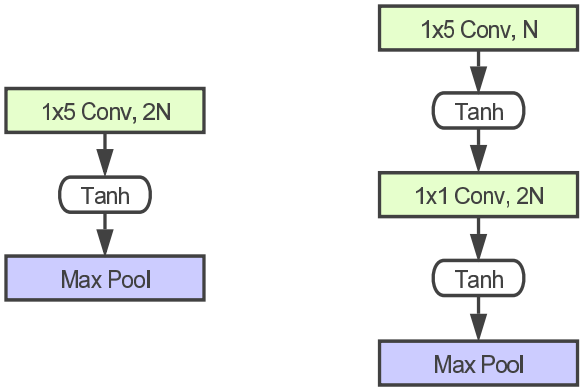


Figure 2: (a): a classical building block. (b): a building block with 1x1 convolutional layer.

softmax layer. This two layer perform like a classifier, producing the final output as the two class probabilities (i.e. cover or stego).

2.2 Fixed Convolutional Layer

CNN is a prevailing framework used in many tasks on image and audio classification. CNN has achieved great success because it is capable to learn discriminative features that represent the underlying properties of the original image/audio contents. However, steganalysis is different from the tasks in conventional image and audio classifications. In steganalysis, the key to perform a successful classification is capturing the artifacts introduced by steganography rather than modeling the media contents. Since the signal introduced by steganography is much weaker than the image/audio contents, applying CNN directly to the image/audio data may suffer from the negative impact of the contents, and thus leading to a poor local minima of the trained model. To solve this problem, some previous works such as [16, 25] on image steganalysis usually first applied a high-pass filtering to the input image, and then fed the filtered image (image residual) into a CNN architecture. In this paper, we try to attenuate the impact of audio contents in a similar manner, and use a convolutional layer with a kernel (-1, 2, -1) as it did in [14] to transform the input audio data into residuals. The kernel acts as a one dimensional high-pass filtering that suppresses the content of input data, thus it can prevent the model from learning the content features and make it more effective and robust. Unlike the common layer in CNN whose parameters are trainable, the parameters of this layer is fixed and thus we call it fixed convolutional layer.

2.3 1×1 Convolutional Layers

Typically, a building block of CNN consecutively consists of a convolutional layer, an activation function, and a subsampling layer, as shown in Fig. 2(a). Here we use a convolutional layer with size 1×5, the Tanh activation function and max pooling. To preserve enough feature information after subsampling, the convolutional layer right before the subsampling needs to increase the number of channels (usually doubles the channel). As a result, this layer tends to introduce a lot of parameters especially when the kernel size is large. Take Fig. 2(a) as an example, supposing the input channel of the 1×5 convolutional layer is N and the output channel is $2N$, the number of introduced parameters (including the weights and biases) is:

$$1 \times 5 \times N \times 2N + 2N = 10N^2 + 2N.$$

When $N = 100$, the number of parameters is up to 100200. As introducing too many parameters would lead to the danger of overfitting, we need an approach for reducing the parameters. To this end, we adopt the 1×1 convolutional layer into the classical CNN building block, which is also used in some famous CNN architectures such as GoogLeNet [23] and ResNet [3] for controlling the number of parameters. Specifically, we add the 1×1 convolutional layer to increase the number of channels, while keeping the number of output channels of the 1×5 convolutional layer the same as its input channels, as shown in Fig. 2(b). In this way, the total number of parameters is given as follows:

$$1 \times 5 \times N \times N + N + 1 \times 1 \times N \times 2N + 2N = 7N^2 + 3N.$$

We can see that the number of parameters decreases by $3N^2 - N$. When $N = 100$, the decrement of parameters is 29900, about 30% of the parameters used in a classical CNN block. Due to the reduction of parameters, including the 1×1 convolutional layers can prevent overfitting to some extent, and thus can improve the detection performance.

2.4 Subsampling Layers

In order to reduce the spatial size of previous feature map and extract robust invariant feature, it is very common to insert a subsampling layer right after one or more convolutional layers in CNN. Subsampling is usually performed by pooling layer such as max pooling or average pooling, and max pooling is the most popular choice. Recently, some research, such as [21], prefer performing subsampling by convolutional layers with stride larger than 1. Such a subsampling approach can outperform max pooling or average pooling in certain tasks. In the proposed network, we found that a convolutional layer with stride 2 is more suitable for extracting features in lower layers compared with max pooling. Therefore, a convolutional layer with stride 2 is used for subsampling in the first two groups, while max pooling is chosen in the deeper groups. In the last group, we adopt a type of average pooling called global average pooling [12], which uses a kernel size equal to the size of the feature map and thus sums up the whole feature map learned by previous layers. The average pooling is with size 1×250 and stride 250.

2.5 Activation Function

In the proposed network, we choose Tanh as the activation function instead of the more popular Relu [20]. The reason is that the saturation region of the Tanh limit the range of data value, and thus it can enhance the performance and robustness of our model (refer to Section 3.2, compared with

Table 1: Network Indices and Descriptions

Index	Network Description
#1	The proposed CNN
#2	Remove the fixed convolutional layer
#3	Replace the activation function "Tanh" with "Relu"
#4	Remove the fixed convolutional layer & retain the activation functions in Group 1 & 2
#5	All groups use convolutional layers to perform subsampling
#6	All groups use max pooling to perform subsampling
#7	The first two groups use max pooling while the others use convolutional layers to perform subsampling

network #3). Besides, we discover that removing the activation function in the first two groups can slightly improve performance. Similar operation is found in some existing forensic works, such as [1].

3 EXPERIMENTAL RESULTS

3.1 Experimental Setups

In our experiments, we randomly select uncompressed speech clips from the public data set [7], and cut then into 40,000 small clips. The duration of each audio clip is 1s and the sampling rate is 16kHz. We use the ± 1 LSB matching to obtain stego clips with an embedding rate of 0.50 bps (bit per sample). Totally, we obtain 40,000 cover-stego pairs. Half of the pairs are used for training, and the rest are used for testing. In the training stage, 4,000 pairs are set aside for validation, and the rest 16,000 pairs are used to train the network. To obtain convincing results, all the experiments are repeated 30 times by randomly splitting the training and testing data. In the following, we report the average results over 30 experiments for all the networks and two other conventional methods.

Instead of using the popular stochastic gradient descent (SGD), we use Adam algorithm[9] to train our model, which can make the model converge faster and perform better. The learning rate is fixed at 0.0001. We train the networks with 50,000 iterations, and in each iteration a mini-batch of 64 audio clips (32 cover/stego pairs) is used as input. At the beginning of the training, the trainable weights are initialized by random numbers generated from zero-mean truncated Gaussian distribution with standard deviation of 0.1, the trainable bias are initialized to zero. Please note that L2 regularization and Dropout [22] are not included because of little performance improvement based on our extensive experiments.

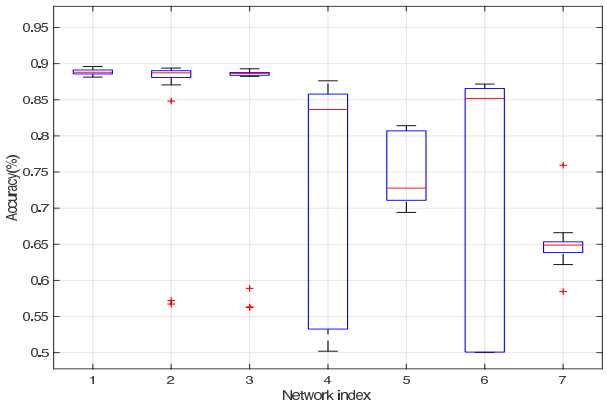


Figure 3: Box plots of the accuracies obtained by different networks.

Table 2: The average validation accuracy and the variance for each network

Index	Average	Variance
#1	**0.8885**	<0.0001
#2	0.8640	0.0063
#3	0.8551	0.0090
#4	0.7465	0.0226
#5	0.7512	0.0021
#6	0.6941	0.0327
#7	0.6472	0.0007

3.2 Comparison with Different Variants

In this experiment, we try to show the effectiveness of the proposed CNN by comparing it with its several variants. As listed in Table 1, six variant networks are used in the experiment, indexing from #2 to #7, whose components are slightly different from the proposed network #1. All the networks are trained for 30 times with the parameters described above. Since the accuracies of a certain network will fluctuate during training, and different networks may reach their highest accuracies after different iterations, it is unreliable to compare different networks using the accuracies after the same iteration. Therefore, during each time of training, the accuracy of validation set is checked every 1000 iterations and the highest validation accuracy is recorded as the accuracy for a network. In Fig. 3, we show the box plot of 30 accuracies for each network. After 30 times of training, we obtain the average accuracy and the variance of accuracies for each network, as show in Table 2.

From Fig. 3 and Table 2, we can observe that network #1 achieves the highest average accuracy of 88.85% and has a very stable performance over different times of training. Although network #2 and #3 achieve average accuracies between 85% and 87%, they sometimes fall into poor local minima and result in low accuracies. The average accuracies of the network #4, #5, #6 and #7 are between 64% and 76%,

Table 3: Accuracy Comparison

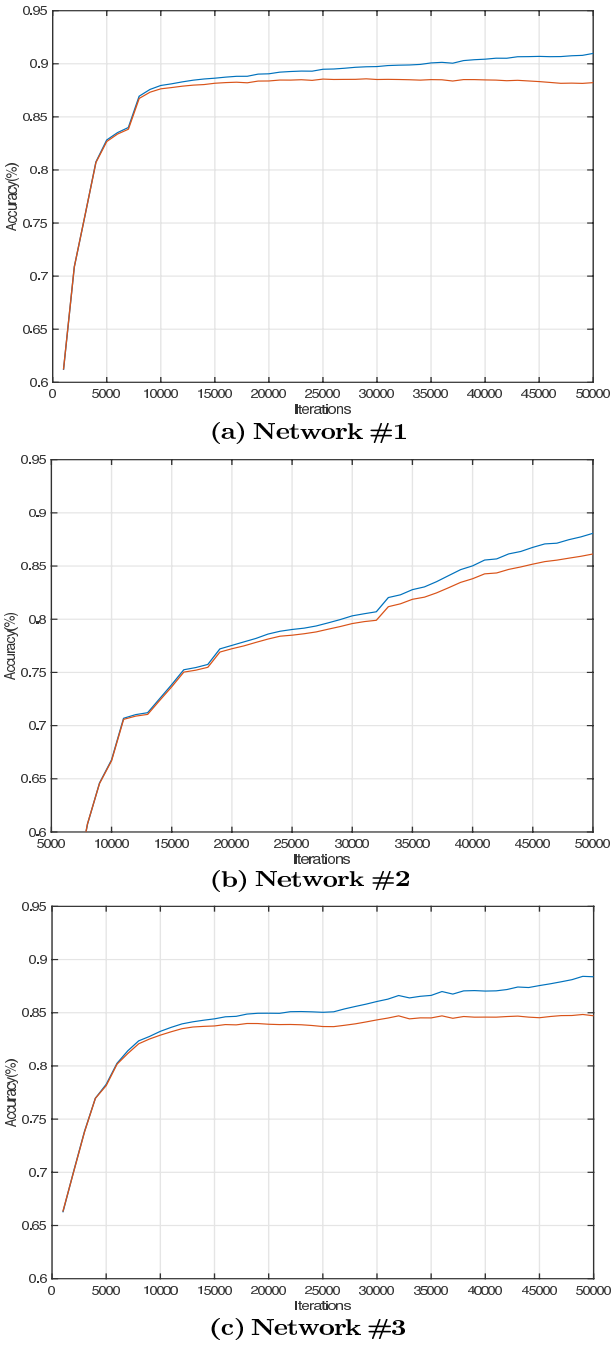
Approach	Accuracy
`Proposed CNN`	**0.8830**
`Method #1 [14]`	0.6313
`Method #2 [13]`	0.5449

which are much lower than that of the proposed network. Moreover, we can observe from Fig. 3 that the accuracies of network #4, #5 and #6 spread out in wide ranges, indicating that these networks are not stable enough.

To further evaluate the networks with relatively good performance (i.e., network #1, #2 and #3), we draw the curves of their training and validation accuracies during the training stage in Fig. 4. It is observed that the training accuracies of all these networks steadily increased with more iterations, while the validation accuracies of network #1 and #3 almost did not increase after about 10,000 iterations, meaning that network #1 and #3 both converged in this case and more iterations could not improve their validation accuracies. Network #2, which did not converge even after 50,000 iterations, can eventually converge after more iterations based on our experiment, although its final accuracy is still lower than that of network #1. Besides, we have additionally tested some other variants. For example, removing the 1×1 convolutional layers, keeping the activation function in early groups. The experimental results show that they always obtain lower accuracies compared with network #1, although the degradation of performance is not significant in some cases. As a result, the proposed network (#1) converges relatively fast and achieves the best validation accuracy, so it is the most effective network compared to other variants.

3.3 Comparison with Previous Methods

In this experiment, we compare the performance of proposed network with two typical steganalytic approaches (i.e. Method#1 [14], Method#2 [13], which employ conventional hand-crafted features and classification techniques. The classification accuracies on the testing data for different methods are given in Table 3. We can see that the proposed network achieves the classification accuracy of 88.3%, which is very close to the validation accuracy, meaning that the network does not suffer from overfitting. Compared with the conventional audio steganalytic methods Method#1 [14], Method #2 [13], the proposed method increases the testing accuracy by about 25% and 34%, respectively. Such a promising result indicates that the proposed network is very effective for audio steganalysis.



(a) Network #1

(b) Network #2

(c) Network #3

Figure 4: Accuracy curves of network #1, #2 and #3. Blue lines denote the training accuracy and orange lines denote the validation accuracy.

4 CONCLUSION

In this paper, we propose a CNN based method for digital audio steganalysis, and show that compared with the conventional steganalytic methods with handcrafted features, a well-designed CNN architecture can significantly improve the detection performances for audio stegangraphy in the time domain. We expect that some modern CNN architectures, such as ensembles of CNN [24], can be adopted for further improving the steganalytic performance.

In future, we will extend the proposed the CNN to detect other steganography in the frequency domain, such as MP3

and AMR, and detect various audio manipulations, such as scaling, compression, and electronic disguised voices.

ACKNOWLEDGMENTS

This work is supported in part by the NSFC (61672551), the Fok Ying-Tong Education Foundation (142003), and the Special plan of Guangdong Province (2015TQ01X365).

REFERENCES

[1] Belhassen Bayar and Matthew C Stamm. 2016. A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. In *ACM Workshop on Information Hiding and Multimedia Security*. ACM, 5–10.

[2] Jiansheng Chen, Xiangui Kang, Ye Liu, and Z. Jane Wang. 2015. Median Filtering Forensics Based on Convolutional Neural Networks. *IEEE Signal Processing Letters* 22, 11 (Nov 2015), 1849–1853.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[4] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Doug Paul John Garofolo, David Graff and David Pallett. 2013. Wall Street Journal Speech Database. https://catalog.ldc.upenn.edu/LDC93S6A

[8] Micah K. Johnson, Siwei Lyu, and Hany Farid. 2005. Steganalysis of recorded speech. *Proc. SPIE* 5681 (2005), 664–672.

[9] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Christian Kraetzer and Jana Dittmann. 2007. Mel-cepstrum-based steganalysis for VoIP steganography. *Proc. SPIE* 6505 (2007), 650505–650505–12. DOI: http://dx.doi.org/10.1117/12.704040

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[12] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[13] Qingzhong Liu, A.H. Sung, and Mengyu Qiao. 2009. Temporal Derivative-Based Spectrum and Mel-Cepstrum Audio Steganalysis. *IEEE Transactions on Information Forensics and Security* 4, 3 (Sept 2009), 359–368.

[14] Qingzhong Liu, Andrew H. Sung, and Mengyu Qiao. 2011. Derivative-based Audio Steganalysis. *ACM Transactions on Multimedia Computing Communications and Applications* 7, 3, Article 18 (Sept. 2011), 19 pages.

[15] Da Luo, Rui Yang, Bin Li, and Jiwu Huang. 2017. Detection of Double Compressed AMR Audio Using Stacked Autoencoder. *IEEE Transactions on Information Forensics and Security* 12, 2 (2017), 432–444.

[16] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. 2015. Deep learning for steganalysis via convolutional neural networks. *Proc. SPIE* 9409 (2015), 94090J–94090J–10.

[17] Mengyu Qiao, Andrew H. Sung, and Qingzhong Liu. 2013. MP3 audio steganalysis. *Information Sciences* 231 (2013), 123 – 134.

[18] Yanzhen Ren, Tingting Cai, Ming Tang, and Lina Wang. 2015. AMR Steganalysis Based on the Probability of Same Pulse Position. *IEEE Transactions on Information Forensics and Security* 10, 9 (Sept 2015), 1801–1811.

[19] Daniel Seichter, Luca Cuccovillo, and Patrick Aichroth. 2016. AAC encoding detection and bitrate estimation using a convolutional neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2069–2073.

[20] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[21] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[22] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.

[24] Guanshuo Xu, Han-Zhou Wu, and Yun Q Shi. 2016. Ensemble of CNNs for steganalysis: an empirical study. In *ACM Workshop on Information Hiding and Multimedia Security*. ACM, 103–107.

[25] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. 2016. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Processing Letters* 23, 5 (2016), 708–712.