Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 17(2): 1041–1058.
DOI: 10.3934/mbe.2020055
Received: 25 January 2019
Accepted: 24 October 2019
Published: 11 November 2019

Research article

A convolutional neural network-based linguistic steganalysis for synonym substitution steganography

Lingyun Xiang[1,2,3,*], Guoqing Guo[2], Jingming Yu[2], Victor S. Sheng[4] and Peng Yang[5]

[1] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha 410114, Hunan, China
[2] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, Hunan, China
[3] Hunan Provincial Key Laboratory of Smart Roadway and Cooperative Vehicle-Infrastructure Systems, Changsha University of Science and Technology, Changsha 410114, Hunan, China
[4] Department of Computer Science, University of Central Arkansas, Conway, AR, USA 72035
[5] Hunan Branch of CNCERT/CC, Changsha 410004, Hunan, China

* **Correspondence:** Email: xiangly210@163.com; Tel: +86-13874829742; Fax: +86-0731-85258462.

Abstract: In this paper, a linguistic steganalysis method based on two-level cascaded convolutional neural networks (CNNs) is proposed to improve the system's ability to detect stego texts, which are generated via synonym substitutions. The first-level network, sentence-level CNN, consists of one convolutional layer with multiple convolutional kernels in different window sizes, one pooling layer to deal with variable sentence lengths, and one fully connected layer with dropout as well as a softmax output, such that two final steganographic features are obtained for each sentence. The unmodified and modified sentences, along with their words, are represented in the form of pre-trained dense word embeddings, which serve as the input of the network. Sentence-level CNN provides the representation of a sentence, and can thus be utilized to predict whether a sentence is unmodified or has been modified by synonym substitutions. In the second level, a text-level CNN exploits the predicted representations of sentences obtained from the sentence-level CNN to determine whether the detected text is a stego text or cover text. Experimental results indicate that the proposed sentence-level CNN can effectively extract sentence features for sentence-level steganalysis tasks and reaches an average accuracy of 82.245%. Moreover, the proposed steganalysis method achieves greatly improved detection performance when distinguishing stego texts from cover texts.

Keywords: steganalysis; steganography; synonym substitution; word embedding; convolutional neural network

1. Introduction

Due to the explosion of data on the Internet, information security has attracted increasing attention worldwide [1]. Recently, given the growing desire to ensure information security, a technique known as linguistic steganalysis, which serves as the counter-technique of linguistic steganography, has been extensively developed. The main goal of linguistic steganalysis is to detect the existence of secret messages in natural texts. These messages are usually embedded via natural language processing techniques, which are utilized to create equivalent linguistic transformations such as synonym substitution [2] or syntactic transformation [3]. Thus, linguistic steganalysis can prevent covert communication among criminal offenders who are exploiting linguistic steganography.

Since English is rich in synonyms, synonym substitution can provide a relatively higher embedding capacity; this has made synonym substitution-based linguistic steganography one of the most popular and dominant methods at present. Thus, most researchers currently focus on steganalysis against the linguistic steganographic method, which involves substituting words with their synonyms to hide messages, to ensure information security. The first related work is the N-gram language model-based steganalysis by Taskiran et al. [4]. This work extracted features from the N-gram language model to distinguish between unmodified and steganographically modified sentences. However, its performance was not satisfactory.

As unmodified sentences and their corresponding, steganographically modified sentences are semantically similar and their differences are very slight, sentence-level steganalysis is a very challenging task. Researchers typically focus on text-level linguistic steganalysis [5–10], which is aimed at identifying the stego texts among the cover texts to reveal the presence of hidden information in a text rather than in a sentence. In the current literatures, this type of linguistic steganalysis method formulates the steganalysis task as a binary classification problem involving distinguishing stego texts from cover ones. This generally includes two main processes: feature extraction and feature classification. The feature extraction process usually involves extracting a set of handcrafted features from each text in order to capture the impact on the linguistic and statistical characteristics made by information embedding operations. In the feature classification process, classifiers such as the Bayesian classifier, support vector machine [11], ELM [12], etc. are trained using the extracted features.

In this type of linguistic steganalysis method, the extracted features are the most critical aspect and generally determine the detection performance. Yu et al. [9] extracted statistical features by analyzing the suitability of a synonym in its context, which was weighted by the words' IDF (inverse document frequency). Similarly, Chen et al. [5] estimated context fitness by introducing the context cluster, which is composed of a synonym and its contextual words, to extract distinguishable features. Moreover, Chen et al. [6] derived features from the expectation and variance of the natural relative frequency (NRF) values of a word and its synonymous words. Later, Xiang et al. [7] also investigated relative-frequency-based features. These authors sorted the synonymous words by their word frequencies in a certain order to divide all synonyms employed into different categories. Synonyms in the same category were designated as belonging to the same attribute pair. Finally, the detection features were extracted from the statistical characteristics of the attribution pairs in the text. Motivated by the development of word representation, [8] first introduced word embeddings to represent a synonym and its contextual words in order to measure their semantic distance and context fitness. As a result, effective detection features could be extracted to characterize the changes caused by synonym substitutions,

which promoted the improvement of the steganalysis method.

The extracted features in the abovementioned forms of linguistic steganalysis have made great contributions to the detection of stego texts generated by synonym-substitution-based steganography. However, they mainly depend on hand-crafted design. Whether or not effective or practical features can be extracted for linguistic steganalysis tasks is greatly determined by researchers' ability to deal with natural language understanding and text processing. Owing to the lack of mature language models for processing texts, it is difficult to find a way to perfectly represent the semantic information in a text to capture subtle steganographic changes; thus, extracting hand-crafted features in the field of linguistic steganalysis is extremely challenging and gives rise to great difficulties. In particular, given the increased sophistication of linguistic steganography [13–15], more complex statistical and linguistic dependencies among individual words have been considered in order to reduce steganographic distortion [16]. Thus, it is necessary to incorporate more complex, effective and high-dimensional linguistic and statistical features into linguistic steganalysis in order to capture the impact of steganography on the original text as sensitively as possible. On the other hand, the feature extraction process is separate from the feature classification process; accordingly, the useful information in the extracted features cannot be fully captured by classifiers, as they cannot be optimized simultaneously.

Motivated by the above problems, the present paper aims to employ deep learning frameworks in order to learn feature representations automatically, as well as optimize the feature extraction and classification in a unified framework for linguistic steganalysis purposes. In recent years, deep learning frameworks have achieved great success in many fields of computer vision [17–19], natural language processing [20, 21], etc. Researchers have also tried to investigate the potential in the fields of image steganography [22] and steganalysis [23]. As early as 2014, Tan and Li [23] first introduced deep learning architecture in image steganalysis. They constructed a nine-layer Convolutional Neural Network (CNN)-based blind steganalysis method to distinguish stego images from cover images. Subsequently, a series of studies on image steganalysis using deep learning frameworks were conducted [24–27]. These studies all employed CNN and its variants to carry out feature learning, as well as to calculate effective residual signals by adjusting the convolution kernel or updating the kernel parameters, thus obtaining improved steganalysis performance.

Although deep learning has been successfully applied in image steganalysis tasks, the related frameworks and methods cannot be applied to linguistic steganalysis directly. As a subcategory of digital signal processing, digital image processing for image steganalysis has more advantages than natural language processing. Image steganalysis allows for a much wider range of mathematical operations to be applied to the input data, which can be fed directly into the deep learning algorithm. However, the natural text (composed of character symbols) must first be transformed into digital signals for deep learning models to be effective. A good representation is able to capture rich semantic information and can thereby help to improve the performance of deep learning models. In addition, the size of an image is determined by only two parameters; while it is easy to tune the images in a deep learning model to a fixed size, the length of a text is indefinite, meaning that it varies over a large range. Accordingly, this paper proposes a two-level cascaded convolutional neural network (CNN) for text-level linguistic steganalysis, which aims to detect the stego texts generated by means of synonym substitution-based steganography.

The proposed steganalysis method, which is based on two-level cascaded CNNs, first builds a sentence-level CNN to automatically learn two steganographic features from each sentence; this can

also provide an effective method for sentence-level steganalysis tasks. The words, including the synonyms and their contextual words in a sentence, are represented as word embeddings in order to form the input. With respect to the length of sentences, the training dataset of a sentence-level CNN can be scaled far more easily to accommodate huge amounts of sentences than is the case for a text-level CNN. Given the limited memory capacity of GPUs, moreover, it is difficult to train a CNN-based model directly on a large number of texts with variable length for text-level steganalysis tasks. Thus, for text-level steganalysis tasks, we first build a sentence-level CNN to optimize the architecture of the proposed text-level steganalysis. Subsequently, the steganographic features of the sentence are inputted into the second-level CNN to facilitate the classification of stego texts and cover texts. Experimental results demonstrate that the detection performance of the proposed two-level cascaded CNN-based steganalysis method is higher than that of other similar steganalysis methods. Moreover, by using the features automatically learned from the sentence-level CNN, we are able to obtain comparable performance when distinguishing between unmodified cover sentences and steganographically modified stego sentences.

The remainder of this paper is organized as follows. In Section 2, we describe the framework of the proposed two-level cascaded CNN-based linguistic steganalysis in detail. Experimental results and analysis are presented in Section 3. Finally, a conclusion is drawn in Section 4.

2. Two-level cascaded CNN-based linguistic steganalysis

2.1. General framework

In order to improve the performance of detecting synonym-substitution-based stego texts, we propose a linguistic steganalysis method via two-level cascaded CNNs, the framework of which is shown in Figure 1. CNN is one of the most representative deep learning frameworks, given its increasing hierarchical complex feature representations and superior classification performance on other artificial intelligence-related tasks [28]. A stego or cover text contains an indefinite number of words, which will be represented as 100, 200 or higher-dimensional word embeddings; thus, when all the words in a text are inputted to train a CNN model for a text-level steganalysis task, it is difficult for the CNN to achieve a good effect. Thus, our proposed method makes use of a two-level architecture with cascading CNNs. The CNN at the first level, called sentence-level CNN, takes the unmodified and steganographically modified sentences as the input to automatically learn steganographic features from the sentences. The outputs of the sentence-level CNN can be regarded as strong prior knowledge for the following second level CNN, which finishes the text-level steganalysis task. The CNN at the second level, called the text-level CNN, pays more attention to the classification of stego texts and normal texts using the sentences' steganographic features sent from the sentence-level CNN.

1) Sentence-level CNN

As its basic framework, the sentence-level CNN at the first level employs the CNN architecture first proposed by Yoon Kim [29] for sentence-level classification tasks. The upper part of Figure 1 depicts the architecture of extracting sentence-level steganographic features using the sentence-level CNN. This primarily involves the following five parts: *Word representation*, *Multi − kernel convolution*, *Pooling*, *Fully connected* and *Feature output*.

In the first part, each word in a sentence is mapped into a word embedding, which refers to the low-dimensional dense vector representations of natural words; this forms the input of the next part of

the convolutional layer. These word embeddings can be trained by means of distributed representation methods to capture the semantic and syntactic information effectively. As the sentence-level CNN is constructed for the task of extracting features in order to discriminate between an unmodified cover sentence and a stego sentence that has been steganographically modified by means of synonym substitutions, only those sentences that include synonyms are fed into the CNN. If a sentence contains n words, each of which is expressed as a word embedding with m dimensions, then an $n \times m$ matrix is obtained as the input of the next convolutional layer.

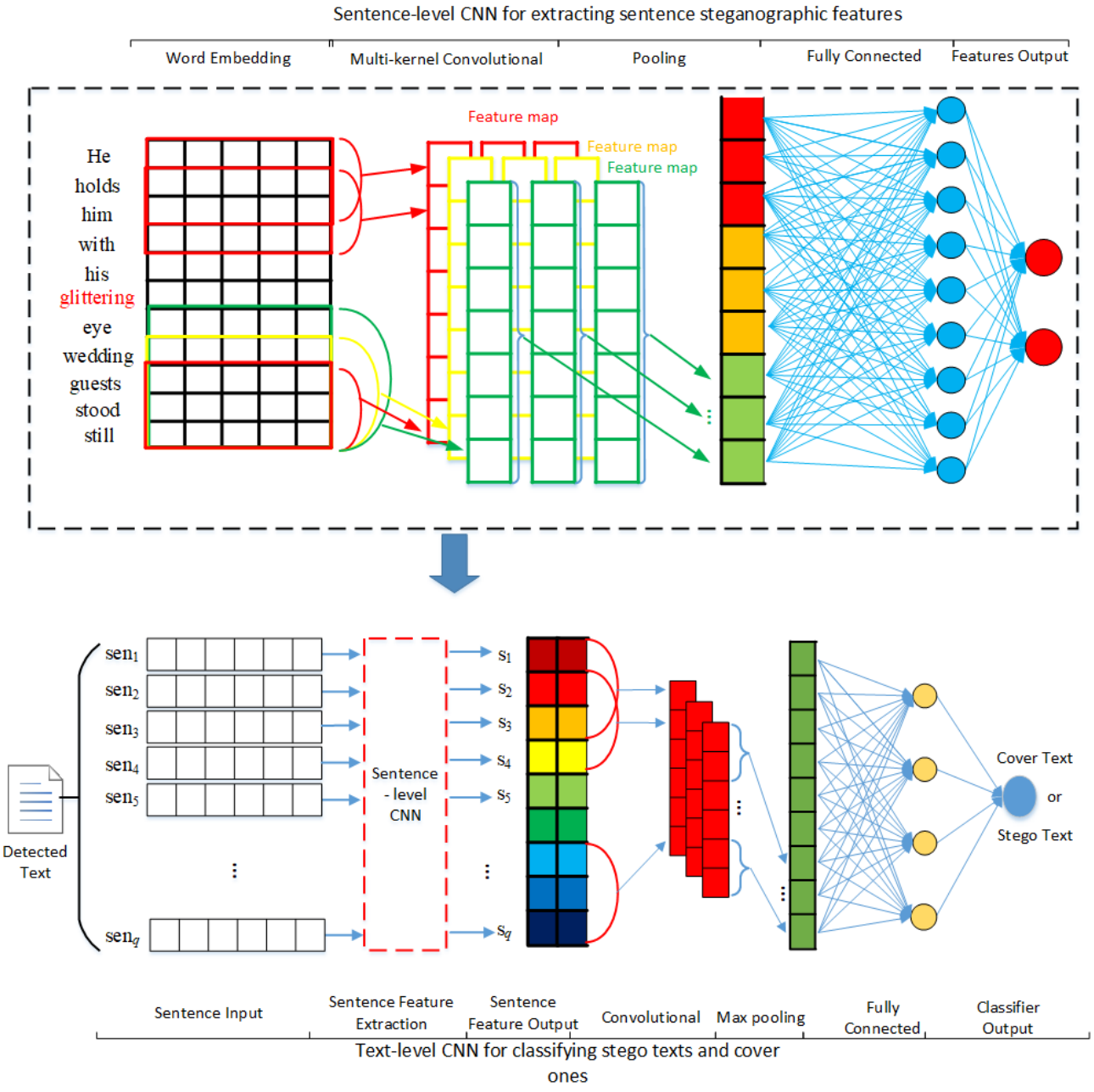

Figure 1. The framework of the proposed linguistic steganalysis based on two-level cascaded CNNs.

The next part of the convolutional layer is the core of the sentence-level CNN. Through combining trainable filters with multiple convolutional kernels, higher-order hierarchical and complex features can be automatically generated and outputted to the pooling layer. In order to ensure that the convolutional kernel does not truncate the vector representation of a word, each convolutional kernel is repeatedly worked on the input matrix row by row. In this way, the width of the convolutional kernel is selected to be the same as the width of the input matrix, which is also the dimensions of the word embedding. Moreover, the height (i.e. window size) is freely chosen and is usually set to 2-5 words. Different convolutional kernels with different window sizes will learn various different features. In the figure above, multiple convolutional kernel with three window sizes are selected to learn the respective feature maps from the input matrix.

The pooling layer is applied over the feature maps from the convolutional layer. It samples the features to solve the problem of the input sentences having different lengths and outputs the local optimal features. The convolutional and pooling operations result in the automatic learning of the sentence's steganographic features. Different features of the pooling layer obtained from different convolutional kernels are concatenated into one complete feature vector, which is then input to the fully connected layer. For its part, the fully connected layer utilizes the softmax function to output the final two steganographic features, which are the probabilities that the sentence is a stego sentence or a cover sentence, respectively.

2) Text-level CNN

In this paper, the method proposed for the text-level steganalysis task is formulated as a two-stage classification via two-level cascaded CNNs. The sentence-level CNN at the first level is responsible for extracting sentence-level steganographic features; this can be employed to classify stego and cover sentences. The text-level CNN at the second level takes the output of the sentence-level CNN and uses it to discriminate between stego and cover text. As only two features are outputted by the sentence-level CNN, the architecture of the text-level CNN is much simpler, as can be seen in the bottom part in Figure 1.

Firstly, each sentence that contains synonyms in a detected text is inputted into the pre-trained sentence-level CNN to enable its steganographic features to be learned and the sentential clues to be captured. All the learned features are then concatenated into a feature vector, which forms the input of the convolutional layer in the second-level CNN. The convolutional layer, pooling layer and the next fully connected layer use the features of the sentences to extract the text-level steganographic features. Finally, the classifier output calculates a confidence score for each text category candidate; this score is employed to classify the stego text and cover text. During training of the text-level CNN, all parameters are learned automatically, and the text feature extraction and classification are optimized in a single framework.

In the next subsection, we describe the above two-level CNNs in more detail.

2.2. Extracting sentence features using the sentence-level CNN

The sentence-level CNN is designed to automatically learn sentence-level steganographic features in order to capture the differences between stego and cover sentences. The steps involved are described in more detail below.

1) Word representation

In order to properly capture the semantics of natural language words, particularly synonymous

words that enable the CNN to learn more effective steganographic features, we employ word representation [30] to represent a word as a low-dimensional vector. Word representation, which involves constructing a feature vector that is associated with each word, is a very worthwhile pursuit in this context. Each feature corresponds to the value of each dimension, which might have a semantic or grammatical interpretation. As word representation is a basic building block in many natural language processing tasks, it has thus received increasing attention in the field of Natural Language Processing (NLP). Currently, the most popular word representation method is distributed representation, which involves representing a word using a dense real-valued low-dimensional vector, which is also called word embedding. A word embedding represents latent features of the word, which are typically induced by neural language models [31]. Most works focusing on learning word embeddings are based on the idea of modeling the semantic relationship between a word and its context words. The most representative of these works was by Bengio et al. [32], who employed a feedforward Neural Network Language Model (NNLM) to learn word representations. In order to eliminate linear dependency on vocabulary size for the training and testing of neural language models, some works [33, 34] have made efforts to scale to very large training corpora. Moreover, Mikolov et al. [35] proposed two novel model architectures, namely Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram, to improve the quality of word embeddings and the training speed. They computed the continuous vector representations of words from very large training corpora, resulting in high-quality word embeddings. Word embeddings are widely used in NLP tasks to improve performance [36, 37].

For the sentence-level CNN, we begin our sentence feature extraction with learning the embedding of words, which captures fine-grained syntactic and semantic regularities. We adopt the Skip-gram language model to learn word embeddings by training large-scale corpora to gradually converge words with similar meanings to nearby areas in the vector space. The Skip-gram model is a bag-of-words model, the goal of which is to learn word representations that can adequately predict a center word's neighborhood words within a certain context window. The Skip-gram model has a very simple structure and is extremely efficient and effective, especially for infrequent words. To the best of our knowledge, some synonyms employed in synonym substitution-based linguistic steganography to embed secret messages have very low frequencies. The word embeddings of these low-frequency synonyms will exert an important influence on the linguistic steganalysis. Thus, to improve the performance of the learned word embeddings, we choose the Skip-gram model, which is more appropriate than CBOW for our purposes when engaging in linguistic steganalysis tasks.

For the synonym substitution-based steganography, which is the detection target of our proposed linguistic steganalysis, synonym substitutions generally only affect the local parts of words; that is, they only impact on the quality of the sentence in which a synonym is located. Thus, to train a sentence-level CNN for sentence-level steganalysis, we only use the sentences containing synonyms, which are recognized using a pre-prepared synonym database.

When learning word embeddings by CBOW, each word in a sentence is represented as a dense low-dimensional word vector. Let $\vec{V}_i \in \mathbb{R}^m$ be the m-dimensional word embedding corresponding to the i-th word in a cover or stego sentence. The number of words in the sentence is fixed at n. The sentence can thus be represented as follows:

$$\vec{V}_{1:n} = \vec{V}_1 \oplus \vec{V}_2 \oplus \cdots \oplus \vec{V}_n \tag{2.1}$$

where $\oplus$ is the concatenation operator, and $\vec{V}_{i:i+j}$ denotes the concatenation of the word embeddings of

words $\vec{V}_i$, $\vec{V}_{i+1}$, $\cdots$, $\vec{V}_{i+j}$. If the length of a sentence is not n, $\vec{V}_{1:n}$ is padded with zeros. The sentence is then transformed into a matrix $\vec{V}_{1:n} \in \mathbb{R}^{n\times m}$, which is fed into the next convolutional layer.

2) Multi-kernel convolutional layer

In order to capture the compositional semantics of a sentence, we employ multiple convolutional kernels with different window sizes to extract multiple features in the convolutional layer simultaneously to improve the reliability of the final sentence's steganographic features. Each convolutional kernel with a fixed window size can produce a feature in a window. Suppose a convolutional kernel involves a filter $\mathbf{w} \in \mathbb{R}^{h\times m}$, which is applied to a window that includes h word embeddings. Thus, a feature c_i is produced after applying a convolutional kernel on a window of word embeddings $\vec{V}_{i:i+h-1}$ according to the following equation:

$$c_i = f(\mathbf{w} \cdot \vec{V}_{i:i+h-1} + b) \tag{2.2}$$

where b is a bias term, and $f(\cdot)$ is a non-linear activation function such as rectified linear units (ReLU). By sliding the convolutional kernel to the input sentence and extracting the features of each possible window, the n words in the sentence can be divided into $n - h + 1$ possible windows, which are $\vec{V}_{1:h}, \vec{V}_{2:h+1}, \cdots, \vec{V}_{n-h+1:n}$; thus, a feature map $\mathbf{c}$ including $n - h + 1$ features is produced, and can be expressed as follows:

$$\mathbf{c} = [c_1, c_2, \cdots, c_{n-h+1}] \tag{2.3}$$

where $\mathbf{c} \in \mathbb{R}^{n-h+1}$.

One feature map is extracted from one convolutional kernel with a fixed window size. To capture different feature maps, let us suppose that we utilize l convolutional kernels , such that $W = \mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_l$ feature maps will be produced for window size h. A feature map $\mathbf{c}_j$ can be expressed as:

$$\mathbf{c}_j = [c_{j1}, c_{j2}, \cdots, c_{j(n-h+1)}] \tag{2.4}$$

where j ranges from 1 to l.

It should be noted here that the window size h can be set to different values. In this paper, we set h to be 3, 4, and 5. For each window size, l feature maps will be generated. Thus, the convolutional layer will generate $3 \times l$ feature maps in total.

3) Pooling layer

In image processing tasks, all input matrices of CNN can be easily unified to the same fixed size. By contrast, in our task, the size of the input and output matrices of the convolutional layer vary with the sentence length n. In order to solve this problem, a max-over-time pooling in the pooling layer is applied to enable learning of a fixed-length feature vector by sampling the feature maps from the convolutional layer. The max-over-time pooling is a good choice when aiming to capture the semantics of long-distance words within a sentence [38]; this will help in capturing the semantic changes caused by synonym substitutions to embed information into a sentence.

For the max-over-time pooling, one feature map is taken as a pool, and only one max value is extracted as the most important feature within each feature map. Through implementing the max-over-time pooling operation over the feature map $\mathbf{c}_j$, the most important feature $\hat{c}_j$ with the highest value corresponding to the jth convolutional kernel with window size h is obtained by means of Eq.2.5, where $1 \le j \le l$:

$$\hat{c}_j = max\{\mathbf{c}_j\} \tag{2.5}$$

For each window size of h, l features are captured. There are three different window sizes; thus, in the end, $3 \times l$ features are obtained and denoted as $\{\hat{c}_{ij}\}$, where $1 \leq i \leq 3$ and $1 \leq j \leq l$. We then concatenate all $\hat{c}_{ij}$ j to form a feature vector $\hat{z} \in \mathbb{R}^{3l}$, which can be expressed as follows:

$$\hat{z} = \{\hat{c}_{11}, \hat{c}_{12}, \cdots, \hat{c}_{1l}, \hat{c}_{21}, \hat{c}_{22}, \cdots, \hat{c}_{2l}, \hat{c}_{31}, \hat{c}_{32}, \cdots, \hat{c}_{3l}\} \tag{2.6}$$

It can be clearly seen that this pooling scheme naturally deals with variable sentence lengths. The dimension of the output $\hat{z}$ in this pooling layer is simply related to the number of convolutional kernels employed and their window sizes. $\hat{z}$ can be considered as the higher-level features, and will be fed to the next fully connected layer.

4) Fully-connected layer and output

To extract the final steganographic features of the sentence, the learned feature vector $\hat{z}$ is passed to a fully connected layer. In the fully connected layer, a two-way softmax activation function is used to produce a distribution over the two class labels (i.e. the stego sentence and cover sentence), as follows:

$$\mathbf{s} = g(\mathbf{w_g} \cdot \hat{z} + b) \tag{2.7}$$

where $\mathbf{s}$ is the final output, $g(\cdot)$ is a softmax function whose output is the probability distribution over labels, and $\mathbf{w_g}$ is a weight matrix applied to the features $\hat{z}$.

To minimize the over-fitting problem caused by excessive network structure parameters and insufficient training data, regularization is necessary for dealing with neurons. Thus, to regularize the fully connected layer, we apply the dropout on the Pooling layer with a constraint on the l_2-norms of the weight vectors. When training using dropout, the dropout operation randomly sets the weight of some neurons to zero with a probability pre-established during forward backpropagation. In essence, the dropout will prevent the co-adaptation of hidden units by randomly dropping out a proportion p of the hidden units.

Once the dropout function is included, Eq.2.8 is used instead of Eq.2.7 for output unit s in the fully connected layer in forward propagation:

$$\mathbf{s} = g(\mathbf{w_g} \cdot (\hat{z} \otimes \mathbf{r}) + b) \tag{2.8}$$

where $\otimes$ is the element-wise multiplication operator, while $\mathbf{r} \in \mathbb{R}^{3l}$ is a 'masking' vector that is randomly generated using Bernoulli random variables with a probability p of being 1. Thus, the use of $\hat{z} \otimes \mathbf{r}$ means that an element value in $\hat{z}$ corresponding to zero in $\mathbf{r}$ will be invalid; if this occurs, it will be masked and its gradient cannot be back-propagated (gradients are only back-propagated through the unmasked elements).

Finally, the sentence-level CNN outputs a two-dimensional vector $\mathbf{s} = \{s_1, s_2\}$ for each sentence, where s_1 and s_2 represent the probability that the sentence belongs to the class of stego sentence and cover sentence, respectively. $\mathbf{s}$ is considered as the higher-level sentence features for the steganalysis task.

2.3. Stego text classification using the text-level CNN

Incorporating the knowledge behind the sentence-level CNN model, the text-level CNN takes the sentence steganographic features as input and attempts to reconstruct them via CNN over the training text samples in order to obtain a discriminative model for the text-level steganalysis task. Similar to

the sentence-level CNN, the text-level CNN has a feature extraction structure with an input layer, a convolutional layer, a pooling layer and a fully connected layer.

In the input layer, a detected text is broken into sentences and any sentences without synonyms are deleted. Each of the remaining sentences (containing synonyms) is inputted into the sentence-level CNN for extraction of the sentence-level steganographic features. If we suppose that the detected text contains q sentences with synonyms, q sentence steganographic feature vectors are obtained by employing the sentence-level CNN. Thus, a detected text can be represented as follows:

$$\mathbf{s}_{1:q} = \mathbf{s}_1 \oplus \mathbf{s}_2 \oplus \cdots \oplus \mathbf{s}_q \tag{2.9}$$

where $\mathbf{s}_i$ is the two-dimensional feature vector of the ith sentence derived from the sentence-level CNN. If the number of sentences with synonyms in the detected text is not q, we will pad $\mathbf{s}_{1:q}$ with zeros. $\mathbf{s}_{1:q}$ will then be fed into the next convolutional layer.

In the convolutional layer, suppose the window size of a convolution kernel is h and the convolutional kernel is a filter $\mathbf{w}_s \in \mathbb{R}^{h\times 2}$. When the convolutional kernel is applied to the input window $\mathbf{s}_{i:i+h-1} = \mathbf{s}_i \oplus \mathbf{s}_{i+1} \oplus \cdots \oplus \mathbf{s}_{i+h-1}$, the new feature t_i is produced according to the following equation:

$$t_i = ReLU(\mathbf{w}_\mathbf{s} \cdot \mathbf{s}_{i:i+h-1} + b_s) \tag{2.10}$$

where b_s is a bias term and $ReLU$ is an activation function. By obtaining features from each possible window of the input $\mathbf{s}_{1:q}$, a feature map $\mathbf{t}$ is produced:

$$\mathbf{t} = [t_0, t_1, \cdots, t_{q-h+1}] \tag{2.11}$$

In the pooling layer, the max-pooling method is used to reduce the number of dimensions of the feature map outputted from the convolutional layer. Unlike the max-over-time pooling method used in the sentence-level CNN, extracting multiple values (rather than only the maximal value) from the entire feature map can save more text feature information. In particular, the matrix inputted to the convolutional layer of the text-level CNN is simpler than that of the sentence-level CNN. Therefore, we consider a local max pooling strategy, which performs pooling over small local regions rather than over the entire feature map [39]. Assume that the size of the local region is k; thus, each local region including k features from the feature map $\mathbf{t}$ will generate a max value from pooling. These values can be concatenated to form a feature vector $\mathbf{t}$, which can be expressed as follows:

$$\hat{t} = \left(max\left\{t_{0:k-1}\right\}, max\left\{t_{k:2k-1}\right\}, \cdots, max\left\{t_{\left(\left\lfloor\frac{q-h+1}{k}\right\rfloor-1\right)*k:\left\lfloor\frac{q-h+1}{k}\right\rfloor*k-1}\right\}\right) \tag{2.12}$$

Each feature map will generate a feature vector from pooling. As the convolutional layer will employ multiple convolutional kernels to produce many feature maps, many feature vectors will thus be generated from the pooling layer. All obtained feature vectors will then be concatenated together to form a single feature vector for the next fully connected layer. Assuming that the input feature vector of the fully connected layer is $\hat{t}$, we first perform a ReLU activation function to obtain a feature vector T:

$$T = ReLU\left(\mathbf{W}_\mathbf{t} \cdot \hat{t} + b_t\right) \tag{2.13}$$

where $\mathbf{W}_\mathbf{t}$ is a weight matrix applied to the features $\hat{t}$, while b_t is a bias term. Finally, a softmax activation function is employed to produce a classification result for the stego and cover texts. The final output of the text-level CNN is L, as follows:

$$L = softmax\left(\mathbf{W}_\mathbf{L} \cdot T + b_L\right) \tag{2.14}$$

where $\mathbf{W_L}$ is a weight matrix applied to the features T, while b_L is a bias term. According to L, a detected text can be successfully recognized as either a stego text or a cover text.

3. Experimental results and analysis

In this section, we present the results of several experiments designed to test both the proposed sentence-level CNN (for the sentence-level steganalysis task) and the proposed two-level cascaded CNNs (for the whole-text-level steganalysis task). To validate the effectiveness of the sentence-level CNN, we classify stego sentences and cover sentences using two different pooling methods by using the obtained sentence steganographic features. We then demonstrate the effectiveness of the proposed two-level cascaded CNN-based steganalysis through comparison with three similar methods on the same datasets.

3.1. Experimental setup

1) Model Training

The sentence-level CNN takes the word embeddings of the words in a sentence as input. In these experiments, we employ Google's open source implementation, Word2vec, with the Skip-gram language model to train the Gutenberg corpus* in order to obtain word embeddings, which is consistent with the approach used in the experiments in Reference [8]. Each learned word embedding is only 100 dimensions.

When setting the parameters for the sentence-level CNN, three kinds of convolutional kernels with window sizes of 3, 4, and 5 are selected. For each window size, l is set to 100; namely, 100 convolutional kernels are randomly generated to obtain multiple features. In the fully connected layer, the probability of dropout is set to 0.5. Cross-entropy loss is used as the loss function, and the Adam update rule is utilized to reduce stochastic gradient descent (SGD) for random small-batch processing. The learning rate is set to 0.001, and 10% of training sentences are randomly selected for verification.

We next configure the text-level CNN. In the convolutional layer, 32 convolutional kernels with a window size of 5 are selected. In the pooling layer, the size of the local region k is set to 26, which is estimated by the minimum number of synonyms contained in a single text in the training set. The fully connected layer uses 256 neurons; the learning rate is set to 0.01, and categorical-cross-entropy loss is employed as the loss function. In the training process, 10% of the training texts are randomly selected for verification.

2) Data Preparation

We utilized the same cover and stego text sets (Tlex-25%, Tlex-50%, Tlex-75% and Tlex-100%) as in Reference [8]. The cover text set contains 5000 texts, which are selected from the Gutenberg corpus. Each cover text is embedded with random secret information using the T-lex steganographic tool†; embedding rates of 100%, 75%, 50%, and 25% are used. The generated stego texts are used to form the corresponding four stego text sets. Here, the embedding rate means the ratio of the total number of synonyms embedded with secret information to the total number of synonyms appearing in a text [7]. To the best of our knowledge, the T-Lex is the only text-level linguistic steganography

*https://www.gutenberg.org/

†http://www. imsa. edu/k̃eithw/tlex

system based on synonym substitution available on the Internet. Currently, all linguistic steganalysis against this kind of steganography has mainly attacked T-Lex or one of its variants.

Since we expect that the improvement of the proposed sentence-level CNN will simultaneously improve the performance of the proposed steganalysis, it is necessary to evaluate the effectiveness of the sentence-level CNN for classifying stego sentences and cover sentences. We therefore prepare the sentence sets for the sentence-level steganalysis task. Here, all stego and cover sentences contain synonyms, which can be employed for the embedding of secret information. Moreover, the cover sentences are original and unmodified, while the stego sentences are steganographically modified by means of synonym substitution for information embedding. From the prepared cover text set and four stego text sets, we directly extracted 2,025,199 cover sentences and 970,037 stego sentences for training the sentence-level CNN, while 1,605,752 cover sentences and 893,220 stego sentences were used as the test set. Before training or testing, all duplicate cover and stego sentences are deleted. We denote the newly generated sentence set as 'tlex_ sen'; this includes all sentences used for training and testing.

It is worth noting here that not all synonyms in a stego text with an embedding rate below 100% are embedded information. The synonyms without hidden information remain unmodified; thus, the sentences containing these kinds of synonyms will be regarded as cover sentences, even though they are located in a stego text. In addition, only some synonyms with hidden information are performed synonym substitutions, as when a synonym is fortunately encoded as the embedded secret bits, no synonym substitution is required. As a result, parts of sentences containing hidden information should also be treated as cover sentences. Therefore, there are more cover sentences than stego sentences extracted from the text sets. In order to balance the proportion of cover and stego sentences for training, we thus enhanced the training set in tlex_sen before training began. Moreover, in order to guarantee the uniqueness of the sentences, we randomly selected cover sentences for random substitution of synonyms so as to generate more stego sentences. Finally, the number of stego sentences was equal to that of cover sentences for training. We named the updated sentences set 'tlex_sen_bal'. The details of the sentence sets are presented in Table 1.

Table 1. Sentence sets for sentence-level CNN model training and testing.

	Training		Test	
	cover sentence	stego sentence	cover sentence	stego sentence
tlex_sen	2,025,199	970,037	1,605,752	893,220
tlex_sen_bal	2,025,199	2,025,199	1,605,752	893,220

3.2. Evaluating sentence-level CNN performance for sentence-level steganalysis task

In order to test the performance of the sentence-level CNN model, we employed its extracted sentence steganographic features to classify the stego and cover sentences. The training and test sentences in tlex_sen_bal are used to train and test the sentence-level CNN. Here, we evaluate the reliability of the sentence-level CNN in terms of its classification of stego and cover sentences through three performance measures: *Precision*, *Recall*, *Accuracy*. These are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.3}$$

Here, TP is the number of correctly predicted stego sentences, TN is the number of correctly predicted cover sentences, FP is the number of cover sentences incorrectly predicted as stego sentences, and FN is the number of stego sentences incorrectly predicted as cover sentences.

In addition, in order to investigate the effect of different pooling strategies, we also perform average pooling instead of max-over-time pooling to produce different feature vectors from the same feature maps. The test results of the sentence-level CNN for the sentence-level steganalysis task are listed in Table 2. The experimental results demonstrate that the sentence-level CNN can effectively discriminate between stego and cover sentences, regardless of which pooling strategy is employed. However, the sentence-level CNN with max-over-time pooling achieves 82.25% accuracy, which is slightly higher than that obtained by the CNN with average pooling.

In the existing research, only the N-gram language model-based steganalysis proposed in Reference [4] is for sentence-level steganalysis tasks. From the experimental results in Reference [4], this method's accuracy on stego sentences was found to be 84.9%, while its accuracy for cover sentences emerged as 38.6%. Moreover, its *Precision*, *Recall* and *Accuracy* were 15.01%, 84.90%, and 42.04%, respectively. It is therefore evident that the proposed sentence-level CNN greatly outperforms the N-gram language model-based steganalysis.

Table 2. The results of the proposed sentence-level CNN for the sentence-level steganalysis task.

	Precision	Recall	Accuracy
sentence-level CNN with max-over-time pooling	73.19%	79.46%	82.25%
sentence-level CNN with average pooling	73.17%	79.46%	82.24%

3.3. Detection performance of the proposed steganalysis for text-level task

For convenience, the proposed steganalysis via two-level cascaded convolutional neural network is here abbreviated to TCNNS. The proposed TCNNS employing sentence-level CNN with max-over-time pooling is denoted as TCNNS_MAX, while the TCNNS using average pooling is denoted as TCNNS_AVG. Three steganalysis methods (namely NRF in Reference [6], PP in Reference [7] and WES in Reference [8]) are compared to our proposed TCNNS. In particular, for WES, the same language model will learn word embeddings with different qualities from different corpora. The qualities here have different properties according to the requirements of downstream tasks. For steganalysis tasks, more attention is paid to the perception of the latent linguistic regularities. In addition to the word embeddings learned from the Gutenberg corpus, WES also employed the pre-trained 300-dimensional word embeddings from the Google News dataset, which can be directly downloaded from the Internet[‡]. WES using the word embeddings from the Google News dataset is denoted as WES_GOOGLE, while that using the Gutenberg corpus is denoted as WES_Gutenberg.

[‡]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlS S21pQmM/edit

In order to accurately evaluate the reliability of the proposed linguistic steganalysis method based on two-level cascaded CNNs, we measure the detection results of the chosen methods in terms of *Accuracy*,, the definition of which is similar to Eq. 3.3. The results of the different steganalysis methods are shown in Table 3. As shown in Table 3, TCNNS_MAX and TCNNS_AVG have similar performance, and their average detection accuracy for different stego text sets are 97.93% and 97.90%, respectively. The experimental results also show that the proposed TCNNS performs best in terms of average detection accuracy. TCNNS_MAX improves the average accuracy of PP by 11.03%, NRF by 4.4%, WES_GOOGLE by 3.24%, and WES Gutenberg by 2.82%. Moreover, TCNNS can accurately detect stego texts with low embedding rates. When the embedding rate is 25%, the detection performance of PP and NRF is very poor, while WES also performs somewhat poorly; by contrast, TCNNS achieves a detection accuracy of over 94.80%. Significantly, the TCNNS's detection accuracy figuring for stego texts with various embedding rates are similar. This shows that TCNNS can achieve relatively stable performance regardless of the embedding rate of the stego texts.

In order to clearly compare the performance of different steganalysis methods, the detection accuracy results in Table 3 are illustrated in Figure 2. From the figure, it can be clearly seen that the curves of TCNNS_MAX and TCNNS_AVG almost coincide with each other, and are also the smoothest out of all compared steganalysis methods. Experimental results further show that the proposed TCNNS has the strongest generalization ability compared with PP, NRF and WES. In short, it is easy to see that TCNNS, with its improved detection accuracy, outperforms all other steganalysis methods.

Table 3. The detection accuracy results of different steganalysis methods.

Stego text	PP	NRF	WES_GOOGLE	WES_Gutenberg	TCNNS_MAX	TCNNS_AVG
Tlex25%	68.42%	81.45%	90.15%	90.48%	94.82%	94.98%
Tlex50%	87.50%	95.36%	97.73%	97.95%	98.56%	98.49%
Tlex75%	94.97%	98.26%	99.30%	99.48%	99.10%	99.02%
Tlex100%	96.71%	99.03%	99.67%	99.83%	99.22%	99.11%
average	86.90%	93.53%	94.69%	95.11%	97.93%	97.90%

4. Conclusion

In this paper, we propose a linguistic steganalysis method, based on two-level cascaded convolutional neural networks, which automatically learns the steganographic features from sentences and texts in order to classify stego and cover texts. Firstly, the sentence-level CNN is presented, which automatically extracts the steganographic features of all sentences with synonyms in a detected text. Then, the text-level CNN is employed to extract text-level features and distinguish between stego and cover texts. Experimental results demonstrate that although the proposed steganalysis method has a more expensive training process in terms of computational costs than previous methods, it greatly improves the reliability and generalizability of the steganalysis method. Moreover, the proposed sentence-level CNN can be used for sentence-level steganalysis tasks.

In future work, we will focus on improving the sentence-level CNN, so that it can extract more effective steganographic features and thereby promote the performance of both sentence-level and text-level steganalysis methods. Moreover, we will also try to locate [40] or extract [41] the embedded

information in the successfully detected stego texts, which are challenging tasks and also the primary goals of steganalysis .

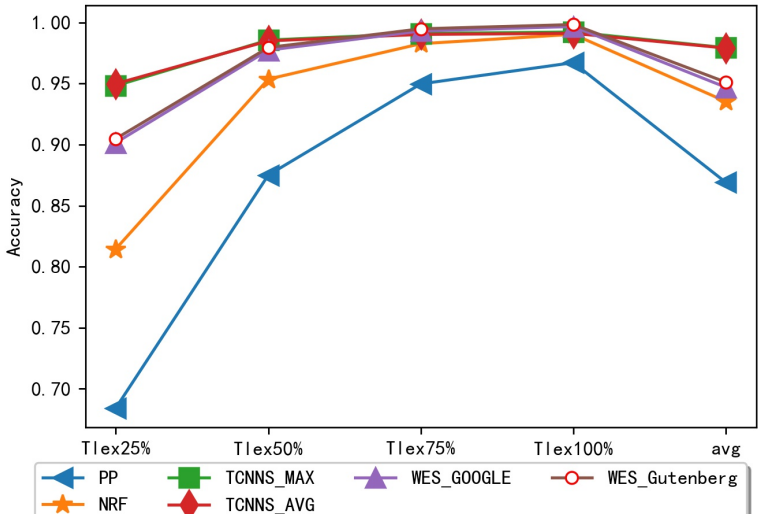

Figure 2. Detection performance comparison chart.

Acknowledgments

This project is supported by National Natural Science Foundation of China under grant 61972057 and U1836208, Hunan Provincial Natural Science Foundation of China under Grant 2019JJ50655, Scientific Research Fund of Hunan Provincial Education Department of China under Grant 18B160, Open Fund of Hunan Key Laboratory of Smart Roadway and Cooperative Vehicle Infrastructure Systems (Changsha University of Science and Technology) under Grant kfj180402, and ”Double First-class” International Cooperation and Development Scientific Research Project of Changsha University of Science and Technology (No. 2018IC25).

Conflict of interest

The author declares there is no conflict of interest.

References

1 Y. L. Liu, H. Peng and J. Wang, Verifiable diversity ranking search over encrypted outsourced data, *CMC-Comput. Mater. Con.*, **55** (2018), 37–57.

2 L. Y. Xiang, Y. Li, W. Hao, et al., Reversible natural language watermarking using synonym substitution and arithmetic coding, *CMC-Comput. Mater. Con.*, **55** (2018), 541–559.

3 H. M. Meral, B. Sankur, A. S. Ozsoy, et al., Natural language watermarking via morphosyntactic alterations, *Comput. Speech Lang.*, **23** (2009), 107–125.

4 C. M. Taskiran, M. Topkara and E. J. Delp, Attacks on lexical natural language steganography systems, *Proceed. SPIE*, **6072** (2006), 607209–607209-9.

5 Z. L. Chen, L. S. Huang, H. B. Miao, et al., Steganalysis against substitution-based linguistic steganography based on context clusters, *Comput. Electr. Eng.*, **37** (2011), 1071–1081.

6 Z. L. Chen, L. S. Huang and W. Yang, Detection of substitution-based linguistic steganography by relative frequency analysis, *Digit. Invest.*, **8** (2011), 68–77.

7 L. Y. Xiang, X. M. Sun, G. Luo, et al., Linguistic steganalysis using the features derived from synonym frequency, *Multimed. Tools Appl.*, **71** (2014), 1893–1911.

8 L. Y. Xiang, J. M. Yu, C. F. Yang, et al., A word-embedding-based steganalysis method for linguistic steganography via synonym-substitution, *IEEE Access*, **6** (2018), 64131–64141.

9 Z. S. Yu, L. S. Huang, Z. L. Chen, et al., Steganalysis of synonym-substitution based natural language watermarking, *Int. J. Mult. Ubiquit. Eng.*, **4** (2012), 21–34.

10 Z. S. Yu, L. S. Huang, Z. L. Chen, et al., Detection of synonym-substitution modified articles using context information, *Second International Conference on Future Generation Communication and Networking*, (2008), 134–139.

11 Y. T. Chen, J. Xiong, W. H. Xu, et al., A novel online incremental and decremental learning algorithm based on variable support vector machine, *Cluster Comput.*, Available from: https://doi.org/10.1007/s10586-018-1772-4.

12 L. Y. Xiang, G. H. Zhao, Q. Li, et al., TUMK-ELM: A fast unsupervised heterogeneous data learning approach, *IEEE Access*, **6** (2018), 35305–35315.

13 I. A. Bolshakov, A method of linguistic steganography based on collocationally-verified synonymy, *International Workshop on Information Hiding*, (2004), 180–191.

14 C. Y. Chang and S. Clark, Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method, *Comput. Linguist.*, **40** (2014), 403–448.

15 X. Yang, F. Li and L. Y. Xiang, Synonym substitution-based steganographic algorithm with matrix coding, *J. Chinese Comput. Syst.*, **36** (2015), 1296–1300.

16 H. H. Hu, X. Zuo, W. M. Zhang, et al., Adaptive text steganography by exploring statistical and linguistical distortion, *IEEE Second International Conference on Data Science in Cyberspace*, (2017), 145–150.

17 O. Russakovsky, J. Deng, H. Su, et al., Imagenet large scale visual recognition challenge, *Int. J. Comput. Vision*, **115** (2015), 211–252.

18 L. Y. Xiang, X. B. Shen, J. H. Qin, et al., Discrete multi-graph hashing for large-scale visual search, *Neural Process. Lett.*, **49** (2019), 1055–1069.

19 J. Wang, J. H. Qin, X. Y. Xiang, et al., CAPTCHA recognition based on deep convolutional neural network, *Math. Biosci. Eng.*, **16** (2019), 5851–5861.

20 N. Kalchbrenner, E. Grefenstette and P. Blunsom, A convolutional neural network for modelling sentences, preprint, arXiv:1404.2188.

21 D. J. Zeng, Y. Dai, F. Li, et al., Aspect based sentiment analysis by a linguistically regularized CNN with gated mechanism, *J. Intell. Fuzzy Syst.*, **36** (2019), 3971–3980.

22 R. H. Meng, S. G. Rice, J. Wang, et al., A fusion steganographic algorithm based on faster R-CNN, *CMC-Comput. Mater. Con.*, **55** (2018), 001–016.

23 S. Q. Tan and B. Li, Stacked convolutional auto-encoders for steganalysis of digital images, *Signal and Information Processing Association Summit and Conference*, (2014), 1–4.

24 J. Q. Ni, J. Ye and Y. Yang, Deep learning hierarchical representations for image steganalysis, *IEEE T. Inf. Foren. Sec.*, **12** (2017), 2545–2557.

25 Y. L. Qian, J. Dong, W. Wang, et al., Deep learning for steganalysis via convolutional neural networks, *Proceed. SPIE*, **9409** (2015), 94090J–94090J–10.

26 G. S. Xu, H. Z. Wu and Y. Q. Shi, Structural design of convolutional neural networks for steganalysis, *IEEE Signal Proc. Let.*, **23** (2016), 708–712.

27 J. S. Zeng, S. Q. Tan, B. Li, et al., Large-scale jpeg image steganalysis using hybrid deep learning framework, *IEEE T. Inf. Foren. Sec.*, **13** (2018), 1200–1214.

28 A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Proceedings of the 25th International Conference on Neural Information Processing Systems*, **1** (2012), 1097–1105.

29 Y. Kim, Convolutional neural networks for sentence classification, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, (2014), 1746–1751.

30 J. Turian, L. Ratinov and Y. Bengio, Word representations: A simple and general method for semi-supervised learning, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (2010), 384–394.

31 G. E. Hinton, Learning distributed representations of concepts, *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, **1** (1986), 12–23.

32 Y. Bengio, H. Schwenk, J. Sencal, et al., Neural probabilistic language models, *J. Mach. Learn. Res.*, **3** (2003), 1137–1155.

33 F. Morin and Y. Bengio, Hierarchical probabilistic neural network language model, *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, (2005), 246–252.

34 R. Collobert and J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, *Proceedings of the 25th International Conference on Machine Learning*, (2008), 160–167.

35 T. Mikolov, I. Sutskever, K. Chen, et al., Distributed representations of words and phrases and their compositionality, *International Conference on Neural Information Processing Systems*, (2013), 3111–3119.

36 B. Shen, C. W. Forstall, A. Rocha, et al., Practical text phylogeny for real-world settings, *IEEE Access*, **6** (2018), 41002–41012.

37 D. J. Zeng, Y. Dai, F. Li, et al., Adversarial learning for distant supervised relation extraction, *CMC-Comput. Mater. Con.*, **55** (2018), 121–136.

38 R. Collobert, J. Weston, L. Bottou, et al., Natural language processing (almost) from scratch, *J. Mach. Learn. Res.*, **12** (2011), 2493–2537.

39 Y. L. Boureau, N. L. Roux, F. Bach, et al., Ask the locals: Multi-way local pooling for image recognition, *2011 International Conference on Computer Vision*, (2011), 2651–2658.

40 C. F. Yang, F. L. Liu, S. K. Ge, et al., Locating secret messages based on quantitative steganalysis, *Math. Biosci. Eng.*, **16** (2019), 4908–4922.

41 C. F. Yang, X. Y. Luo, J. C. Lu, et al., Extracting hidden messages of MLSB steganography based on optimal stego subset, *Sci. China Inform. Sci.*, **61** (2018), 119103:1–119103:3.

AIMS Press