Linguistic Steganography Based on Adaptive Probability Distribution

Xuejing Zhou, Wanli Peng, Boya Yang, Juan Wen, Yiming Xue, *Member, IEEE*, and Ping Zhong

Abstract—Text has become one of the most extensively used digital media in Internet, which provides steganography an effective carrier to realize confidential message hiding. Nowadays, generation-based linguistic steganography has made a significant breakthrough due to the progress of deep learning. However, previous methods based on recurrent neural network have two deviations including exposure bias and embedding deviation, which seriously destroys the security of steganography. In this article, we propose a novel linguistic steganographic model based on adaptive probability distribution and generative adversarial network, which achieves the goal of hiding secret messages in the generated text while guaranteeing high security performance. First, the steganographic generator is trained by using generative adversarial network to effectively tackle the exposure bias, and then the candidate pool is obtained by a probability similarity function at each time step, which alleviates the embedding deviation through dynamically maintaining the diversity of probability distribution. Third, to further improve the security, a novel strategy that conducts information embedding during model training is put forward. We design various experiments from different aspects to verify the performance of the proposed model, including imperceptibility, statistical distribution, anti-steganalysis ability. demonstrate that our proposed model outperforms the current state-of-the-art steganographic schemes.

Index Terms—Linguistic steganography, natural language generation, adaptive probability distribution

1 Introduction

Steganography is the art and science of concealing secret information within carriers (image [1], [2], text [3], [4], [5], [6], [7], [8], [9], video [10], [11], etc.), mainly used for covert communication. On the contrary, steganalysis aims to reveal the presence of secret information by detecting whether there are abnormal artifacts left by data embedding [12], [13], [14]. They compete with each other, and both benefit and evolve from the competition.

As the most widely used information carrier in daily life, text is an ideal steganographic carrier with great value and practical significance. J. Fridrich [15] has summarized that, according to different embedding mechanisms, steganography can be divided into cover selection, cover modification, and cover synthesis. Cover selection methods selected different covers to transmit secret messages, which can ensure the cover is always "100percent natural", but an obvious drawback is an impractically low payload [16], [17], [18]. For the purpose of increasing the embedding capacity, format-based text steganography, which belongs to cover modification, was proposed. These kind of methods used the text format information to embed information [19], [20], [21], such as changing the line spacing [19], using the format redundancy of PDF document and webpage file [21], etc. Although these kind of methods have high payload and are easy to deceive the detection of human eyes, an obvious disadvantage is that they cannot resist the OCR-based attack and the statistic-based detection. [22]. Since then, in order to improve the security of steganography, modification-based text steganography was proposed, which embedded secret information by modifying and replacing the text content with different granularity, such as synonym substitution [22], [23], [24], [25] and syntactic change [26]. Because these methods directly use the content of text to embed information, they can resist OCR attack. However, they are still easy to be detected due to the statistical change of word frequency [27], [28]. In addition, because of their low embedding capacity, they lack practical value. Therefore, current text steganography mainly focuses on the generation-based steganography. This kind of method usually builds a language model and uses the language model to generate stego text directly. The stego text generated by the statistical language model can maintain the same statistical characteristics as the nature language, so it is hard to be detected by statistical-based steganalysis method. Besides, this method also has high embedding capacity.

In order to build the language model, Wayner [29] first proposed a mimic function to construct a Huffman tree to learn the statistical distribution of each character. Although this method adheres to the statistical law of characters, the generated text is hard to meet the grammar rules. Then, Chapman *et al*. [30] tried to use a syntactic template or syntax structure tree to generate texts. This method matches the grammar rules, but it is challenging to maintain semantic consistency and be easily detected by steganalysis based on statistical language model. Therefore, some researchers utilized the Markov chain to calculate the number of common occurrences of each phrase and obtained the transition

- *Xuejing Zhou, Wanli Peng, Boya Yang, Juan Wen, and Yiming Xue are with the Department of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China. E-mail: {18585080313, 17801066830}@163.com, { hunanpwl, wenjuan, xueym}@cau.edu.cn.*
- *Ping Zhong is with the Department of Science, China Agricultural University, Beijing 100083, China. E-mail: zping@cau.edu.cn.*

Manuscript received 14 Sept. 2020; revised 26 Mar. 2021; accepted 9 May 2021. Date of publication 13 May 2021; date of current version 2 Sept. 2022. (Corresponding Author: Yiming Xue.)
Digital Object Identifier no. 10.1109/TDSC.2021.3079957

probability. However, due to the two limitations of Markov hypothesis, the generated stego text is still unsatisfactory [9] and easily detected by rich features based steganalysis [31]. In recent years, many researchers employed long short-term memory (LSTM) to build the language model, which overcomes the limitations of the Markov hypothesis. Tina Fang *et al.* [8] utilized a 3-layer LSTM to build the language model, and divided the dictionary in advance and fixed the code for each word. Then in the generation process, the most appropriate word in the corresponding subset was selected as the output according to the secret bitstream. Yang *et al.* [9] built the same language model as Tina Fang, but in the embedding process, they dynamically coded each word according to its conditional probability distribution by two coding methods: fix-length coding (FLC) and variable-length coding (VLC). By fully considering the conditional probability distribution at each moment, the generated text is more natural and of higher quality.

However, with the development of steganalysis technology, more and more steganalysis models based on high-dimensional features were proposed, making it difficult for steganographic models to resist. For example, Wen [14] proposed a text steganalysis model based on convolutional neural network (CNN) for automatically learning text features. Niu [32] proposed a hybrid text steganalysis method based on bidirectional LSTM-CNN. Bao [33] proposed an attentional LSTM-CNN model to tackle the text steganalysis problem. The powerful supervised machine learning (ML) schemes remarkably improve the detection rate of steganalysis and pose a great challenge to steganography. Therefore, how to design a high secure steganographic model has become a key issue.

The anti-steganalysis ability of image steganography can be effectively improved with GAN (Generative Adversarial Networks) [34], [35]. The model training and information hiding are divided into two independent processes, which cause serious deviation and limit its performance. This motivates us to construct a text steganographic model based on GAN, and adopt a novel strategy to combine the model training with the information embedding, which makes the trained model more suitable for steganography and reduces the model deviation. Besides, the existing word choosing strategy for deep learning-based text steganography only considers the words with the highest probability, resulting in a significant decline in the diversity of the generated stego texts. The problem of pattern collapse will be aggravated if we apply this kind of technique to GAN. Thus this word choosing strategy is also not suitable for GAN-based linguistic steganograpy. Consequently, designing an algorithm that can effectively keep the diversity of the generated text is a vital issue for GAN-based linguistic steganography. Instead of using the rank of word probability to encode the secret bit, in this paper, we select the target word through pseudo-random probability sampling, obtain the similar word candidates through a word similarity function, and finally encode the words to complete the information embedding. On the one hand, the introduction of pseudo randomness dramatically alleviates the mode collapse caused by GAN and ensures the diversity of generated text. On the other hand, it makes the stego text distribution consistent with the natural text distribution to improve the quality of stego texts. Based on the probability adaptive embedding algorithm, LSTM is employed as the generator and the convolutional neural network as the discriminator. In the training process, the outcome of discriminator is used as the feedback signal to the generator, making the stego text generated by the generator as close to the normal text as possible, so that the discriminator can not distinguish it.

In this work, our contribution is threefold:

1) A generative adversarial network based linguistic steganographic model is proposed to fool the high-dimensional steganalyzers. We believe the introduction of intentional adversarial operations is a promising way to counter text steganalysis.
2) We propose a novel probability-based adaptive embedding algorithm. As opposed to conventional approaches simply used the largest transition probability to encode words, the adaptive embedding algorithm with a similarity function is capable of keeping the embedded distribution consistent with the real distribution.
3) A novel training strategy is proposed. Instead of dividing the training and steganography into two isolated stages, information hiding is fully considered in the training process, thus a more suitable model for information embedding is obtained, which reduces the embedding deviation and improves its performance.

The rest of this paper is organized as follows. Preliminaries and prior work are provided in Section 2. In Section 3, we will describe the architecture of our proposed framework, including the generator, the discriminator, the loss function, and the embedding algorithm. In Section 4, we will present the experimental setup and show the performance of the proposed steganography. The conclusions and prospects for future research are summarized in Section 5.

2 Preliminaries and Prior Work

2.1 Notation

In this article, capital letters in bold are used to represent matrices. The corresponding lowercase letters are used for matrix elements. The flourish letters are used for sets. The normal text with n words is denoted by $\mathbf{C} = \{c_1, c_2, \ldots, c_n\}$, where the signal c_i is the ith word in text. $\hat{\mathbf{C}} = \{\hat{c_1}, \hat{c_2}, \ldots, \hat{c_n}\}$ denotes the generated text without embedding secret message. $\mathbf{S} = \{s_1, s_2, \ldots, s_l\}$ denotes the stego text. Secret message is denoted as $\mathbf{M} = \{m_1, m_2, \ldots, m_k\}$, where $m_i \in \{0, 1\}$ is the ith secret bit in it. $P(Sentence) = P(w_1, w_2, \ldots, w_l)$ denotes the language model, which is the probability distribution of sentences composed of sequence of words. In the process of generating stego text, $P(w_t|w_1, w_2, \ldots, w_{t-1})$ is the transition probability of the word sequence in time step t. $\mathcal{CP}_t = \{cp_t^1, cp_t^2, \ldots, cp_t^k\}$ is the set of the first k words with the highest transition probability at time step t.

2.2 The Security of Linguistic Steganography

Security is the most critical foundation of covert communication. In general, to ensure the security of linguistic steganography, it is necessary to keep the stego texts analogous to the normal texts, which is mainly manifested in two

aspects: text quality and text statistical distribution. Text quality is reflected primarily in the semantic fluency, grammatical correctness, and word consistency. Only by generating high-quality steganographic text, can it avoid being detected by human eyes or steganalysis model based on language model. Besides, maintaining a normal text statistical distribution is also essential because many steganalysis algorithms are designed to detect the changes of text distribution. For example, the traditional steganalysis method detects stego text by manually designed statistical features, such as word frequency [26] and word position distribution [40]. And the neural network based steganalysis model automatically learns the implicit statistical features by minimizing the classification error rate of stego text and normal text. Therefore, to improve the anti-steganalysis performance, we should keep the stego text distribution resemblant to the natural text distribution.

2.3 Generation-Based Linguistic Steganography

Generation-based linguistic steganographic method usually has a high hidden capacity and is therefore considered as a promising research direction in the field of text steganography [5], [6], [7], [8], [9]. The development of generation-based linguistic steganography has gone through three stages: template-based steganography, Markov chain-based steganography, and neural network-based steganography. Although natural language steganography based on neural network has significantly improved the stego text quality, it's worthy to notice that the security of steganography is not only reflected in stego text quality, but also in the stego text statistical distribution. And with the development of steganalysis methods based on neural network, the existing steganographic methods are difficult to meet the security requirements. The main reason is that there are two deviations in them.

The first deviation is the so-called exposure bias caused by the LSTM model. This is due to the discrepancy between training and inference stage: the model sequentially generates the next word based on previously generated words during inference but it is trained to generate words given ground-truth words, as shown in Fig. 1. In the training phase, LSTM learns the language model through a hidden layer, which is denoted as:

$$h_t = f_{LSTM}(\hat{c}_t | c_1, c_2, \ldots, c_{t-1}). \tag{1}$$

Obviously, the state $\{c_1, c_2, \ldots, c_{t-1}\}$ of the hidden layer must exist in the corpus, regardless of its actual predicted output $\{\hat{c_1}, \hat{c_2}, \ldots, \hat{c_{t-1}}\}$. However, in the inference stage, the hidden layer is denoted as:

$$h_t = f_{LSTM}(s_t | s_1, s_2, \ldots, s_{t-1}), \tag{2}$$

where $\{s_1, s_2, \ldots, s_{t-1}\}$ may not exist in the training corpus due to the embedding operation. For the LSTM which is trained based on the maximum likelihood criterion, when it encounters word sequence that has never been seen before, it will significantly reduce the prediction performance and cause deviation. Moreover, as the sentence length increases, the deviation will accumulate.

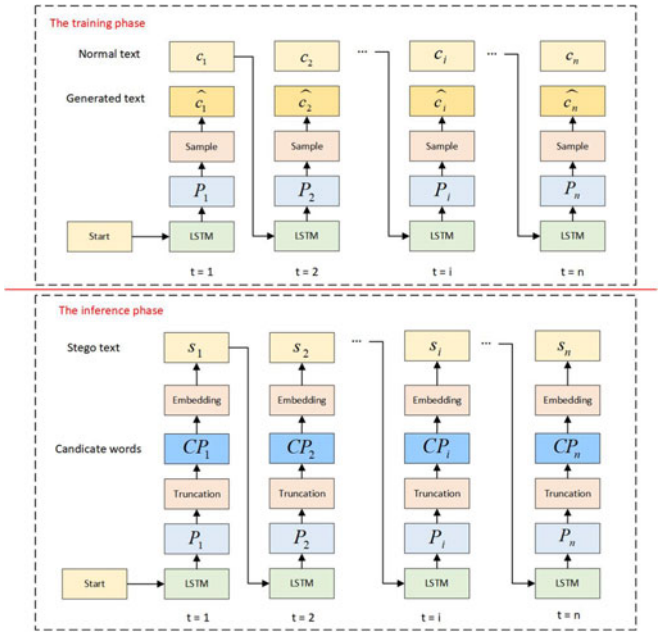


Fig. 1. Exposure bias in LSTM based on MLE.

The second deviation is the embedding deviation, which is caused by the embedding operation. In non-embedding cases, the predicted words are directly sampled based on the real probability distribution, which is defined as:

$$\begin{aligned} P_t &= f_o(W_o \cdot h_t + b_o) \\ &= \{p_{t1}, p_{t2}, \ldots, p_{t|V|}\} \end{aligned} \tag{3}$$

$$P'_t = \{p'_{t1}, p'_{t2}, \ldots, p'_{t|V|}\}, \tag{4}$$

where f_o is softmax function. W_o, b_o are learned weight matric and bias. $\{p_{t1}, p_{t2}, \ldots, p_{t|V|}\}$ is the probability of each word in the vocabulary at time step t. $|V|$ is the vocabulary size. P'_t is the probability distribution vector after ranking the probability values from large to small.

However, in the case of embedding, the actual sampling distribution will be significantly changed. For example, for the FLC which encodes the maximum probability words based on a perfect binary tree [9]. As long as the embedding rate is determined, the probability distribution of the word is determined. Assume the embedding rate is n bpw (bits per word), the actual sampling distribution is denoted as:

$$\hat{p'_{ti}} = \begin{cases} \frac{1}{2^n}, & if \quad i \leq 2^n. \\ 0, & if \quad i > 2^n. \end{cases} \tag{5}$$

It means that the actual sampling probability of the candidate words is equal regardless of the real theoretical sampling distribution. And there is a contradiction. When n is small, the number of words with probability is set to 0 increases, which will reduce the diversity of the generated text. When n is large, it is very likely to sample words with poor quality, resulting in a significant decline in the quality of the generated text. Both cases will reduce the security of the steganography.

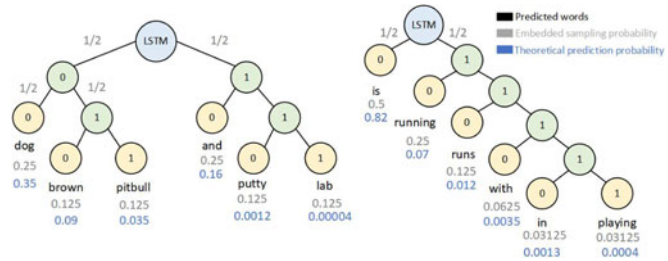


Fig. 2. The deviation of variable-length coding.

Although VLC [9], which encodes the maximum probability words based on a Huffman tree, considers the transition probability in a more reasonable way (the words with high probability have high sampling probability), the actual sampling distribution is still quite different from the real theoretical distribution. For example, as shown in Fig. 2, with different theoretical probability distribution, the construction of Huffman tree will be different, which will lead to a slight difference in the sampling probability of each word. However, the sampling probability of the word with the largest theoretical transition probability will not be greater than $1/2$, while the word with the smaller probability is assigned a larger sampling probability. And for the remaining $|V| - 2^n$ words (n is the embedding rate), their probability is still 0. Generally, there are two problems in both FLC and VLC: 1) only a tiny number of words are considered, resulting in a significant reduction in text diversity; 2) the actual sampling distribution severely deviates from the real sampling distribution, which leads to the deviation between the stego text distribution and normal text distribution, and the security is reduced.

Quite recently, Yang [36] presented a preliminary attempt to solve the problem of LSTM, while the above two deviations have not been addressed. In this paper, we conduct a linguistic steganographic model based on generative adversarial networks with a probability-based adaptive embedding algorithm to alleviate the above two deviations. First of all, in order to solve the deviation of exposure bias, we build a steganographic model based on the generative adversarial network, which generates the next word based on the previously generated word in the training process, coinciding with the inference process. When the whole stego text generation is done, the generated text samples are fed to the discriminator, which is a classifier trained to distinguish the real and generated stego text samples, to get reward signals for updating the generator. Second, in order to solve the deviation caused by the embedding operation, we propose a probability-based adaptive embedding algorithm. We do not directly use the transition probability to encode the words. Still, first, like the normal text generation, we obtain the target words by random sampling according to the probability, and obtain the candidate embedding words through the probabilistic similarity function, and then determine the number of secret information that can be embedded. In this way, it is greatly guaranteed that the probability distribution of the actual sampling is close to the real theoretical probability distribution.

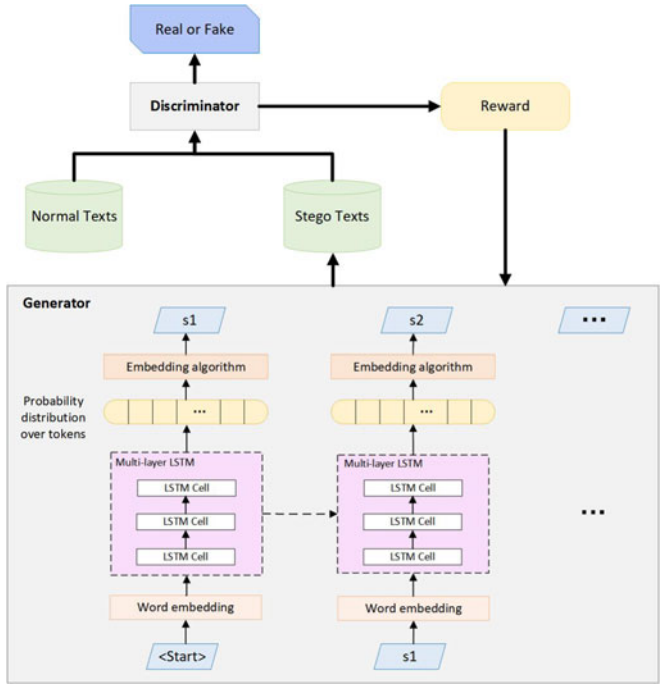


Fig. 3. The overall architecture of the proposed method.

3 The Proposed Method

In this section, we first demonstrate the overall architecture of the proposed method based on GANs, which incorporates a generator based on LSTM and a discriminator based on CNN. Second, the details of the generator are given. After that, we explain the design considerations for the discriminator. Then, we introduce the definition of the loss functions and the learning strategy. Lastly, the adaptive embedding algorithm with the similarity function is described.

3.1 Overall Architecture

The proposed linguistic steganographic architecture mainly contains two modules: a generator and a discriminator, as shown in Fig. 3. We utilize a multi-layer LSTM as generator due to its powerful modeling ability for sequential signals. After obtaining the word probability distribution of each time step through the multi-layer LSTM network, we use the embedding algorithm to generate the stego word, and take it as the output of the current time and the input of the next time. After the generator generates a large number of stego texts during the training process, we utilize the stego texts and the normal texts to train the discriminator. We use an LS-CNN based discriminator due to the effectiveness of LS-CNN for linguistic steganalysis [14]. After the discriminator D is trained for a pre-set number of steps, to avoid data leakage, the generator generates another batch of stego texts, which are fed to D to obtain the probability to reveal how close they are to the real text. Next, the probability will be used as the reward value to guide the generator to continue training. In this way, the generator and the discriminator are alternately trained so that they can make progress together in the confrontation. The training steps are listed in Algorithm 1.

Algorithm 1. Training Steps of the Proposed Model

Input:
C: cover texts;
R: random matrix with the normal distribution with expectation 0 and variance 0.1.
Output:
Trained generator model
1: Initialize generator G and discriminator D with random weights R.
2: Pre-train G using Maximum Likelihood Estimation on C.
3: Generate negative samples S using G for training D.
4: Pre-train D via minimizing the cross entropy on C and S.
5: **repeat**
6: **for** g-stegps **do**
7: Generate stego texts S using G.
8: Calculate reward value for each word in S using D.
9: Update generator parameters via policy gradient.
10: **end for**
11: **for** d-stegps **do**
12: Use current G to generate negative examples S and combine with given positive examples C.
13: Train discriminator D for k epochs.
14: **end for**
15: **until** model converges

3.2 The Design of the Generator

In the process of automatic text generation, we mainly take advantage of the LSTM's powerful ability in feature extraction and expression for sequential signals. As shown in Fig. 3, the first layer of our neural network is a word embedding layer and it maps each word s_i to a dense semantic space with a dimension of d, that is $s_i \in R^d$.

$$s_i = [e_{i1}, e_{i2}, \ldots, e_{id}]. \tag{6}$$

In general, a recurrent neural network consists of multiple network layers, each with multiple LSTM units. And the more layers of neural network in space, the stronger the ability to extract and express features. Thus, we use n_j to indicate the number of LSTM units in jth hidden layer U_j, so the units of jth layer can be represented as

$$U_j = \{u_1^j, u_2^j, \ldots, u_{n_j}^j\}. \tag{7}$$

For the first hidden layer, the input of each unit u_i^1 at time step t is the weighted sum of the elements in word s_t. s_0 is the start token, otherwise it means the output word of the previous time. Thus, at time t, u_i^1 can be mathematical as:

$$u_i^1 = \sum_{m=1}^{d} w_{im}^1 \cdot e_{tm} + b_i^1 \tag{8}$$

where e_{tm} is the mth element in the word embedding of word s_t. w_{im}^1 and b_i^1 are learned weights and biases of the first layer. Then the output value of u_i^1 at time step t is

$$o_i^1 = f_{LSTM}(u_i^1) = f_{LSTM}\left(\sum_{m=1}^{d} w_{im}^1 \cdot e_{tm} + b_i^1\right), \tag{9}$$

where $f_{LSTM}(\cdot)$ denotes the output O_t of the LSTM unit, which can be obtained using the following formulas:

$$\begin{cases} I_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\ F_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ C_t = F_t \cdot C_{t-1} + I_t \cdot tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \\ O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\ h_t = O_t \cdot tanh(C_t) \end{cases} \tag{10}$$

where I_t indicates the input gate, F_t is the forget gate and O_t is the output gate, C_t is the memory cell, h_t is the hidden state, and $W_i, W_f, W_o, b_i, b_f, b_o$ are the learned weights and biases.

We can use a vector Out^j to represent the output of the jth hidden layer at time step t, and each element in Out^j indicates the output value of each unit in the jth hidden layer at time step t, that is

$$Out^j = [o_1^j, o_2^j, \ldots, o_{n_j}^j]. \tag{11}$$

Then the input of each unit u_i^l in the lth hidden layer at time step t is the weighted sum of the output values of the units in the previous layer, that is

$$u_i^l = Out^{l-1} \cdot W^l = \sum_{m=1}^{n_{l-1}} w_{im}^l \cdot o_m^{l-1} + b_i^l. \tag{12}$$

And the ith output unit of the lth layer at time step t is

$$o_i^l = f_{LSTM}(u_i^l) = f_{LSTM}\left(\sum_{m=1}^{n_{l-1}} w_{im}^l \cdot o_m^{l-1} + b_i^l\right). \tag{13}$$

Then we use this learned output units of last hidden layer to calculate the score for each word in the vobabulary V, that is

$$y_i = \sum_{m=1}^{n_{p-1}} w_{im}^p \cdot o_m^{p-1} + b_i^p, \tag{14}$$

where w_{im}^p and b_i^p are the learned weight and bias of the last layer p.

Thereafter, we use the softmax function to calculate the predictive probability of each word s_i in vocabulary V, which can be written as:

$$p(s_i) = softmax(y_i) = \frac{exp(y_i)}{\sum_{j=1}^{|V|} exp(y_j)}, \tag{15}$$

where $|V|$ is the number of words in dictionary.

Finally, the stego words will be generated by embedding algorithm based on the probability score $P = \{p_{s_1}, p_{s_2}, \ldots, p_{s_{|V|}}\}$, which will be introduced in section E.

3.3 The Design of the Discriminator

Depth discrimination models such as deep neural network (DNN), CNN, and RNN have shown good performance in complex sequence classification and text steganalysis [14], [32], [33], [47]. In this paper, we choose CNN as our discriminator, because CNN can effectively extract the features of different level of granularity in text, and has shown great effectiveness in text steganalysis.

For each input sentence S, we can illustrate it with a matrix $D \in R^{l\times d}$, where the ith row indicates the ith word in passage S, each word is represented as a d-dimension vector. Generally, let $X_{i:j}$ refer to the matrix which consists of the word vectors of the ith word to the jth word. The width of each convolution kernel is the same as the width of the input matrix. If the height of the convolution kernel is h, and the convolutional kernel is expressed as $W \in R^{h\times K}$, then the feature c_i extracted from $X_{i:j}$ by the convolutional kernel can be written as:

$$c_i = f(W \cdot X_{i:j} + b_i), \tag{16}$$

where f is a non-linear function and b_i is the bias. The convolutional kernel slides from the top to the bottom of the text matrix, which produces a feature map named as c:

$$c = [c_1, c_2, \ldots, c_{N-h+1}]. \tag{17}$$

We then apply a maximum pooling layer, which takes the maximum value as $\widetilde{c} = max\{c\}$. The process described above is a process in which one convolution kernel produces one feature. In order to better capture the multi-granularity features of text, we use several convolution kernels of different sizes. The generation of each feature is described above. Let the feature vector produced at the last layer be:

$$z = [\widetilde{c_1}, \widetilde{c_2}, \ldots, \widetilde{c_m}]. \tag{18}$$

Finally, we can get the predictive probability of a normal text by

$$p = Softmax(W \cdot z + b). \tag{19}$$

3.4 The Learning Strategy

As we all know, GAN is designed to generate real-valued continuous data, so it is challenging to generate discrete token sequences (such as text) directly. This is because, for real-valued continuous data, the loss gradient of D is used to guide the generator G (parameter) to slightly change the output value to make it more realistic. But if the generated data is based on discrete tokens, the slight change guidance from the discriminator makes little sense because there is probably no corresponding token for such a slight change in the limited dictionary space. Thus, we use the policy gradient in reinforcement learning to update the model parameters [41]. The objective function is defined as:

$$J(\theta) = \sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0) \cdot Q^{G_\theta}_{D_\phi}(s_0, y_1), \tag{20}$$

where $G_\theta(y_1|s_0)$ is the generator (policy) which is used to generate a sequenec from a start token s_0. $Q^{G_\theta}_{D_\phi}(s_0, y_1)$ is the action-value function of a sequence, that is the expected accumulative reward starting from state s, taking action a, and following policy G_θ. The goal of the generator is to maximize the objective function.

Then, we use the estimated probability of being real by the discriminator as the reward. Formally, we have:

$$Q^{G_\theta}_{D_\phi}(y_T, s_0) = D_\phi(Y_{1:T}). \tag{21}$$

However, the discriminator only provides a reward value for a finished sequence. Since we actually care about the long-term reward, at every timestep, we apply Monte Carlo search with roll-out policy G_θ to sample the unknown last $T-t$ tokens according to the previous tokens. We represent an N-time Monte Carlo search as

$$\{Y^1_{1:T}, \ldots, Y^N_{1:T}\} = MC^{G_\theta}(Y_{1:t;N}). \tag{22}$$

To reduce the variance and get more an accurate assessment of the action value, we run the roll-out policy starting from the current state till the end of the sequence for N times to get a batch of output samples. Thus, we have:

$$Q^{G_\theta}_{D_\phi}(s = Y_{1:t-1}, a = y_t) = \tag{23}$$

$$\begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=1}^{n} D_\phi(Y^N_{1:T}), \quad Y^N_{1:T} \in MC^G_\theta(Y_{1:t;n}) \;\; for \quad t < T \\ D_\phi(Y_{1:T}), \qquad\qquad\qquad\qquad\qquad\qquad for \quad t = T. \end{cases}.$$

Then, the gradient of the objective function $J(\theta)$ can be derived as:

$$\bigtriangledown_\theta = \sum_{t=1}^{T} E_{Y_{1:t-1}\sim G_\theta}\left[\sum_{y\in\mathcal{Y}} \bigtriangledown_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q^{G_\theta}_{D_\phi}(Y_{1:t-1}, y_t)\right], \tag{24}$$

and the generator's parameters can be updated as:

$$\theta \leftarrow \theta + \alpha \bigtriangledown_\theta J(\theta). \tag{25}$$

3.5 Probability Adaptive Embedding Algorithm

For the data embedding module, we propose a probability-based adaptive embedding algorithm illustrated in Fig. 4. After obtaining the transition probability $p(w_t|w_1, w_2, \ldots, w_{t-1})$ of each moment through the multi-layer LSTM network, we no longer simply encode the 2^n words with the largest probability according to the embedding rate n, but first obtain the target word and its probability by pseudo-random sampling according to the probability. Then we obtain similar words to form the candidate pool through the probability similarity function. Finally, the words in the candidate pool are encoded, and the words corresponding to the binary bitstream of secret information are selected as the output. Our thought is mainly based on the following facts.

First of all, if only the words with the largest probability are encoded, The low-frequency words will appear fewer, and even none of them would be chosen because of the embedding strategy. In contrast, the high-frequency words will appear more than expectations. And the word frequency distribution will be destroyed, which does not meet the Zipf's law and is easily detected by steganalysis tools. Thus, we use the pseudo-random function to sample the target word and obtain its probability according to the probability distribution. Assume the word probability distribution at time step t, which is obtained from the LSTM, is denoted as P_t. And the target word and its probability sampled by a pseudo-random function can be represented as:

$$w_{target}, p_{target} = f_{pr}(P_t, key) \tag{26}$$

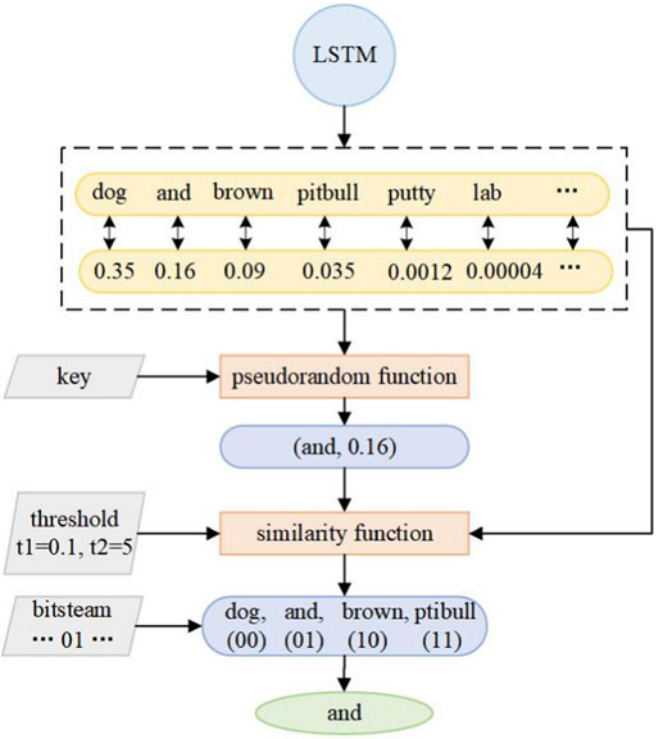


Fig. 4. Illustration of the process of the proposed adaptive embedding.

where f_{pr} indicates the pseudo-random function, and key is a random seed. In this way, high-frequency words have a larger sampling probability, while low-frequency words have a smaller sampling probability, making the distribution of the generated stego text close to the real text distribution and has higher security. In addition, the introduction of the key key further improves the security of the model and meets the Kerckhoffs' principle.

Second, although there is always more than one feasible solution at each time step due to the diversity of the text, it is obvious that the number of replaceable words is different at each time because of the limitation of lexical and syntactic. Therefore, it is unreasonable to encode a fixed number of words at each time step according to the embedding rate. Consequently, we propose an adaptive embedding algorithm based on a similarity function. Specifically, we select the words similar to the target word probability according to the similarity function (SF) to build the candidate pooling. And the similarity function is defined as:

$$SF(w_i) = \begin{cases} True, & if \quad T_1 \le p_r \le T_2, p_{target} \ge T_{mid} \\ & or \quad T_3 \le p_r \le T_4, p_{target} < T_{mid}., \\ False, & else. \end{cases} \tag{27}$$

where $p_r = \frac{p_{target}}{P_{w_i}}$, and p_{w_i} is the probability of ith word in vocabulary, which is calculated by LSTM. T_{mid} is an empirical value selected by experiment. When p_{target} is greater than T_{mid}, select a smaller threshold range$(T1, T2)$; when p_{target} is smaller than T_{mid}, select a larger threshold range $(T3, T4)$. This is because when the probability of target words is high, a small threshold range can reduce the possibility of selecting words with low probability. In the case of low probability of target words, using a wide range of threshold can effectively improve the embedding capacity and make high probability words easier to be selected, so as to improve the text quality.

When $SF(w_i)$ is true, the probability of the candidate word w_i and target word p_{target} is considered to be similar. Through the similarity function, we can get the candidate pool (CP), which can be written as:

$$CP = [w'_{t,1}, w'_{t,2}, \ldots, w'_{t,n}], \tag{28}$$

where $w'_{t,i}$ is the ith replaceable words at time step t, which are the words with similar probability to the target word w_{target} screened from all words in the dictionary, through the similarity function. n is denoted the number of replaceable words at time step t. When the candidate pool is constructed, it is encoded by the perfect binary tree, and the corresponding word is selected as the stego output word according to the secret information bitstream that needs to be embedded. And the embedding rate at the current time is determined to be $\lfloor log_2 n \rfloor$.

Algorithm details of the proposed information hiding method are shown in Algorithm 2. With this method, we can generate a large number of high secure and smooth natural sentences according to the secret bitstream. Then these generated texts can be sent out through the open channel to achieve the purpose of hiding and sending confidential information with a high concealment.

Algorithm 2. Information Hiding Algorithm

Input:
Secret bitstream: $B = \{0, 0, 1, 0, 1, \ldots, 0, 1, 0\}$
Embedding threshold T_1, T_2
Seed of the similarity function: key
Output:
Multiple generated steganographic text: $Text = \{S_1, S_2, \ldots, S_N\}$

```
Data preparation and steganographic model training.
while not the end of B do
  if not the end of current sentence then
    Calculate the probability distribution of the next word according to the previously generated words using well trained generator model;
    Sample the target word by pseudo-random function f_pr(P_t, key) based on probability and key;
    Obtain the candidate words through the similarity function by Eq. (27) to compose the candidate pool (CP);
    Construct a perfect binary tree according to the probability distribution of each word in the CP and encode the tree;
    Read the secret binary stream, and search from the root of the tree according to the encoding rules until the corresponding leaf node is found and outputs its corresponding word;
  else
    Input the start token < start > as the start of the next sentence;
  end if
end while
return Generated sentences
```

Information extraction and embedding are two opposite operations. After receiving the transmitted sentence, the

receiver needs to decode the confidential information contained therein correctly. The process of information extraction and embedding is basically the same. We also need to use the same LSTM network to calculate the conditional probability distribution of each word at each time, and use the same seed to sample the same target words as the embedding process. Then construct the same candidate pool through similarity function, and use the same encoding method to encode the words in the candidate pool. Finally, according to the actual transmitted word at the current moment, the path of the corresponding leaf node to the root node is determined so that we can successfully and accurately decode the bits embedded in the current word. In this way, the bitstream embedded in the original texts can be extracted very quickly and without errors. The detailed process of extraction algorithm is shown in Algorithm 3.

Algorithm 3. Information Extraction Algorithm

Input:
Stego text: $Text = \{S_1, S_2, \ldots, S_N\}$
Embedding threshold T_1, T_2
Seed of the similarity function: key
Output:
Secret bitstream: $B = \{0, 0, 1, 0, 1, \ldots, 0, 1, 0\}$

```
Enter the start token < start > into the trained generator model;
for each word in sentence S do
  if not the end of current sentence then
    Calculate the probability distribution of the next word according to the previously words using generator model;
    Sample the target word by random sampling based on probability and key by Eq. (26);
    Obtain the candidate words through the similarity function;
    Using binary tree to encode words in the candidate pool;
    if Word_i in candidate pool then
  Based on the actual accepted word Word_i at each moment, determine the path from the root node to the leaf node;
  According to the tree coding rule, ie, the left side of the child node is 0 and the right side is 1, extract the corresponding bitstream and append to B;
    else
      The information extraction process ends;
    end if
  else
    Input the start token < start > as the start of the next sentence;
  end if
end for
return Extracted secret bitstream B;
```

4 Experimental Result and Analysis

In this section, we conduct several experiments to evaluate the performance of steganographic model in terms of imperceptibility, statistical distribution, and anti-steganalysis ability. The imperceptibility is mainly reflected in the quality of the generated stego text. For statistical distribution, we use the Earth Mover's Distance to analyze the distribution similarity between stego texts and normal texts. In addition, we use a variety of steganalysis models to test the anti-steganalysis ability of the proposed scheme.

TABLE 1
The Details of the Training Corpora

Dataset	Twitter [42]	MSCOCO [43]	IMDB [44]
Average Length	6.73	10.43	14.56
Words Number	673214	1042700	2184111

In Section 4.1, we introduce the experimental setup and the corpora for training. The evaluation metrics are presented in Section 4.2. Then, in Section 4.3, we perform some experiments to show our model performance under different threshold settings (values and symmetry) and compare it with other steganographic algorithms to demonstrate the effectiveness of the proposed algorithm. The effect of the generative adversarial network on the quality and anti-steganalysis ability of the generated stego texts is shown in Section 4.4. In Section 4.5, we compare the proposed model with other steganographic models in terms of text quality, statistical distribution, and anti-steganalysis ability.

4.1 Experimental Setup

We conducted experiments based on three kinds of corpora with different complexity, length, and type, including Twitter [42], Microsoft Coco [43], and Movie Review [44], to demonstrate the imitation ability of the proposed model for human writing. For Twitter, we chose the sentiment140 dataset published by Alec Go *et al.* [42]. It contains 1,600,000 tweets extracted by the Twitter API. The sentences in twitter corpus are generally short (usually less than 10 words), and have a high degree of freedom without strict grammar rules. Whereas Microsoft Coco (MSCOCO) is a large-scale dataset for object detection, segmentation, and captioning released by Microsoft Corporation. We chose the part of the dataset used for image caption as our corpus, which contains 112,887 sentences. Most of these descriptive sentences are of medium length (about 10 words). And the sentence structure is simple, mainly describing the characters, objects, and scenery. In addition, we chose a movie review dataset [44], namely IMDB dataset published by Maas *et al.* This corpus mostly has long sentences (about 15 words), and the text is relatively diversiform, involving a variety of subjects. We randomly selected 100,000 sentences from Twitter and MSCOCO, 150,000 sentences from IMDB corpus to form multiple training sets. The details of the preprocessed training sets are shown in Table 1.

We used two kinds of steganographic schemes, one is based on the Markov model, and the other is based on the LSTM model, as baselines to reflect the performance of the proposed steganographic model. The Markov-chain based steganographic method was proposed by Dai *et al.* [6]. For LSTM-based methods, we chose the model proposed by Tina Fang [8] and the model proposed by Yang *et al.* [9]. Since both the Tina Fang's model and our proposed model are trained by using the whole sentence in each iteration, and do not constrain the probability of the first word. For fair comparison, we re-implement the Yang's model with

the same approach instead of the training strategy based on splitting sentence with fixed length and selecting the top 100 words with the highest probability as the first input.

Our experiments were implemented based on Tensorflow with Python interface. To accelerate the training process, we use the GeForce GTX 2080Ti GPU and CUDA 10.0. The details of the hyper-parameters are as follows. We set the dimension of word vector to 300. The generator utilizes a 3-layer LSTM network, each with 800 hidden units. The filter sizes of the discriminator were set to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15 and 20, corresponding to the number of 100, 200, 200, 200, 200, 100, 100, 100, 100, 100, 160 and 160, respectively. The learning rates of generator and discriminator were initially set to 0.001 and 0.0001, respectively. We chose Adam [45] as the optimization method. Moreover, the dropout mechanism was used to prevent overfitting [46].

4.2 Evaluation Metrics

The purpose of steganography is to hide the existence of information in the carrier to ensure the security of important information. Therefore, security and embedding capacity are the primary evaluation criteria of the concealment system. Embedding capacity is usually measured by embedding rate (ER), which is defined as the average number of secret bits that can be embedded in each word (bpw), formalized as:

$$ER = \frac{1}{N}\sum_{i=1}^{N}\frac{K_i}{L_i}, \tag{29}$$

where N is the number of generated sentences, K_i is the number of bits embedded in the ith sentence, and L_i is the length of the ith sentence.

Security is mainly reflected in three aspects: the generated stego text quality, text statistical distribution characteristics, and anti-steganalysis ability. We use perplexity (ppl), which is a standard metric for sentence quality testing in Natural Language Processing, as evaluation criterion for the quality of texts. Perplexity is defined as:

$$perplexity = 2^{-\frac{1}{N}\sum_{i=1}^{N} logp_i(w_{i1},w_{i2},\ldots,w_{in})}, \tag{30}$$

where N is the number of generated sentences, words sequences $\{w_{i1}, w_{i2}, \ldots, w_{in}\}$ indicate the ith sentence, and $p_i(\cdot)$ is the probability over words in the ith sentence. The smaller the value is, the higher probability of generating texts with coherent meaning and correct grammar is, i.e., the higher text quality is reached.

To measure the similarity between normal text distribution and stego text distribution, we use the Wasserstein Distance, known as Earth Mover's Distance (EMD). The Earth Mover's Distance is written as follows:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}d_{i,j}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}}, \tag{31}$$

where P is the distribution of normal texts and Q is the distribution of stego texts. m, n are the numbers of samples of normal texts distribution and stego texts distribution, respectively. We set $m = n = 1000$. $d_{i,j}$ is the distance between p_i and q_j calculated by euclidean distance, where p_i is the ith example of the normal text and q_j is the jth example of the stego text. $f_{i,j}$ is regarded as the weight of connection between p_i and q_j, which is set to 1.

To test the anti-steganalysis ability, we select three neural network based steganalysis models, including LS-CNN [14], TS-RNN [47], and R-BI-C [32]. We use the accuracy, and F1 score as the evaluation criteria, which are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{32}$$

$$F1 = \frac{2TP}{2TP + FP + FN}, \tag{33}$$

where TP is true positives, TN is true negatives, FP is false positive and FN is false negative, and we assume the stego texts are positive samples.

4.3 Investigation on Embedding Algorithm

Unlike other embedding algorithms, which directly truncate the words with the highest probability for embedding, the core idea of our proposed steganography algorithm is to select words with similar probability to the target word to construct the candidate pool. Therefore, how to select reasonable thresholds has become a crucial issue. We use different threshold schemes, including Symmetric Single Threshold (SST), Asymmetric Single Threshold (AST), and Segmented Threshold (SGT), to test the performance of the proposed embedding algorithm. Symmetric single threshold treats words with higher and lower probability than target words equally, that is $\frac{1}{T} < \frac{p_{target}}{p_{candidate}} < T$. Using SST means that the probability of the candidate word should not be too large or too small compared to the probability of the target word, and the two thresholds are reciprocal. The AST method adopts the idea of SST, but increases the upper limit of candidate words probability by expanding the threshold on one side, that is $\frac{1}{kT} < \frac{p_{target}}{p_{candidate}} < T$, and k is a variable parameter greater than 1. With AST, the candidate words with a greater probability than the target word, are more likely to be selected. Using SGT means that when the probability of target words is greater than a certain threshold T_{mid}, we use a smaller threshold range $\frac{1}{k_1 T_1} < \frac{p_{target}}{p_{candidate}} < T_1$, otherwise, we use a larger threshold range $\frac{1}{k_2 T_2} < \frac{p_{target}}{p_{candidate}} < T_2$.

Using SGT means that when the probability of target words is greater than a certain threshold T_{mid}, we use a smaller threshold range, otherwise, we use a larger threshold range.

From Table 2, we can get the following conclusions:

1) With the expansion of threshold range, the embedding rate and perplexity value increase. This is because as the threshold range increases, the number of candidate words increases, which leads to the embedding capacity increases. But at the same time, the disturbance brought by embedding is also increased, which results in a decrease in the generated stego text quality.

TABLE 2
Comparison of Symmetric Threshold and Asymmetric Threshold on Twitter Corpus

bpw	Method	Threshold	ppl
5*1	SST	T = 2	49.10
	AST	T = 1, k = 2	42.10
	3*SGT	T_{mid} = 0.7	3***28.12**
		T_1 = 1, k_1 = 2	
		T_2 = 2/3, k_2 = 150	
5*2	SST	T = 3	53.35
	AST	T = 2.5, k = 2	46.22
	3*SGT	T_{mid} = 0.7	3***40.08**
		T_1 = 100/9, k_1 = 0.18	
		T_2 = 100/49, k_2 = 49	
5*3	SST	T = 10	85.69
	AST	T = 10, k = 1	82.05
	3*SGT	T_{mid} = 0.7	3***67.93**
		T_1 = 125/6, k_1 = 0.096	
		T_2 = 20/3, k_2 = 15	

2) The performance of AST is better than SST. This is because, with a certain embedding capacity, expanding the selection range of candidate words with higher probability will also reduce the number of selected low-probability words, making the model more inclined to choose high probability words to ensure the generated stego text quality.
3) The performance of SGT is better than that of AST. This is because segmentation thresholds better fit the real probability distribution. When the probability of target words is large, a small threshold range is adopted to reduce the possibility of selecting words with low probability. When the probability of target words is small, using a broad range of asymmetric threshold can effectively improve the embedding capacity, and make the high probability words more likely to be selected, thus improve the text quality.

In order to verify the effectiveness of the proposed embedding algorithm, we compare the several evaluated metrics of the 1,000 steganographic sentences generated by FLC, VLC and ours on the well-trained GAN, including word diversity, text quality, and anti-steganalysis ability. Since the embedding rate (bpw) in VLC is uncertain, we calculated the average number of bits hidden in each word of the generated text, which is listed on the second column as bpw (VLC). As can be seen in Tables 3, 4, and 5, we can draw the following conclusions:

1) The word diversity of our proposed algorithm is richer than those of FLC and VLC algorithms. This is because the proposed algorithm first leverages the random sampling to maintain the randomicity of the embedding sampling probability, and then the algorithm is dynamically embedded based on words with similar sampling probability, so as to avoid allocating large embedded sampling probability to some words with small probability.
2) The proposed method is dramatically superior to the FLC and VLC algorithms in terms of text quality and the anti-steganalysis ability. This is because the proposed method effectively alleviates the embedding deviation and maintains the consistency of probability distribution between the stego texts and cover texts.

TABLE 3
The Word Diversity of the Steganographic Texts Generated by Different Embedding Algorithms on Twitter Corpus

2*bpw	2*bpw(VLC)	the number of unique words		
		FLC [9]	VLC [9]	ours
1	1.00	1733	1438	**2215**
2	1.81	2117	1925	**2545**
3	2.92	2357	2063	**2688**

TABLE 4
The Text Quality of the Steganographic Texts Generated by Different Embedding Algorithms on Twitter Corpus

2*bpw	2*bpw(VLC)	text quality		
		FLC [9]	VLC [9]	ours
1	1.00	34.85	42.86	**28.12**
2	1.81	51.12	51.19	**40.80**
3	2.92	85.33	68.38	**67.93**

4.4 The Investigation of the GAN-Based Language Model

In this part, we conduct several ablation studies to investigate the architecture of GAN-based language model, including the generator with different number of LSTM layers and the discriminator with different types of kernel sizes. We compare the security performance of these different GAN-based language models in terms of text quality, EMD and anti-steganalysis ability.

4.4.1 The Investigation of Generator

We first fix the parameters and architecture of the discriminator, and then change the number of LSTM layers of the generator. Specifically, there are three different generator architectures equipped with 1 LSTM layer, 2 LSTM layers, and 3 LSTM layers, respectively. The experimental results are shown in Table 6. It is can be observed that, with the increase of the number of LSTM layers, the consistency of probability distribution is gradually enhanced, meanwhile the anti-steganalysis ability is increasingly improved. These results demonstrate that the design of the generator with 3 LSTM layers can achieve better security performance than

TABLE 5
The Anti-Steganalysis Ability of the Different Embedding Algorithms on Twitter Corpus

2*bpw	2*bpw(VLC)	anti-steganalysis		
		FLC [9]	VLC [9]	ours
1	1.00	0.8741	0.8721	**0.6528**
2	1.81	0.8778	0.8438	**0.6588**
3	2.92	0.9213	0.8212	**0.6675**

TABLE 6
The Comprehensive Comparison for the Generators With Different Number of LSTM Layers on Twitter Corpus

number of layer	EMD			anti-steganalysis		
	1bpw	2bpw	3bpw	1bpw	2bpw	3bpw
1	1.2477	1.3522	1.5408	0.7173	0.7497	0.7588
2	1.2405	1.3489	1.5376	0.6593	0.6743	0.6775
3	**1.2312**	**1.3456**	**1.5324**	**0.6528**	**0.6558**	**0.6675**

TABLE 7
The Comprehensive Comparison for the Discriminator With Different Types of Kernel Sizes Of CNN Layer on Twitter Corpus

type of kernel	EMD			anti-steganalysis		
	1bpw	2bpw	3bpw	1bpw	2bpw	3bpw
small [14]	1.2512	1.3559	1.5407	0.6693	0.6771	0.6833
odd [38]	1.2466	1.3508	1.5384	0.6621	0.6712	0.6792
all	**1.2312**	**1.3456**	**1.5324**	**0.6528**	**0.6558**	**0.6675**

other tested architectures. This is because the deeper LSTM architecture has better ability of long sequence modeling.

4.4.2 The Investigation of Discriminator

Based on the investigation of generator, we first fix the parameters and architecture of the generator equipped with 3 LSTM layers, and then change the type of kernel size in the discriminator. Specifically, we compared the proposed discriminator architecture equipped with the richer kernel sizes ("all" (1,2,3,4,5,6,7,8,9,10,15,20)) with other two architectures equipped with types of kernel sizes proposed in LS-CNN [14] ("small" (5,7,9)) and SeqGAN [38] ("odd" (1,3,5,7,9,15),), respectively. The experimental results are shown in Table 7. It is can be seen that as the number of kernel size increases, the security performance of the proposed text steganography is becoming better. This is because the more kernel size can capture better feature representation leading to yielding a better reward to guide the generator training phase.

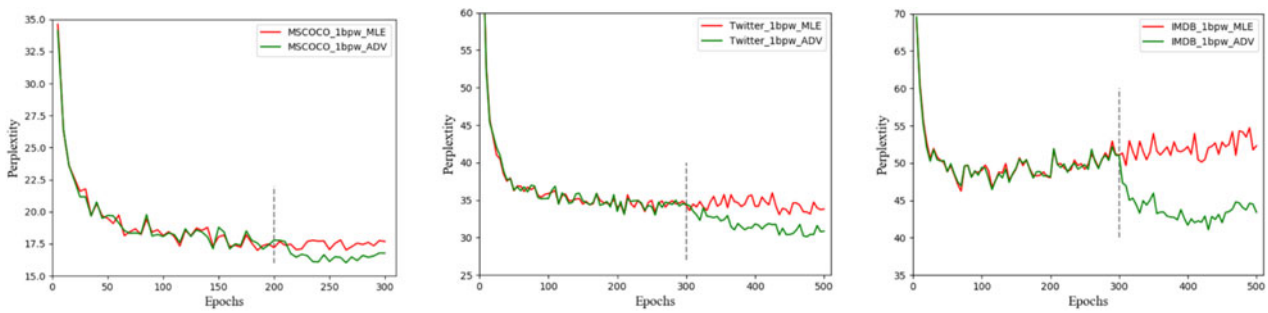
Based on the above investigations, we design an effective GAN-based language model to generate stego texts. Different from the original SeqGAN architecture [38], the generator of the proposed GAN-based language model is equipped with three LSTM layers rather than single LSTM layer. In addition, the "all" type of kernel size is leveraged in convolutional layer of the discriminator, which contains richer kernel size than LS-CNN and SeqGAN.

4.5 Discussion on the Role of Generative Adversarial Network

In order to explore the role of the generative adversarial network, we have carried out experimental analyses from multiple dimensions. We show the change curves of text quality, statistical distribution, and anti-steganalysis ability with 1 bpw in the Figs. 5 and 6, to demonstrate the quality, and security of text can be improved through the adversarial operation. The first 200, 300, and 300 epochs constitute the pre-training phase on Twitter, MSCOCO, and IMDB, respectively, and the rest constitute the adversarial training phase. In Fig. 5, the red curves are the LSTM model trained by MLE algorithm, and the green curves are the GAN-based model. And in Fig. 6, the red curves are the accuracy of steganalysis detection, and the green curves represent EMD. It can be found that from epoch 150 to epoch 200, the model tends to be stable, and perplexity and EMD hardly drops. But after starting the adversarial training, the performance of the model has been further improved, and the perplexity and EMD decrease, which means the text quality is improved and its distribution is closer to the real distribution. From Fig. 6, we can find that the steganalysis accuracy significantly decreases after adversarial operation. This is because the generative adversarial network solves the model deviation problem, which is caused by the inconsistent training and inference process, mentioned in Section 2. And by leaking the information of the discriminator to the generator and guiding its training, the anti-steganalysis ability of the steganographic algorithm is greatly improved. In the following experiments in Section 4.5, we will prove that the proposed model also has a satisfying anti-steganalysis ability against different steganalysis models.

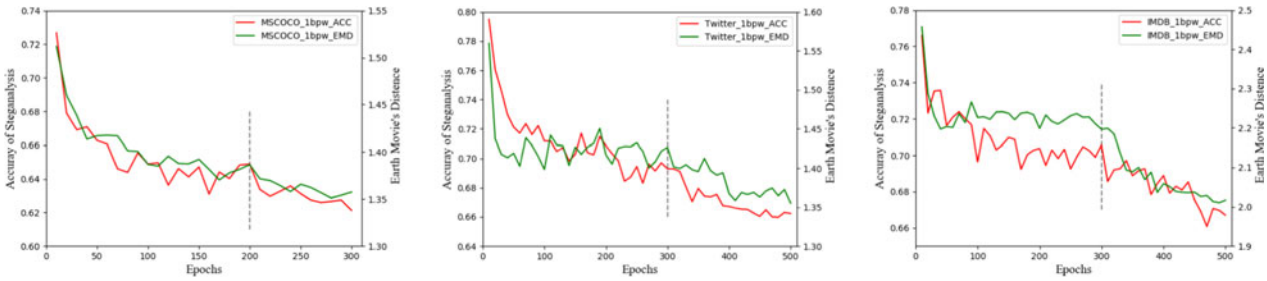
4.6 Comparison With Other Models

In this part, we compare the proposed model with other steganographic models, including Morkov-based method, Tina Fang's LSTM-based method (LSTM-Block), LSTM-FLC, and LSTM-VLC, in terms of text quality, statistical distribution, anti-steganalysis ability, and so on. First, we compare the quality of stego text generated by different steganographic models. The results are shown in Table 8. We can draw the following conclusions based on Table 5.



(a) MSCOCO (b) Twitter (c) IMDB

Fig. 5. PPL curve of adversarial training.



(a) MSCOCO (b) Twitter (c) IMDB

Fig. 6. EMD curve and steganalysis accuracy curve of adversarial training.

TABLE 8
The Mean and Standard Deviation of the Perplexity Results of Different Steganographic Models

Dataset	bpw	bpw(VLC)	Markov [6]	LSTM-Block [8]	LSTM-VLC [9]	LSTM-FLC [9]	The proposed
3*Twitter	1	1.00	94.34 ± 0.65	44.28 ± 1.04	45.10 ± 0.71	38.64 ± 1.07	$\mathbf{28.12 \pm 0.11}$
	2	1.81	103.99 ± 1.24	57.31 ± 1.95	52.17 ± 1.99	54.59 ± 2.42	$\mathbf{40.80 \pm 0.24}$
	3	2.92	108.07 ± 2.58	98.57 ± 3.06	78.58 ± 3.80	86.46 ± 2.72	$\mathbf{67.93 \pm 0.98}$
3*MSCOCO	1	1.00	62.15 ± 0.84	31.24 ± 1.52	28.68 ± 0.52	21.15 ± 0.72	$\mathbf{15.62 \pm 0.28}$
	2	1.79	68.47 ± 1.14	58.49 ± 3.85	35.89 ± 1.48	37.29 ± 1.38	$\mathbf{17.57 \pm 0.20}$
	3	2.94	71.25 ± 0.89	66.91 ± 2.77	42.47 ± 2.61	87.46 ± 0.58	$\mathbf{24.81 \pm 0.57}$
3*IMDB	1	1.00	117.98 ± 2.58	84.53 ± 2.24	40.58 ± 1.47	$\mathbf{37.16 \pm 0.94}$	38.66 ± 0.29
	2	1.79	135.93 ± 4.55	100.61 ± 1.89	46.77 ± 2.11	51.91 ± 1.26	$\mathbf{45.97 \pm 0.55}$
	3	2.91	148.21 ± 7.01	140.31 ± 2.57	$\mathbf{65.98 \pm 1.96}$	88.21 ± 2.01	87.33 ± 2.13

1) The types of corpora have a great influence on perplexity. It can be seen that the perplexity on MSCOCO is the lowest, which is mainly caused by two reasons. One is the characteristics of the corpus itself. MSCOCO consists of simple sentences, so the corresponding perplexity will be relatively low. Second, it is easy for GAN to build simple language model and generate text with similar structure. This is like when GAN is used to learn how to draw pictures, it is easier to learn and draw simple cartoon face rather than elaborate photos. Compared with MSCOCO, Twitter belongs to a free and personalized language style, and IMDB is relatively long and diversiform. Both of them contain many hard-to-understand colloquial sentences, which is relatively tricky to model. Hence, the perplexity on Twitter and IMDB is a slightly higher than that on MSCOCO.
2) For each corpus, with the increase of embedding rate, perplexity gradually increases and the text quality is destroyed. The reason is that as the number of bits embedded in each word increases, the corresponding disturbance will be larger, and it will be more difficult to choose a coherent word.
3) The LSTM-based is significantly better than Markov-based. This is because Markov model utilize the low order N-Gram to approximate language model, and use the frequency to approximate probability. Unlike Markov model, LSTM can better model the long-distance dependency of language model by using its hidden state.
4) The adaptive GAN-based steganographic method is better than the LSTM-based method. This is also in line with our analysis in Section 2. On the one hand, because the GAN-based network eliminates the model deviation, which is the inconsistency between training and prediction phrases, the generated text distribution is closer to the normal text distribution. On the other hand, the adaptive embedding algorithm reasonably selects the candidate words with similar probability to the target words for embedding. It dramatically reduces the disturbance caused by the embedding operation and improves the quality of the text.

In addition, we also compare the statistical distribution of the stego text generated by each steganographic model with the normal text distribution. We quantitatively evaluate the distance between the stego text distribution and the normal text distribution based on the Earth Mover's Distance, and the results are shown in Table 9. It can be found that for each steganographic model, the EMD distance increases with the increase of embedding capacity. This is because, with the increase of embedding capacity, the disturbance also increases, making the distribution of the generated stego text gradually deviate from the normal text distribution, and increasing of the distance between the two distributions. In addition, we can see that the distribution distance of our method is significantly lower than the other steganography models. This is because we adopt the adaptive embedding algorithm to dynamically determine the word candidate pool according to the probability similarity, so as to ensure the consistency of the distribution of stego text and normal text.

After that, we test the anti-steganalysis ability of the steganographic model. The results are shown in Table 10 and Fig. 7. We can conclude with previous experiments that the

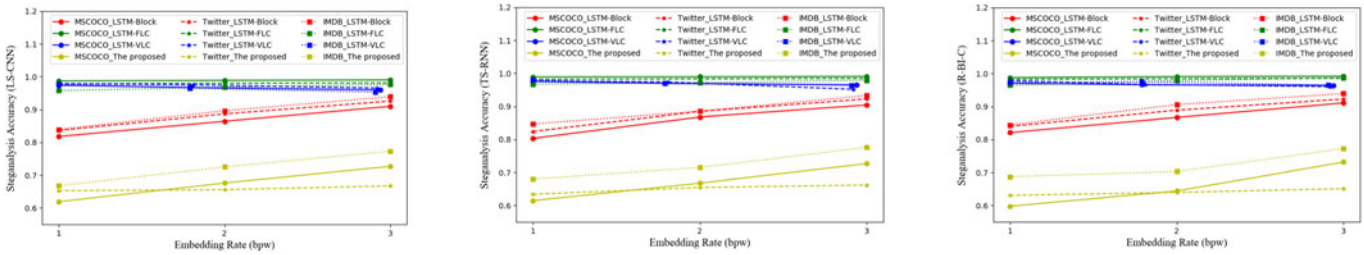
TABLE 9
The Earth Mover's Distence of Stego Text Distribution Generated by Different Steganographic Models

Dataset	bpw	bpw(VLC)	Markov	LSTM-Block	LSTM-VLC	LSTM-FLC	The proposed
3*Twitter	1	1.00	2.2675	1.5388	1.8525	2.1497	**1.2312**
	2	1.81	2.6712	1.7225	1.9531	2.5744	**1.3456**
	3	2.92	3.0015	1.9822	2.5578	2.8845	**1.5324**
3*MSCOCO	1	1.00	1.8611	1.6624	1.8524	1.7354	**1.3613**
	2	1.79	2.2105	1.9517	1.9577	2.2448	**1.4426**
	3	2.94	2.5025	2.1808	2.3024	2.6674	**2.0147**
3*IMDB	1	1.00	2.4974	2.6854	2.6729	2.6018	**2.0122**
	2	1.79	2.8433	2.7043	2.7155	2.7581	**2.1507**
	3	2.91	3.1245	2.7860	2.9021	2.8876	**2.2245**

proposed model has a better ability to resist steganalysis detection than MLE based method. It can be seen that the steganalysis detection rate of our proposed model is significantly lower than that of other algorithms on different corpora and different embedding capacity. When the detection rate of other steganalysis methods is 85 to 99percent, the detection rate of our model is about 65percent, which is significantly lower than the previous model. This is because in the model training process, we use adversarial training. By continually feeding the information of the discriminator to the generator to guide the training of the generator, we can obtain a model more resistant to steganalysis. It can also be seen from the experiment that our model can not only resist the detection of steganalysis model which is similar to the discriminator, but also resist the detection of many other types of steganalysis models. In addition, it can be known that the mixing of multiple corpora will lead to further degradation of detection accuracy.

TABLE 10
The Anti-Steganalysis Ability of Different Steganographic Models

2*Dataset	2*bpw	bpw (VLC)	2*Method	LS-CNN [14]		TS-RNN [47]		R-BI-C [32]	
				Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
12*Twitter	4*1	4*1.00	LSTM-Block	0.8365	0.8364	0.8240	0.8240	0.8400	0.8400
			LSTM-VLC	0.9795	0.9795	0.9773	0.9772	0.9786	0.9795
			LSTM-FLC	0.9786	0.9786	0.9823	0.9823	0.9827	0.9827
			The proposed	**0.6528**	**0.6531**	**0.6341**	**0.6337**	**0.6306**	**0.6306**
	4*2	4*1.81	LSTM-Block	0.8870	0.8869	0.8850	0.8850	0.8890	0.8889
			LSTM-VLC	0.9758	0.9768	0.9742	0.9745	0.9670	0.9692
			LSTM-FLC	0.9803	0.9803	0.9837	0.9837	0.9847	0.9847
			The proposed	**0.6558**	**0.6558**	**0.6545**	**0.6542**	**0.6396**	**0.6401**
	4*3	4*2.92	LSTM-Block	0.9260	0.9260	0.9230	0.9229	0.9230	0.9230
			LSTM-VLC	0.9658	0.9660	0.9518	0.9521	0.9609	0.9607
			LSTM-FLC	0.9815	0.9815	0.9853	0.9853	0.9857	0.9857
			The proposed	**0.6675**	**0.6673**	**0.6620**	**0.6620**	**0.6509**	**0.6509**
12*MSCOCO	4*1	4*1.00	LSTM-Block	0.8180	0.8180	0.8030	0.8030	0.8210	0.8210
			LSTM-VLC	0.9741	0.9743	0.9758	0.9757	0.9709	0.9709
			LSTM-FLC	0.9875	0.9875	0.9896	0.9896	0.9880	0.9880
			The proposed	**0.6192**	**0.6192**	**0.6145**	**0.6137**	**0.5978**	**0.5981**
	4*2	4*1.79	LSTM-Block	0.8645	0.8643	0.8680	0.8680	0.8670	0.8669
			LSTM-VLC	0.9679	0.9697	0.9700	0.9717	0.9662	0.9682
			LSTM-FLC	0.9892	0.9892	0.9899	0.9899	0.9910	0.9910
			The proposed	**0.6761**	**0.6761**	**0.6676**	**0.6666**	**0.6439**	**0.6404**
	4*3	4*2.94	LSTM-Block	0.9100	0.9100	0.9045	0.9044	0.9110	0.9110
			LSTM-VLC	0.9600	0.9600	0.9655	0.9656	0.9640	0.9642
			LSTM-FLC	0.9905	0.9905	0.9911	0.9911	0.9921	0.9921
			The proposed	**0.7270**	**0.7271**	**0.7270**	**0.7275**	**0.7315**	**0.7314**
12*IMDB	4*1	4*1.00	LSTM-Block	0.8395	0.8395	0.8470	0.8470	0.8445	0.8444
			LSTM-VLC	0.9775	0.9775	0.9800	0.9799	0.9730	0.9730
			LSTM-FLC	0.9580	0.9579	0.9675	0.9675	0.9645	0.9645
			The proposed	**0.6678**	**0.6672**	**0.6798**	**0.6798**	**0.6874**	**0.6875**
	4*2	4*1.79	LSTM-Block	0.8965	0.8965	0.8855	0.8855	0.9060	0.9060
			LSTM-VLC	0.9655	0.9654	0.9700	0.9699	0.9757	0.9758
			LSTM-FLC	0.9685	0.9685	0.9720	0.9720	0.9787	0.9787
			The proposed	**0.7235**	**0.7237**	**0.7105**	**0.7105**	**0.7030**	**0.7029**
	4*3	4*2.91	LSTM-Block	0.9395	0.9394	0.9335	0.9335	0.9395	0.9395
			LSTM-VLC	0.9535	0.9535	0.9640	0.9639	0.9655	0.9655
			LSTM-FLC	0.9765	0.9765	0.9795	0.9795	0.9876	0.9876
			The proposed	**0.7730**	**0.7730**	**0.7760**	**0.7753**	**0.7725**	**0.7726**



(a) LS-CNN (b) TS-RNN (c) R-BI-C

Fig. 7. Steganalysis accuracy of different steganographic models.

TABLE 11
The Comprehensive Comparison for Three Different Stego Sentence Datasets on Twitter Corpus

2*Method	2*Metrics	10K			50K			100K		
		1bpw	2bpw	3bpw	1bpw	2bpw	3bpw	1bpw	2bpw	3bpw
2*LSTM-Block [8]	EMD	1.5319	1.7240	1.9850	1.5397	1.7287	1.9819	1.5341	1.7288	1.9810
	Anti-steganalysis	0.8365	0.8870	0.9260	0.8343	0.8808	0.9279	0.8354	0.8819	0.9247
2*LSTM-FLC [9]	EMD	2.1501	2.5788	2.9045	2.1622	2.5823	2.9076	2.1698	2.5911	2.9132
	Anti-steganalysis	0.9786	0.9803	0.9815	0.9823	0.9877	0.9902	0.9867	0.9912	0.9945
3*LSTM-VLC [9]	bpw (VLC)	1.00	1.81	2.53	1.00	1.81	2.53	1.00	1.81	2.54
	EMD	1.8533	1.9587	2.5601	1.8674	2.0132	2.6544	1.8743	2.0254	2.6889
	Anti-steganalysis	0.9795	0.9758	0.9658	0.9822	0.9783	0.9703	0.9877	0.9834	0.9756
2*The proposed	EMD	**1.2352**	**1.3418**	**1.5397**	**1.2328**	**1.3417**	**1.5361**	**1.2396**	**1.3477**	**1.5347**
	Anti-steganalysis	**0.6528**	**0.6558**	**0.6675**	**0.6520**	**0.6577**	**0.6683**	**0.6544**	**0.6598**	**0.6711**

Finally, in order to evaluate the effectiveness of the proposed method for maintaining the consistency of probability distribution, several experiments are conducted on extra three different scale datasets, including the "10K", "50k", and "100k". In other words, there are 10k, 50k and 100k stego sentences generated by the proposed text steganography, and the same number of cover sentences are randomly selected from the training corpus. In this experiment, we compare the proposed method with three advanced text steganographic methods in terms of EMD and anti-steganalysis ability. The experimental results are listed in Table 11. We can draw following conclusions:

1) In three different stego sentence datasets, the proposed text steganographic method can effectively maintain the probability distribution consistency, and dramatically outperforms other tested methods. This is because the probability adaptive embedding algorithm is used to generate words with different probabilities, so that the actual sampling distribution is consistent with the theoretical sampling distribution calculated by language model.
2) The proposed method can achieve the high security against steganalysis. Because unlike other tested methods, the proposed text steganographic method can maintain the same statistical characteristics as the nature language, it is hard to be detected by statistical-based steganalysis method.

5 Conclusion

With the development of deep learning and natural language processing technology, generation-based linguistic steganography shows a strong prospect and has attracted more and more attention. In this paper, we propose a linguistic steganographic model based on Generative Adversary Network, which can automatically generate the stego text with high-quality and high-security driven by secret information for covert communication. In the process of adversarial training, the information of the discriminator is leaked to the generator, which guides the generator to generate the stego text that is more consistent with the normal text. At the same time, the inherent deviation of the model based on MLE algorithm is also eliminated. Therefore, a statistical language model that is more in line with the statistical law of natural language has been established and used to generate the stego text. Furthermore, we propose a probability-based adaptive embedding algorithm, which determines the candidate word space and embedding capacity according to the similarity of word probability, so as to reduce the deviation caused by embedding operation. We designed several experiments to verify the model from the aspects of imperceptibility, statistical distribution and anti-steganalysis ability. Experiments show that our scheme outperforms the previous models, especially in anti-steganalysis ability, which shows a promising way to enhance steganographic security.

Acknowledgments

The authors would like to sincerely thank the editors and reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grants 61872368 and No. 61802410. Xuejing Zhou and Wanli Peng contributed equally to this work.

REFERENCES

[1] B. Li, S. Tan, M. Wang, and J. Huang, "Investigation on cost assignment in spatial image steganography," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 8, pp. 1264–1278, Aug. 2014.

[2] X. Liao, J. Yin, M. Chen, and Z. Qin, "Adaptive payload distribution in multiple images steganography based on image texture features," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.3004708.

[3] A. Desoky, "Comprehensive linguistic steganography survey," *Int. J. Inf., Comput. Secur.*, vol. 4, no. 2, pp. 164–197, 2010.

[4] Y. Luo, and Y. Huang, "Text steganography with high embedding rate: Using recurrent neural networks to generate chinese classic poetry," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2017, pp. 99–106.

[5] J. Wen, X. Zhou, M. Li, P. Zhong, and Y. Xue, "A novel natural language steganographic framework based on image description neural network," *J. Vis. Commun. Image R.*, vol. 61, pp. 157–169, 2019.

[6] W. Dai, Y. Yu, Y. Dai, and B. Deng, "Text steganography system using Markov chain source model and des algorithm," *J. Softw.*, vol. 5, no. 7, pp. 785–792, 2010.

[7] H. Moraldo, "An approach for text steganography based on Markov chains," 2014. [Online]. Available: https://arxiv.org/abs/1409.0915

[8] T. Fang, M. Jaggi, and K. Argyraki, "Generating steganographic text with LSTMs," in *Proc. ACL Student Res. Workshop*, 2017, pp. 100–106.

[9] Z. Yang, X. Guo, Z. Chen, Y. Huang, and Y. Zhang, "RNN-Stega: Linguistic steganography based on recurrent neural networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 5, pp. 1280–1295, May 2019.

[10] Y. Tew, and K. Wong, "An overview of information hiding in H.264/AVC compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24 no. 2, pp. 305–319, Feb. 2014

[11] Y. Xue, J. Zhou, H. Zeng, P. Zhong and J. Wen, "An adaptive steganographic scheme for H.264/AVC video with distortion optimization," *Signal Proc. Image Commun.*, vol. 76, no. 8, pp. 22–30, 2019

[12] R. Zhang, F. Zhu, J. Liu, and G. Liu, "Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-Based steganalysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1138–1151, Aug. 2020.

[13] Z. Lin, Y. Huang, and J. Wang "RNN-SM: Fast steganalysis of VoIP streams using recurrent neural network," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 7, pp. 1854–1969, Jul. 2018.

[14] J. Wen, X. Zhou, P. Zhong, and Y. Xue, "Convolutional neural network based text steganalysis," *IEEE Signal Proc. Lett.*, vol. 26, no. 3, pp. 460–464, Mar. 2019.

[15] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*, Cambridge, U.K.: Cambridge Univ. Press, 2009.

[16] J. Zhang, J. Shen, L. Wang, and H. Lin, "Coverless text information hiding method based on the word rank map," in *Proc. Int. Conf. Cloud Comput. Secur.*, 2016, pp. 145–155.

[17] Z. Zhou, Y. Mu, L. Wang, and Q.J. Wu, "Coverless image steganography using partial-duplicate image retrieval," *Soft Comput.*, vol. 23, no. 13, pp. 1–12, 2018.

[18] J. Zhang, Y. Xie, L. Wang, and H. Lin, "Coverless text information hiding method using the frequent words distance," in *Proc. Cloud Comput. Secur.*, 2017, pp. 121–132.

[19] T. Brassil, S. Low, and F. Maxemchuk, "Electronic marking and identification techniques to discourage document copying," *IEEE J. Selected Areas Commun.*, vol. 13, no. 8, pp. 1495–1504, Oct. 1995.

[20] D. Huang and H. Yan,"Interword distance changes represented by sine waves for watermarking text images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 12, pp. 1237–1245, Dec. 2001.

[21] S. Ekodeck and R. Ndoundam, "PDF steganography based on chinese remainder theorem," *J. Inf. Secur. Appl.*, vol. 29, pp. 1–15, 2016.

[22] Z. Chen *et al.*, "Linguistic steganography detection using statistical characteristics of correlations between words," in *Proc. Int. Workshop Inf. Hiding*, 2008, pp. 224–235.

[23] I. Bolshakov and A. Gelbukh,, "Synonymous paraphrasing using wordnet and internet," in *Proc. Natural Lang. Proc. Inf. Sys.*, 2004, pp. 312–323.

[24] U. Topkara, M. Topkara, and M. Atallah, "The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions," in *Proc. 8th Workshop Multimedia Secur*, 2008, pp. 164–174.

[25] M. Li, K. Mu, P. Zhong, J. Wen, and Y. Xue, "Generating steganographic image description by dynamic synonym substitution," *Signal Process.*, vol. 164, pp. 193–201, 2019

[26] L. Xiang, X. Wang, C. Yang, and P. Liu, "A novel linguistic steganography based on synonym run-length encoding," *Inst. Electron. Inf. Commun. Eng. Trans. Inf. Syst.*, vol. 100, no. 2, pp. 313–322, 2017.

[27] L. Xiang, X. Sun, G. Luo, and B. Xia, "Linguistic steganalysis using the features derived from synonym frequency," *Multimedia Tools Appl.*, vol. 71, no. 3, pp. 1893–1911, 2014.

[28] Z. Chen, L. Huang, P. Meng, W. Yang, and H. Miao, "Blind linguistic steganalysis against translation based steganography," in *Proc. Int. Workshop Digit. Watermarking*, 2010, pp. 251–265.

[29] P. Wayner, "Mimic functions," *Cryptologia*, vol. 16, no. 3, pp. 193–214, 1992.

[30] M. Chapman and G. Davida, "Hiding the hidden: A software system for concealing ciphertext as innocuous text," in *Proc. Int. Conf. Inf. Commun. Secur.*, 1997, pp. 335–345.

[31] H. Yang and X. Guo, "Linguistic steganalysis based on meta features and immune mechanism," *Chin. J. Electron.*, vol. 19, no. 4, pp. 661–666, 2010.

[32] Y. Niu, J. Wen, P. Zhong, and Y. Xue, "A hybrid R-BILSTM-C neural network based text steganalysis," *IEEE Signal Proc. Lett.*, vol. 26, no. 12, pp. 1907–1911, Dec. 2019.

[33] Y. Bao, H. Yang, and Y. Huang, "Text steganalysis with attentional LSTM-CNN," in *Proc. IEEE 5th Int. Conf. Comp. Commun. Syst.*, 2020, pp. 138–142.

[34] W. Tang, S. Tan, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Proc. Lett.*, vol. 24, no. 10, pp. 1547–1581, Oct. 2017.

[35] J. Yang, D. Ruan, J. Huang, X. Kang, and Y. Shi, "An embedding cost learning framework using GAN," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 839–852, Jun. 2020.

[36] Z. Yang, N. Wei, Q. Liu, Y. Huang, and Y. Zhong "GAN-TStega: Text steganography based on generative adversarial networks," in *Proc. Int. Workshop Digit. Watermarking*, 2019, pp. 18–31.

[37] H. Hu, X. Zou, W. Zhang, and N. Yu "Adaptive text steganography by exploring statistical and linguistical distortion," in *Proc. IEEE Conf. Data Sci. Cyberspace*, 2017, pp. 145–150.

[38] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31st Assoc. Advancement Artif. Intell. Conf. Art. Intell.*, 2017, pp. 2852–2858.

[39] B. Browne, E. Powley, and D. Whitehouse, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. AI Game*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[40] P. Meng, L. Hang, and W. Yang, "TBS: A statistical algorithm for steganalysis of translation-based steganography," in *Proc. Int. Workshop Inf. Hiding*, pp. 208–220, 2010.

[41] R. Sutton, D. McAllester, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Conf. Workshop Neural Inf. Proc. Syst.*, 1999, pp. 1057–1063.

[42] A. Go, R. Bhayani, and L. huang, "Twitter sentiment classification using distant supervision," CS224N Project, Stanford, CA, USA, Tech. Rep., vol. 1, no. 12, 2009.

[43] T. Lin, M. Maire, S. Belongie, J. Hays, and P. Perona, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–750.

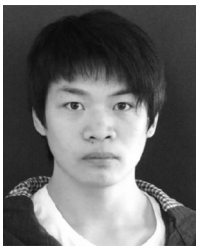
[44] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 9th Annu. Meeting Assoc. Comput. Linguistics Human Lang. Technol.*, 2011, pp. 142–150.

[45] D. Lingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] Z. Yang, K. Wang, J. Li, and Y. Huang, "TS-RNN: Text steganalysis based on recurrent neural networks," *IEEE Signal Proc. Lett.*, vol. 26, no. 12, pp. 1743–1747, Dec. 2019.

Xuejing Zhou received the BE and MS degrees in college of information and electrical engineering from China Agricultural University. Her research interests include artificial intelligence, information hiding and natural language processing.



Wanli Peng received the BE degree in collage of physics and electronic engineering from Harbin Normal University, China, in 2018. He is currently working toward the PhD degree with the College of Information and Electrical Engineering, China Agricultural University, Beijing, China. His research interests include information hiding and natural language processing.

Boya Yang received the BE degree from the Department of Computer Science and Technology, Beijing Jiaotong University, Beijing, in 2019. She is currently working toward the the ME degree with the Department of Information and Electrical Engineering, China Agricultural University, Beijing. Her current research interests include information hiding and natural language processing.

Juan Wen received the BE degree in information engineering and PhD degree in signal and information processing from the Beijing University of Post and Telecommunication. She is currently an associate professor with the College of Information and Electrical Engineering, China Agricultural University. Her research interests include artificial intelligence, information hiding and natural language processing.

Yiming Xue (Member, IEEE) is currently an associate professor with the College of Information and Electrical Engineering, China Agricultural University. His research interests include multimedia processing, multimedia security, VLSI design.

Ping Zhong is a professor and PhD supervisor in College of Science, China Agricultural University, Beijing, China. She has published many papers. Her research interests include machine learning, support vector machines, steganalysis.