Journal of Information Security and Applications 75 (2023) 103512

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

V-A3tS: A rapid text steganalysis method based on position information and variable parameter multi-head self-attention controlled by length ☆

Yihao Wang, Ru Zhang *, Jianyi Liu

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China

ARTICLE INFO

Keywords:
Text steganalysis
Variable parameter
Fast detection
Multi-head self-attention mechanism
Text length

ABSTRACT

Text length varies in social networks, such as IMDB long text, Twitter short text, and long–short mixed text. For these complex situations, the time series, convolution, or fine-tuned BERT models are used in existing text steganalysis methods almost. However, these methods do not simultaneously consider higher detection accuracy and lower training time at the same time. To alleviate this dilemma, this paper proposes a novel text steganalysis method. First, the proposed method maps words into a semantic space containing position information. Second, a variable parameter attention layer scaled appropriately according to text length is designed, it achieves the purpose that the entire parameter amount of the model is not redundant and can ensure effective detection. Finally, the steganalysis features are enhanced by the residual linear layer. For long, short, and mixed text datasets, comparing experiments show that the proposed method has higher detection accuracy, fewer parameters, and shorter training time than existing methods. Among them, the advantage of this method is more obvious for long and mixed texts.

1. Introduction

Steganography is a technique of concealment system in covert security systems [1]. It hides secret information into digital media such as texts [2,3], images [4–7], etc., and transmits them through open channels. In recent years, thanks to the popularity of social platforms, ubiquitous text carriers (i.e. cover) exist on the Internet for researchers to study. Text steganography has achieved explosive growth [2,3,8–14]. However, many hidden dangers exist in texts in social networks, once steganography is abused by criminals to generate steganographic text (i.e. stego), it will cause unimaginable damage [12,15]. In the offensive and defensive game of steganography and steganalysis, steganalysis is weak. Therefore, steganalysis has received extensive attention from scholars [16–29].

Traditional text steganalysis schemes are devoted to constructing statistical features artificially [30–34]. The focus of such schemes is how to construct more effective features of word association [30], word distribution [31], and so on. Since text steganography has evolved from text modification schemes based on typesetting coding structure and grammar [32,33] to generative schemes which can generate high-quality text-based on deep learning [2,3,9–14]. If the traditional schemes are used for steganalysis and the richness of feature types will be limited, the quantification ability of steganographic embedding disturbance is not robust enough. This will fall into the dilemma of low detection accuracy and universality, and it is hard to generalize on distinct types of datasets. So researchers have shifted the focus of their work to how to design deep learning steganalysis that can capture features of high dimensions [16–28].

Compared with traditional artificially constructed steganalysis schemes, deep learning-based steganalysis can extract features with strong diversity. This type of scheme significantly improves the detection effect of stego texts [16,17]. Although these models can analyze effectively, the time series models represented by Long Short-Term Memory (LSTM) result in a slow processing speed [17,20]. Although the convolution models represented by Convolutional Neural Network (CNN) and Graph Convolutional Network (GCN) can process the entire sentence in parallel [18,19,23], the small receptive field in CNN and the short distance node aggregation in GCN are difficult to capture the dependencies at a long distance. The detection accuracy of these schemes needs to be further improved. If it is desired to capture all dependencies in a sentence by various-sized kernels, which would make the number of kernels huge and impractical. In general, the existing deep learning-based text steganalysis either designs a simple network for efficiency or uses complex word vectors and frameworks for high performance such as fine-tuned BERT [22,24,25], etc. They will cause low detection accuracy and high training costs, respectively. Taking fast detection and high performance simultaneously into account becomes

☆ This work was supported by the National Natural Science Foundation of China (Grant numbers U21B2020, U1936216).
* Corresponding author.
E-mail addresses: yh-wang@bupt.edu.cn (Y. Wang), zhangru@bupt.edu.cn (R. Zhang), liujy@bupt.edu.cn (J. Liu).

https://doi.org/10.1016/j.jisa.2023.103512

Available online 20 May 2023

a challenge. Furthermore, in our opinion, sentence structure and dependencies are often quite varied in texts with different lengths, these existing methods with fixed network parameters will be hard to capture appropriate dependencies in long and short texts at the same time.

Considering that a large-scale model is not required in short texts, a better detection effect can be obtained. On the contrary, reducing the size of the model blindly will be difficult to extract long-distance dependencies in long texts. Therefore, this paper proposes a text steganalysis method that focuses on both detection performance and training time, i.e. A **V**ariable Parameters Multi-Head **Att**ention Mechanism for Rapid **T**ext **S**teganalysis (**V-A3tS**). The main work of this method includes the following three points:

1. To reduce the time overhead while ensuring effective detection, the V-A3tS method abandons the fixed-parameter setting of the original attention mechanism and designs an attention layer that adapts to the text length. This model controls the size and head of attention vectors and the number of layers in the entire network. This design reduces the model scale and can adapt to text length, improving the efficiency of training.
2. To improve the expressiveness of features, we introduce absolute position embedding and residual linear layer. Absolute position embedding is different from the trainable position in BERT. This embedding can not only reduce the number of trainable parameters but also ensure detection performance.
3. To verify the effectiveness of V-A3tS method on datasets of distinct lengths, IMDB is used to construct seven long text datasets and three mixed text datasets to enrich the comparative experiments. Experiments indicate that V-A3tS method can adapt to texts with distinct lengths, simultaneously, it has better detection performance and shorter time-consuming than existing methods.

The rest of this paper involves the following parts: Section 2 introduces representative deep learning-based text steganalysis methods. The overall framework and design details of V-A3tS model proposed are described in Section 3. Section 4 is indicated the comparison results and analysis through a series of experiments. Finally, the full text is summarized in Section 5.

2. Related work

In this section, the existing text steganalysis methods will be introduced. Before the emergence of deep learning-based steganalysis, traditional artificially constructed text steganalysis was widely studied by researchers. The representative works are: Taskiran et al. [25] proposed a steganalysis method to detect synonym replacement, which uses the n-gram model to extract the feature vector of the text, and uses the support vector machine (SVM) to judge the text Whether to perform steganography. Yang et al. [27] proposed a steganalysis algorithm for detecting text template modification. The algorithm uses SVM to measure the variance of group distances between N-Window text lines. Given the diversity of syntax and semantics in NLP, Yang et al. [28] proposed a language steganalysis method based on meta-features and an immune cloning mechanism. Although these methods are easy to implement, they rely too much on design experience, and they are almost all designed for a certain class of steganographic schemes, lacking universality.

Because traditional steganalysis methods have bad accuracy for deep-learning-based methods, deep-learning-based text steganalysis methods are becoming research highlights in recent years [16–27]. Yang et al. [16] proposed a steganalysis method based on the correlation between words. This method simply maps words to a low-dimensional space, and a hidden layer is used to capture the dependencies. Yang et al. [17] used a Recurrent Neural Network (RNN) to extract the conditional probability distribution of text words, this model has the capability of fast steganalysis and estimation of payload. Text steganalysis methods based on CNN have been invented [18,19]. They obtained feature representations automatically by convolution kernels of different sizes. Yang et al. [20] believed that fusing multiple low-level features can better use semantic features, and proposed a feature pyramid-based densely connected LSTM steganalysis method. Based on the shortcomings of the above work, Niu et al. [21] combined CNN and LSTM to capture the long and short-distance dependencies. Zou et al. [22] strived for a method based on BERT and bidirectional LSTM. This method inputs the BERT vectors of the words into LSTM. Wu et al. [23] designed a GCN to extract features and adopted a shared matrix to enable each text to better self-representation. Peng et al. [24] introduced the idea of transfer learning in text steganalysis for the first time, BERT is regarded as the teacher model and LSTM and CNN are regarded as the student models to train the word representation fully. Yang et al. [25] designed a novel text steganalysis method by integrating the semantic and syntactic features of the text, which can preserve and utilize the syntactic structure. According to the distinct network models, existing methods are divided into four classes, namely time series models [17,20], convolution models [18,19,23], different network combination models [21,24] and BERT pre-trained models [22,25]. The comparison of their working principles, adapted length, advantage, disadvantage, and parallel is shown in Table 1.

where, $\mathbf{F}$ is the extracted steganalysis feature. hi_t represents the hidden layer node of LSTM at the tth moment, $\mathbf{F}^l$ represents the feature extracted by the lth layer CNN, $\mathbf{b}_i$ represents the feature extracted by the ith layer GCN, h^l represents the word vector extracted by the lth layer Transformer. fo_t, i_t, and O_t represent the forgetting gate, input gate, and output gate at the tth moment; K^l and b^l represent the convolution kernel and bias of the lth layer; v_{i-1} represents the $i-1$th word vector, it can be expressed as $\mathbf{h}_{i-1}^{\mathrm{T}}$, $\mathbf{D}$, $\mathbf{W}$, $\mathbf{a}$, and $\mathbf{w}^{\mathrm{T}}$ are the globally-shared vector, weight matrix, the representation of adjacent nodes, and the trainable parameters vector, respectively. tanh(·), pool(·), and concat(·) represent the tanh activation, pooling operation, and concat operation.

3. The proposed V-A3tS

3.1. Overall architecture

For variable-length social networks, V-A3tS as a rapid text steganalysis method is built, as shown in Fig. 1.

3.1.1. Word embedding layer

The first layer of V-A3tS is the word embedding layer, it maps text words to a specific semantic space to facilitate subsequent extraction of steganalysis features. Given a sequence of words $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$ with length n, where, t_i is the ith word, $i = \{1, 2, \ldots, n\}$. Through this layer, the semantic vector with position information of T can be obtained, i.e. $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_i, \ldots, \mathbf{e}_n]$. The equation involved in this layer is as follows.

$$\mathbf{e}_i = \mathbf{ce}_i + \mathbf{pe}_i \tag{1}$$

$$ce_i^j = random(0,1),$$
$$pe_i^j = \begin{cases} \sin(i/10000^{2j/m}), j = 2k(k \in \mathbb{N}) \\ \cos(i/10000^{2j/m}), j = 2k+1(k \in \mathbb{N}) \end{cases}, \tag{2}$$

where, $\mathbf{ce}_i$ is the m-dimensional character embedding of t_i, $\mathbf{ce}_i = [ce_i^1, ce_i^2, \ldots, ce_i^j, \ldots, ce_i^m]^{\mathrm{T}}$, $(\cdot)^{\mathrm{T}}$ is transpose, $\mathbf{pe}_i$ is the m-dimensional positional embedding of t_i, $\mathbf{pe}_i = [pe_i^1, pe_i^2, \ldots, pe_i^j, \ldots, pe_i^m]^{\mathrm{T}}$. $random(\cdot)$ represents a random value within a specified range, $\mathbb{N}$ is the set of natural numbers, $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. ce_i^j is random at the beginning and updated during training automatically.

Since text generative steganography considers not only the statistical features of each word, but also the semantic features of the entire sentence. If the positional encoding of each word is taken into account, it is more beneficial to capture the positional steganographic features

Table 1
Comparison of distinct types of network models.

Type	Working principle	Adapted len	Advantage	Disadvantage	Parallel
Time series models [17,20]	$fo_t = \sigma(W_f \times [hi_{t-1}, x_t] + b_f)$ $i_t = \sigma(W_i \times [hi_{t-1}, x_t] + b_i)$ $C_t = fo_t \odot C_{t-1} + i_t \odot \tanh(\tilde{C}_t)$ $O_t = \sigma(W_o \times [hi_{t-1}, x_t] + b_o)$ $\mathbf{F} = hi_t = O_t \odot \tanh(C_t)$	Short texts	1. It can effectively capture the longer dependencies between long-distance words. 2. Its implementation is relatively simple.	1. Word-by-word processing, which cannot be processed in parallel. It brings a large time cost.	incapable
Convolution models [18,19,23]	$\mathbf{F}^{n-1} = \sum_i \mathbf{F}_i^{l-1} * K_i^l + b_i^l$ $\mathbf{F}^l = \text{pool}(f^n(\mathbf{F}^{n-1}))$ $\mathbf{b}_{i-1} = \mathbf{h}_{i-1}^{\mathrm{T}}\mathbf{D} \leftarrow v_{i-1}$ $\mathbf{q}_i = (\mathbf{h}_{i-p}^{\mathrm{T}}\mathbf{W}\mathbf{h}_i, \ldots, \mathbf{h}_{i+p}^{\mathrm{T}}\mathbf{W}\mathbf{h}_i)$ $\mathbf{b}_i = \mathbf{w}^{\mathrm{T}}\mathbf{q}_i\mathbf{a}_i + (1 - \mathbf{w}^{\mathrm{T}}\mathbf{q}_i)\mathbf{b}_{i-1}$ $\mathbf{F} = \mathbf{F}^l$ or $\mathbf{b}_i$	Short texts	1. It can be processed in parallel and training speed is faster. 2. Effectively capturing short -distant dependencies in CNN. 3. Words get better self-representation by the shared matrix in GCN.	1. The small receptive field in CNN and the short distance node aggregation in GCN is difficult to capture the long-distant dependencies. 2. Huge convolutional layer leads to large space costs.	capable
Different network combination models [21,24]	$fo_t = \sigma(W_f \times [hi_{t-1}, x_t] + b_f)$ $i_t = \sigma(W_i \times [hi_{t-1}, x_t] + b_i)$ $C_t = fo_t \odot C_{t-1} + i_t \odot \tanh(\tilde{C}_t)$ $O_t = \sigma(W_o \times [hi_{t-1}, x_t] + b_o)$ $hi_t = O_t \odot \tanh(C_t)$ $\mathbf{F}^{l-1} = \text{concat}(hi_1, hi_2, \ldots, hi_t)$ $\mathbf{F}^l = \text{pool}(f^n(\mathbf{F}^{l-1} * K_i^l + b_i^l))$ $\mathbf{F} = \mathbf{F}^l$	Long texts Short texts	1. It combines the advantages of different networks to capture long and short-distance dependencies. 2. The types of combinations are diverse and can be applied to steganalysis in different scenarios.	1. The training time is close to the sum of the two separate training times, which will bring a large time cost.	partial
BERT pretrained models [22,25]	$h^l = \text{BERT}_l(h^{l-1})$ $\overrightarrow{hi_t} = \overrightarrow{\text{LSTM}}(h^l)$ $\overleftarrow{hi_t} = \overleftarrow{\text{LSTM}}(h^l)$ $\mathbf{F}^{n-1} = h^l * K^l + b^l$ $hi_t = [\overrightarrow{hi_t}, \overleftarrow{hi_t}]$ $\mathbf{F}^l = \text{pool}(f^n(\mathbf{F}^{n-1}))$ $\mathbf{F} = hi_t$ or $\mathbf{F}^l$	Long texts Short texts	1. It has a strong expression ability and greatly improves detection performance. 2. The ability to extract efficient semantic features and dependencies between long -short distance words	1. The large-scale pretrained structure such as BERT is huge, and it takes a lot of time to fine-tune it.	partial

of words in stego texts. So we introduced the absolute position embedding [35] to determine a unique encoding for each time step. As far as we know, [17–21] only uses character embedding, and [22,24,25] uses trainable position embedding, there is no reference that uses this absolute position embedding as position information to add to the word embedding layer. The absolute positional embedding is different from the trainable positional embedding in BERT [36]. The absolute positional embedding will not increase the size of trainable parameters, it will reduce the training time.

3.1.2. Variable parameters multi-head self-attention layer

To extract the features of steganographic text, a multi-head attention mechanism is considered. $\mathbf{E}$ of the word embedding layer is input to the designed variable-parameter multi-head self-attention layer, and the self-attention vector of T is obtained, i.e. $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_i, \ldots, \mathbf{h}_n]$. The equation involved in this layer is as follows.

$$\begin{aligned}\mathbf{h}_i &= S(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d_k}})V_i \\ &= S(\frac{\mathbf{W}_i^{(q)}\mathbf{q}_i \times (\mathbf{W}_i^{(k)}\mathbf{k}_i)^{\mathrm{T}}}{\sqrt{d_k}}) \times \mathbf{W}_i^{(v)}\mathbf{v}_i,\end{aligned} \tag{3}$$

$$\mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)} = \text{Lin}(\mathbf{e}_i, \mathbf{e}_i, \mathbf{e}_i), \tag{4}$$

$$\mathbf{q}_i = \mathbf{k}_i = \mathbf{v}_i = \mathbf{e}_i, \tag{5}$$

$$\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i, \in \mathbb{R}^d, \mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)} \in \mathbb{R}^{m \times d}, \tag{6}$$

where, Q_i, K_i, V_i are the query matrix, key matrix, and value matrix of $\mathbf{e}_i$, $\mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)}$ are the weight vector of $\mathbf{e}_i$ by linear transformation, the essence of $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$ is $\mathbf{e}_i$, d represents the dimension of $\mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)}$, $\text{Lin}(\cdot)$ represents the linear transformation operation, $S(\cdot)$ represents the Softmax operation. Using the above equation, $\mathbf{H}$ with the long-distance dependencies of T can be obtained.

Although the original Transformer model [35] or the BERT model [36] based on the attention mechanism can effectively extract long-range dependencies and semantic features of text, fine-tuning them to adapt to different domains will bring huge time and space overhead. Therefore, to achieve the purpose of fast detection by reducing the parameter size, but also to ensure the effectiveness of the detection, we propose a mechanism for model parameter modification, as shown in Section 3.2.

3.1.3. Residual linear layer

To enhance the $\mathbf{H}$ of the previous layer output, we use the residual linear layer. The final text steganalysis features are extracted through this layer, i.e. $\mathbf{F} = [f_1, f_2, \ldots, f_i, \ldots, f_n]$. The equation involved in this layer is as follows.

$$f_i = \text{Lin}(\text{ReLU}(\text{Lin}(\mathbf{Ln}_i))), \tag{7}$$

$$\mathbf{Ln}_i = \alpha \times \frac{\mathbf{h}_i - \mu_{\mathbf{h}_i}}{\sqrt{\sigma_{\mathbf{h}_i}^2 + \epsilon}} + \beta, \tag{8}$$

where, $\mu_{\mathbf{h}_i}$ and $\sigma_{\mathbf{h}_i}$ represent the mean and variance of $\mathbf{h}_i$, α, β, ϵ are parameters, $\text{ReLU}(\cdot)$ is ReLU activation.

This layer consists of two parts: one is the part that the residuals and normalization, where the normalization uses the Layer Normalization (LN) layer [37], and the other is the fully connected layer part with the activation function. Through this layer, $\mathbf{H}$ strengthens the expressive and fitting ability, and obtains $\mathbf{F}$.

Finally, $\mathbf{F}$ is generated as the distribution on the decision space by Softmax, i.e., the probability $Class$ of cover and stego, the equation is as follows.

$$Class = S(\mathbf{F}). \tag{9}$$

3.2. Parameter modification of multi-head self-attention mechanism

V-A3tS defines three parameters: d, $head$, and $layer$ to control self-attention. d is the dimension of the self-attention matrix, which determines the distance to capture text dependencies. $head$ is the dimension representing the attention subspace. $layer$ is the nonlinear fitting ability of the entire network. The relationship between them is

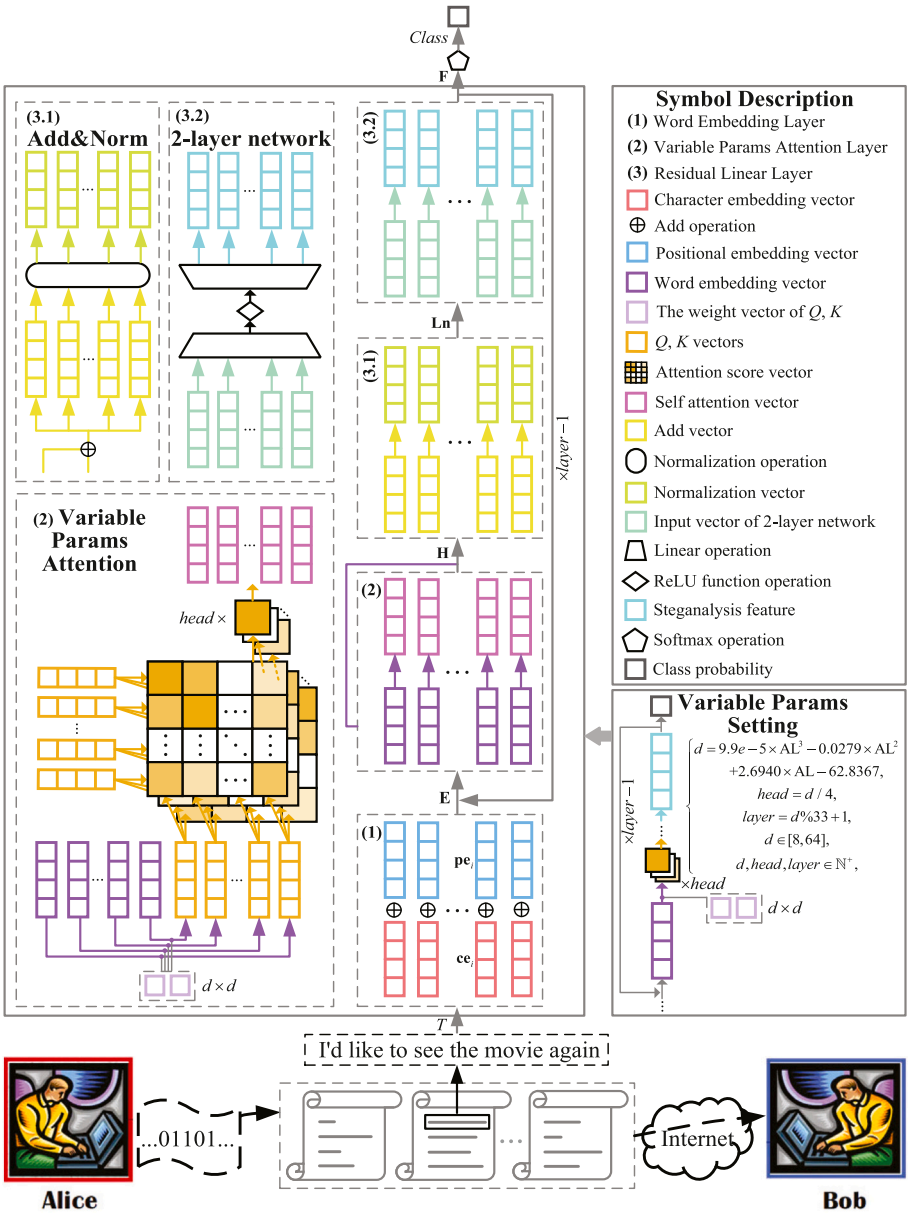


Fig. 1. Overall architecture of V-A3tS method.

empirically obtained as follows.

$$\begin{cases} d = 9.9e-5\times \mathrm{AL}^3 - 0.0279\times \mathrm{AL}^2 \\ \quad +2.6940\times \mathrm{AL} - 62.8367 \\ head = d/4 \\ layer = d\%33+1 \\ d \in [8,64] \\ d, head, layer \in \mathbb{N}^+ \end{cases}, \tag{10}$$

where, $e-n$ represents 10^{-n}, AL is the average length of texts, % is modulo operation, and $\mathbb{N}^+$ is the set of positive integers. Using these values is expected to give better results than other values, and we also use the above equation to calculate these parameters in experiments.

At the same time, using the above equation to obtain V-A3tS method is expected to reduce the parameter size and training time of the entire network. This is because the most significant part of training time is the multi-head self-attention mechanism with variable parameters. It is similar to the convolution operation in structure, that is, the attention of heads can be processed in parallel. Therefore, V-A3tS can be faster than the time series models represented by [17,20].

Although the convolution models can be processed in parallel, the convolution matrices are $(3+5+7)\times 128 = 1,920$ and $(3+4+5)\times 128 = 1,536$ in [18,19]. The attention sizes of V-A3tS method do not exceed $layer \times head \times d = 2\times 8\times 64 = 1,024$, and even less than 100 sizes can extract strong dependencies in shorter texts. Hence, V-A3tS has faster efficiency than convolution models represented by [18,19].

Because the training time of different network combination models [21,24] is close to the sum of the two separate training times. V-A3tS can train faster than combination models represented by [21,24].

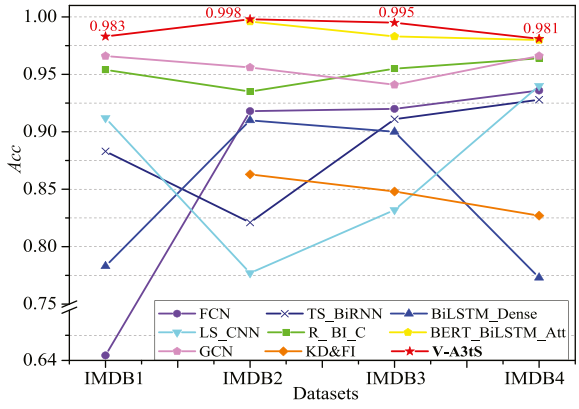
The BERT-based models [22,25] require at least 12-layer Transformer Encoders, the 64-size attention mechanism with 12 heads, and a 768-dimensional word vector representation, i.e. $layer = 12, head = 12, d = 64$. So the parameter size of V-A3tS is much fewer than BERT-based models represented by [22,25].

4. Experiment & analysis

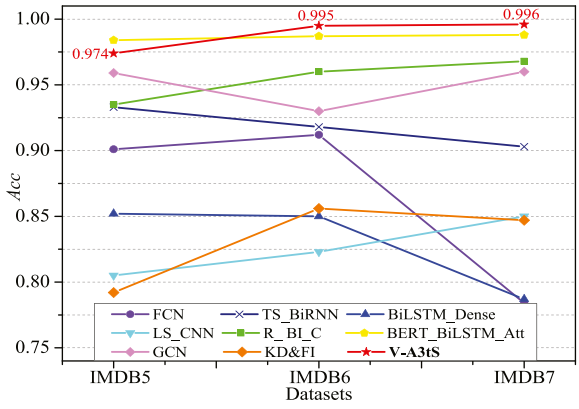
In this section, we designed several experiments to test V-A3tS from the perspectives of steganalysis accuracy, parameter size, and training time.

Table 2
Specific information of datasets.

Long text	Min Len	Max Len	AL	Num of words
IMDB1	10	50	41.58	25,116
IMDB2	51	100	75.74	180,115
IMDB3	101	150	127.66	880,190
IMDB4	151	203	163.46	428,440
IMDB5	10	70	53.00	81,517
IMDB6	71	140	115.40	802,532
IMDB7	141	203	157.21	629,812
Mixed Text	Min Len	Max Len	AL	Num of Words
IMDB8	10	203	121.11	1,513,861
IMDB9	10	203	121.11	1,513,861
IMDB10	10	203	121.11	1,513,861
Short Text	Min Len	Max Len	AL	Num of Words
Movie	1	50	8.11	16,995
Twitter	1	50	4.20	21,980



(a) The "accuracy" comparison of IMDB1 to IMDB4



(b) The "accuracy" comparison of IMDB5 to IMDB7

Fig. 2. Nine steganalysis methods "accuracy" comparison of long texts. The horizontal axis represents the datasets, and the vertical axis represents the detection accuracy.

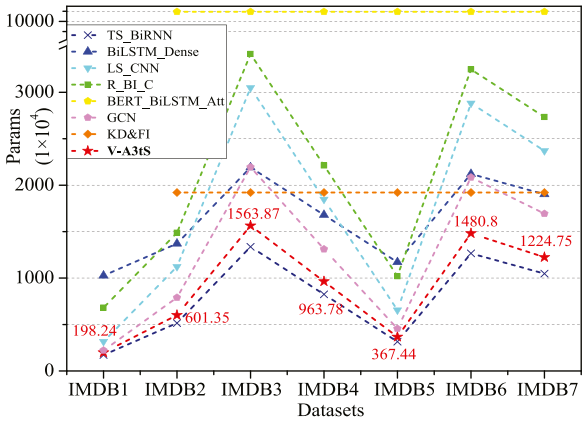


Fig. 3. Nine steganalysis methods "parameter size" comparison of long texts. The horizontal axis represents the datasets, and the vertical axis represents the parameter size.

4.1. Experiment setting

Three kinds of datasets are used: long text, mixed text, and short text datasets, the specific information is shown in Table 2.

From IMDB1 to IMDB7 are long text datasets. The long text dataset is 12,500 pieces of data from the IMDB [38]. IMDB1 is the text extracted with lengths from 10 to 50, and the AL of IMDB1 is 41.58. IMDB2 is the text extracted with lengths from 51 to 100, and the AL of IMDB2 is 75.74. That is IMDB1 to IMDB4 are divided at intervals of 50 text length, and IMDB5, IMDB6, and IMDB7 are divided at intervals of 70 text length. Then use ADG steganography [10] to generate the same number of stego text, and the text length is controlled within the corresponding range.

From IMDB8 to IMDB10 are the mixed text dataset. IMDB8 is combined with IMDB1 to IMDB4, and IMDB9 is combined with IMDB5 to IMDB7. IMDB10 generated the stego texts of the overall 12,500 pieces of data. The generated text control is between 10 and 203. It should be noted that cover of IMDB8 to IMDB10 is the same, they are all 12,500 pieces of IMDB, but stego of them are different.

Movie and Twitter are the short text dataset. The length of the short text datasets is no more than 10 on average, and Fang steganography [3] is used to generate stego text from 1 to 5 bits per word (bpw).

Ten existing methods are chosen for comparison including:

- SS [33]: Language steganalysis method based on meta-features and immune cloning mechanism;
- LSMethod [34]: Text steganalysis based on statistical language model;
- FCN [16]: Based on a simple and effective word-related steganalysis method, this method is the first attempt to use deep learning for text steganalysis;
- TS_BiRNN [17]: Based on steganalysis of bidirectional RNN (BiRNN), the conditional probability distribution of words is efficiently extracted;
- BiLSTM_Dense [20]: The text steganalysis method based on feature pyramid densely connected BiLSTM, which avoids the dilemma that it is hard for a single LSTM to extract more low-level features;

Table 3
V-A3tS steganalysis accuracy comparison of different parameters. Underlined parameters indicate the parameters calculated using Eq. (10), **bold values** represent the best results of each group of experiments.

Dataset	AL = 41.58		*Acc*	Dataset	AL = 75.74		*Acc*	Dataset	AL = 127.66		*Acc*	Dataset	AL = 163.46		*Acc*
IMDB1	$layer = 1$	$d = 8$	0.979	IMDB2	$layer = 1$	$d = 12$	0.992	IMDB3	$layer = 1$	$d = 24$	0.992	IMDB4	$layer = 1$	$d = 36$	0.946
	$head = 1$	$d = 12$	0.975		$head = 2$	$d = 24$	0.994		$head = 4$	$d = 32$	0.993		$head = 8$	$d = 48$	0.975
		$d = 24$	0.958			$d = 36$	0.993			$d = 36$	0.992			$d = 64$	0.979
	$layer = 1$	$d = 8$	**0.983**		$layer = 1$	$d = 12$	0.996		$layer = 1$	$d = 24$	0.992		$layer = 2$	$d = 36$	0.967
	$head = 2$	$d = 12$	0.981		$head = 4$	$d = 24$	0.994		$head = 6$	$d = 32$	0.994		$head = 8$	$d = 48$	0.978
		$d = 24$	0.971			$d = 36$	0.996			$d = 36$	0.993			$d = 64$	**0.981**
	$layer = 1$	$d = 8$	0.981		$layer = 1$	$d = 12$	0.996		$layer = 1$	$d = 24$	0.992		$layer = 2$	$d = 36$	0.973
	$head = 4$	$d = 12$	0.979		$head = 6$	$d = 24$	**0.998**		$head = 8$	$d = 32$	**0.995**		$head = 6$	$d = 48$	0.976
		$d = 24$	0.970			$d = 36$	0.995			$d = 36$	0.994			$d = 64$	0.977
	$layer = 2$	$d = 8$	0.971		$layer = 1$	$d = 12$	0.996		$layer = 2$	$d = 24$	0.991		$layer = 3$	$d = 36$	0.975
	$head = 1$	$d = 12$	0.958		$head = 8$	$d = 24$	0.997		$head = 4$	$d = 32$	0.990		$head = 8$	$d = 48$	0.973
		$d = 24$	0.957			$d = 36$	0.956			$d = 36$	0.990			$d = 64$	0.980
	$layer = 2$	$d = 8$	0.975		$layer = 2$	$d = 12$	0.997		$layer = 2$	$d = 24$	0.990		$layer = 4$	$d = 36$	0.960
	$head = 2$	$d = 12$	0.969		$head = 6$	$d = 24$	0.996		$head = 6$	$d = 32$	0.985		$head = 8$	$d = 48$	0.975
		$d = 24$	0.969			$d = 36$	0.997			$d = 36$	0.981			$d = 64$	0.979

Dataset	AL = 53.00		*Acc*	Dataset	AL = 115.40		*Acc*	Dataset	AL = 121.11		*Acc*	Dataset	AL = 121.11		*Acc*
IMDB5	$layer = 1$	$d = 12$	0.958	IMDB6	$layer = 1$	$d = 24$	0.992	IMDB8	$layer = 1$	$d = 24$	0.993	IMDB9	$layer = 1$	$d = 24$	0.990
	$head = 2$	$d = 16$	0.956		$head = 5$	$d = 28$	0.993		$head = 7$	$d = 30$	0.994		$head = 7$	$d = 30$	0.993
		$d = 20$	0.945			$d = 32$	0.993			$d = 36$	0.995			$d = 36$	0.991
	$layer = 1$	$d = 12$	0.972		$layer = 1$	$d = 24$	0.994		$layer = 1$	$d = 24$	0.995		$layer = 1$	$d = 24$	**0.995**
	$head = 4$	$d = 16$	**0.974**		$head = 7$	$d = 28$	**0.995**		$head = 8$	$d = 30$	**0.997**		$head = 8$	$d = 30$	**0.995**
		$d = 20$	0.972			$d = 32$	**0.995**			$d = 36$	0.996			$d = 36$	0.992
	$layer = 1$	$d = 12$	0.944		$layer = 2$	$d = 24$	0.982		$layer = 2$	$d = 24$	0.993		$layer = 2$	$d = 24$	0.993
	$head = 6$	$d = 16$	0.942		$head = 7$	$d = 28$	0.978		$head = 6$	$d = 30$	0.992		$head = 6$	$d = 30$	0.989
		$d = 20$	0.946			$d = 32$	0.979			$d = 36$	0.995			$d = 36$	0.988
	$layer = 2$	$d = 12$	0.943		$layer = 1$	$d = 24$	0.983		$layer = 2$	$d = 24$	0.994		$layer = 2$	$d = 24$	0.987
	$head = 6$	$d = 16$	0.947		$head = 8$	$d = 28$	0.987		$head = 7$	$d = 30$	0.992		$head = 8$	$d = 30$	0.984
		$d = 20$	0.948			$d = 32$	0.991			$d = 36$	0.993			$d = 36$	0.986
	$layer = 1$	$d = 12$	0.949		$layer = 2$	$d = 24$	0.988		$layer = 2$	$d = 24$	0.992		$layer = 3$	$d = 24$	0.985
	$head = 8$	$d = 16$	0.954		$head = 7$	$d = 28$	0.990		$head = 8$	$d = 30$	0.994		$head = 8$	$d = 30$	0.976
		$d = 20$	0.952			$d = 32$	0.989			$d = 36$	0.990			$d = 36$	0.973

Dataset	AL = 121.11		*Acc*	Dataset	AL = 4.20		*Acc*	Dataset	AL = 4.20		*Acc*	Dataset	AL = 8.11		*Acc*
IMDB10	$layer = 1$	$d = 24$	0.980	Twitter 1bpw	$layer = 1$	$d = 8$	0.764	Twitter 3bpw	$layer = 1$	$d = 8$	0.967	Movie 5bpw	$layer = 1$	$d = 8$	0.973
	$head = 7$	$d = 30$	0.982		$head = 1$	$d = 12$	0.765		$head = 1$	$d = 12$	0.970		$head = 1$	$d = 12$	0.971
		$d = 36$	0.981			$d = 24$	0.762			$d = 24$	0.969			$d = 24$	0.969
	$layer = 1$	$d = 24$	0.979		$layer = 1$	$d = 8$	**0.767**		$layer = 1$	$d = 8$	0.971		$layer = 1$	$d = 8$	**0.975**
	$head = 8$	$d = 30$	0.982		$head = 2$	$d = 12$	0.765		$head = 2$	$d = 12$	**0.974**		$head = 2$	$d = 12$	0.973
		$d = 36$	0.980			$d = 24$	0.766			$d = 24$	0.970			$d = 24$	0.972
	$layer = 2$	$d = 24$	0.970		$layer = 1$	$d = 8$	0.766		$layer = 1$	$d = 8$	0.968		$layer = 1$	$d = 8$	0.972
	$head = 6$	$d = 30$	0.974		$head = 4$	$d = 12$	0.762		$head = 4$	$d = 12$	0.973		$head = 4$	$d = 12$	0.971
		$d = 36$	0.976			$d = 24$	0.759			$d = 24$	0.970			$d = 24$	0.967
	$layer = 2$	$d = 24$	0.979		$layer = 2$	$d = 8$	0.762		$layer = 2$	$d = 8$	0.968		$layer = 2$	$d = 8$	0.969
	$head = 8$	$d = 30$	**0.983**		$head = 1$	$d = 12$	0.761		$head = 1$	$d = 12$	0.967		$head = 1$	$d = 12$	0.972
		$d = 36$	0.977			$d = 24$	0.765			$d = 24$	0.969			$d = 24$	0.970
	$layer = 3$	$d = 24$	0.975		$layer = 2$	$d = 8$	**0.767**		$layer = 2$	$d = 8$	0.971		$layer = 2$	$d = 8$	0.973
	$head = 8$	$d = 30$	0.973		$head = 2$	$d = 12$	0.765		$head = 2$	$d = 12$	0.969		$head = 2$	$d = 12$	0.971
		$d = 36$	0.970			$d = 24$	0.766			$d = 24$	0.970			$d = 24$	0.972

- LS_CNN [18]: CNN-based text steganalysis that uses convolution kernels to capture complex dependencies between words;
- R_BI_C [21]: A text steganalysis method combining BiLSTM and CNN;
- BERT_BiLSTM_Att [22]: A BERT and BiLSTM combined method;
- GCN [23]: The text steganalysis method that can use other word information to extract and update dependencies;
- KD&FI [24]: A transfer learning-based text steganalysis method, it regards BERT as the teacher model and BiLSTM and CNN as the student models.

Considering language pre-trained models such as BERT [36], GPT-2 [39], T5 [40], and so on, cost huge of time and space, V-A3tS randomly initializes vector: m is set to 300, the batch size is set to 32, the linear layer dimension d_{ff} is set to 300, the dropout is set to 0.1, and the learning rate of long text is initialized to 1e-3, and the learning rate of short texts and mixed texts is initialized to 5e-4. Texts in each batch are supplemented to the maximum sentence length by 0. d, $head$, and $layer$ are calculated by Eq. (10). The learning algorithm is the mini-batch gradient descent of Adam [41]. 60% samples of the dataset are randomly selected for training, 20% for verifying, and the other 20% for testing.

Experiments are completed on one NVIDIA GeForce RTX 3080 GPU card, the experimental graphs are made in OriginPro 8.5. Methods performance indices include *Acc* (accuracy rate), P (precision rate), R (recall rate), and Time (evaluate the training speed). The indices are as follows.

$$\begin{aligned} Acc &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \\ P &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ R &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{TN}} \\ \mathrm{Time} &= \mathrm{T}_{\mathrm{end}} - \mathrm{T}_{\mathrm{start}}, \end{aligned} \tag{11}$$

where, TP is the number of correctly predicted cover samples, FP is the number of mispredicted stego samples, TN is the number of correctly predicted stego samples, and FN is the number of incorrectly predicted cover samples. $\mathrm{T}_{\mathrm{end}}$ and $\mathrm{T}_{\mathrm{start}}$ represent the timestamps of the end and the start of training.

Table 4
Nine steganalysis methods "accuracy" comparison of long texts. ×is the network that does not work. Bold means the best performance.

Methods / Dataset	Index	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB1	*Acc*	0.641	0.883	0.783	0.912	0.954	×	0.966	×	**0.983**
	P	0.584	0.819	0.712	0.954	0.966	×	0.931	×	**0.968**
	R	0.983	0.983	0.950	0.866	0.942	×	**1.000**	×	**1.000**
IMDB2	*Acc*	0.918	0.821	0.910	0.777	0.935	0.996	0.956	0.863	**0.998**
	P	0.899	0.761	0.869	0.749	0.892	**0.996**	0.921	0.847	**0.996**
	R	0.941	0.934	0.966	0.831	0.991	0.996	**1.000**	0.886	**1.000**
IMDB3	*Acc*	0.920	0.911	0.900	0.832	0.955	0.983	0.941	0.848	**0.995**
	P	0.908	0.947	0.854	0.870	0.955	0.986	0.914	0.857	**0.994**
	R	0.934	0.870	0.966	0.782	0.956	0.979	0.974	0.835	**0.997**
IMDB4	*Acc*	0.936	0.928	0.773	0.940	0.964	0.980	0.966	0.827	**0.981**
	P	0.948	0.997	0.704	1.000	0.960	0.964	0.953	0.864	**0.990**
	R	0.922	0.858	0.941	0.880	0.968	**0.998**	0.983	0.776	0.971
IMDB5	*Acc*	0.901	0.933	0.852	0.805	0.935	**0.984**	0.959	0.792	0.974
	P	0.856	0.934	0.829	0.743	0.913	**0.990**	0.947	0.745	0.968
	R	0.964	0.931	0.886	0.932	0.960	0.977	0.974	0.886	**0.981**
IMDB6	*Acc*	0.912	0.918	0.850	0.823	0.960	0.987	0.930	0.856	**0.995**
	P	0.916	0.919	0.802	0.867	0.935	0.988	0.876	0.889	**0.994**
	R	0.906	0.918	0.930	0.762	0.988	0.986	0.994	0.813	**0.996**
IMDB7	*Acc*	0.784	0.903	0.787	0.850	0.968	0.988	0.960	0.847	**0.996**
	P	1.000	0.952	0.709	0.843	0.960	0.983	0.942	0.880	**0.997**
	R	0.568	0.849	0.973	0.861	0.976	0.992	0.979	0.802	**0.995**

Table 5
Nine steganalysis methods "parameter size" comparison of long texts. ×is the network that does not work. (Unit is 1×10^4).

Methods	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB1	\	171.57	1026.65	316.68	681.48	×	221.70	×	198.24
IMDB2	\	515.56	1370.64	1122.90	1487.70	10950.60	790.17	1920.75	601.35
IMDB3	\	1336.91	2191.99	3047.94	3412.74	10950.60	2195.16	1920.75	1563.87
IMDB4	\	824.83	1679.91	1847.76	2212.56	10950.60	1311.48	1920.75	963.78
IMDB5	\	315.95	1171.03	655.08	1019.88	10950.60	455.31	1920.75	367.44
IMDB6	\	1266.02	2121.10	2881.80	3246.60	10950.60	2084.85	1920.75	1480.80
IMDB7	\	1047.52	1902.60	2369.70	2734.50	10950.60	1693.53	1920.75	1224.75

Table 6
Nine steganalysis methods "training time" comparison of long texts. ×is the network that does not work. Bold means the shortest time cost. (Unit is second(s)).

Methods	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB1	\	6.11	16.76	5.68	9.24	×	4.22	×	**3.12**
IMDB2	\	23.16	97.64	24.85	42.69	103.12	42.99	497.52	**14.29**
IMDB3	\	70.79	390.71	76.33	149.41	519.35	278.56	2478.78	**45.66**
IMDB4	\	40.79	197.83	26.00	69.40	124.98	345.87	1152.23	**31.18**
IMDB5	\	11.85	47.53	10.04	26.73	28.31	35.22	127.28	**7.87**
IMDB6	\	68.25	372.69	66.81	147.23	519.57	257.85	2499.59	**42.81**
IMDB7	\	48.75	299.35	42.07	104.51	350.44	526.96	1998.71	**25.62**

4.2. Verification experiment

In this subsection, we will show the specific parameters and corresponding performances of V-A3tS method on different datasets, they are shown in Fig. 3.

According to Table 3, the parameter settings in Section 3.2 of this paper can bring better detection performance to V-A3tS model. For example, when $layer = 1, head = 2, d = 8$, the best Acc is 0.983 in IMDB1, when $layer = 1, head = 6, d = 24$, the best Acc is 0.998 in IMDB2, and so on. In other words, the variable parameters modified by Eq. (10) can lead to better or best performance in distinct datasets.

It can also be discovered that the parameter setting is closer to BERT [36], i.e. $layer = 12, head = 12, d = 24$. The overall effect tends to decline. For example, in IMDB5, when $layer = 1, head = 4, d = 16$, the Acc is 0.974, however, when $layer = 2, head = 6$, the Acc does not exceed 0.95; in IMDB9, detection performance are trending down with increasing parameters. The phenomenon may be intuitively different from the better detection performance in BERT-based models [22,24]. This is because, in V-A3tS method, the vocab of BERT, vector representation, and dimension are not used. If large-scale network parameters are set, it will cause over-fitting or even not work.

4.3. Comparative experiment of long texts

4.3.1. Steganalysis accuracy

Table 4 and Fig. 2 demonstrate that V-A3tS method achieves higher detection accuracy than existing methods of long texts. When the dataset is IMDB4, the detection accuracy of V-A3tS is 4.5% higher than that of FCN [16]. Due to its simple structure, FCN [16] is not sensitive enough to capture the distribution differences of high-dimensional statistical features, so its effect is slightly worse than other deep learning methods. The detection accuracy of V-A3tS is 5.3% and 20.8% higher than that of time series models such as TS_BiRNN [17], BiLSTM_Dense [20]. For the time series models, although they can obtain long-term dependence, the dependence degree between words will weaken with the increase of the number of separated words, so the ability to detect long steganography text is limited. The detection accuracy of V-A3tS is 4.1% and 1.5% higher than that of convolution models such as LS_CNN [18], GCN [23]. Because the convolution kernel and shared matrix dimensions are small, the convolution models are difficult to obtain strong dependence of distant words, so the performance of the proposed method is slightly lower than that of the proposed method in detection accuracy.

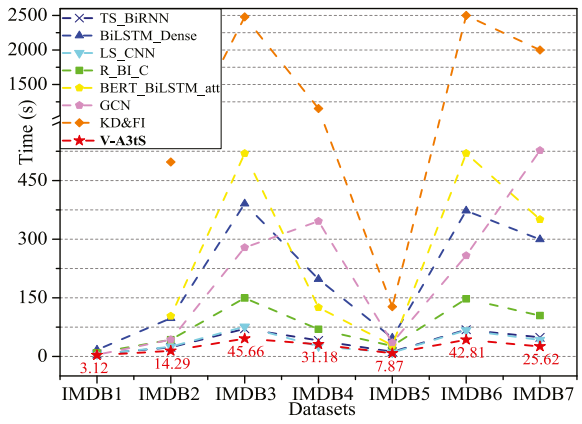


Fig. 4. Nine steganalysis methods "training time" comparison of long texts. The horizontal axis represents the datasets, and the vertical axis represents the training time.

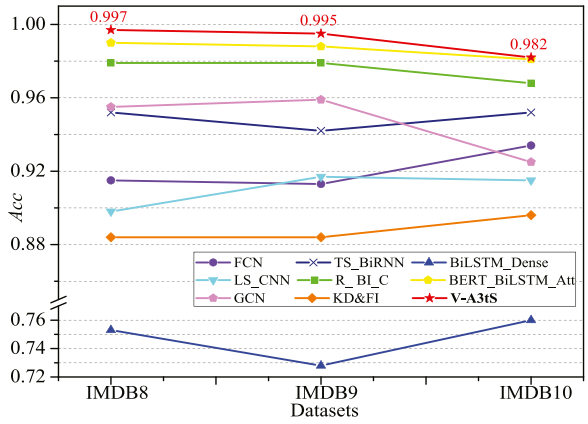


Fig. 5. Nine steganalysis methods "accuracy" comparison of mixed texts. The horizontal axis represents the datasets, and the vertical axis represents the detection accuracy.

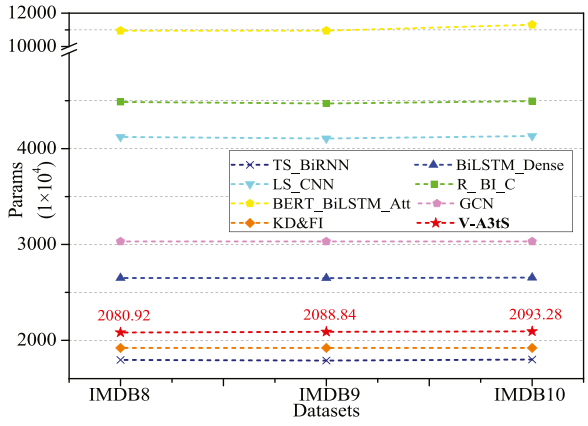


Fig. 6. Nine steganalysis methods "parameter size" comparison of mixed texts. The horizontal axis represents the datasets, and the vertical axis represents the parameter size.

4.3.2. Parameter size

Since FCN [16] does not belong to the four types of methods in Section 2, the parameter size and training time are not compared with FCN. Table 5 and Fig. 3 demonstrate that V-A3tS method has a smaller size of parameters than several other methods except for TS_BiRNN [17]. However, the detection accuracy of V-A3tS method is about ten percentage points higher than that of TS_BiRNN [17] of long text datasets.

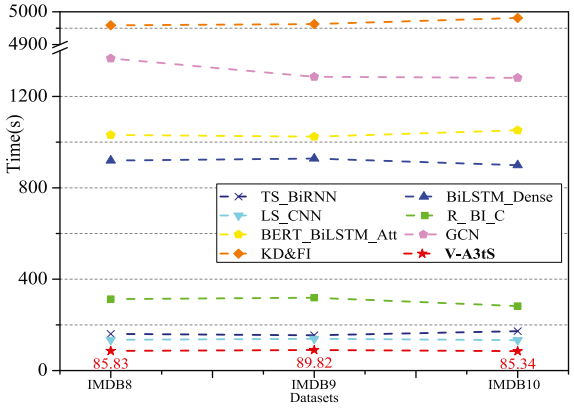


Fig. 7. Nine steganalysis methods "training time" comparison of mixed texts. The horizontal axis represents the datasets, and the vertical axis represents the training time.

4.3.3. Training time

Table 6 and Fig. 4 demonstrate that V-A3tS method takes the least time for training. When the dataset is IMDB3, the training time of V-A3tS method is 1.55 times and 8.56 times shorter than that of time series models such as TS_BiRNN [17] and BiLSTM_dense [20]. Because the time series models needs to be processed word by word, it cannot be processed in parallel, which leads to the time-consuming of model training. The training time of V-A3tS method is 1.67 times and 6.10 times shorter than that of convolution models such as LS_CNN [18] and GCN [23]. Although the convolution model can be processed in parallel, LS_CNN [18] sets a variety of convolution kernels to obtain the dependence of different distances. The process of GCN [23] updating word semantic expression according to the shared matrix also needs to consider the dependencies of nearby words, which increase the number of parameters of the model. It can also be reflected in the comparison of the parameter size in Table 5. Therefore, the result that the proposed method has shorter training time than the existing convolution model is reliable. The training time of V-A3tS method is 11.37 times and 54.29 times shorter than that of convolution models such as BERT_BiLSTM_Att [22] and KD&FI [24].

4.4. Comparative experiment of mixed texts

4.4.1. Steganalysis accuracy

It can be obtained from Table 7 and Fig. 5 that V-A3tS method can extract features of mixed texts effectively, and has a higher detection accuracy than existing methods.

4.4.2. Parameter size

It can be found from Table 8 and Fig. 6 that V-A3tS has a smaller amount of parameters, which is more than 5 times smaller than BERT_BiLSTM_Att [22] in mixed texts, which reduces unnecessary space overhead.

4.4.3. Training time

It is easily obtained from Table 9 and Fig. 7 that V-A3tS method has a lower training time in mixed texts. For example, when the dataset is IMDB10, the training time of V-A3tS method is 2.0 times, 10.5 times, 1.5 times, 3.3 times, 12.3 times, 15.0 times, and 58.3 times shorter than that of TS_BiRNN [17], BiLSTM_Dense [20], and other methods.

Table 7
Nine steganalysis methods "accuracy" comparison of mixed texts. Bold means the best performance.

Methods / Dataset	Index	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB8	*Acc*	0.915	0.952	0.753	0.898	0.979	0.990	0.955	0.884	**0.997**
	P	0.912	0.944	0.671	0.892	0.966	0.990	0.935	0.882	**0.997**
	R	0.920	0.962	0.987	0.906	0.992	0.991	0.980	0.886	**0.996**
IMDB9	*Acc*	0.913	0.942	0.728	0.917	0.979	0.988	0.959	0.884	**0.995**
	P	0.906	0.942	0.654	0.909	0.965	0.989	0.938	0.889	**0.992**
	R	0.921	0.930	0.970	0.927	0.995	0.987	0.983	0.878	**0.997**
IMDB10	*Acc*	0.934	0.952	0.760	0.915	0.968	0.981	0.925	0.896	**0.982**
	P	0.975	0.961	0.687	0.933	0.967	0.989	0.908	0.941	**0.992**
	R	0.892	0.942	0.957	0.893	0.969	**0.973**	0.943	0.845	0.971

Table 8
Nine steganalysis methods "parameter size" comparison of mixed texts. (Unit is 1×10^4).

Methods	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB8	\	1795.33	2650.40	4122.36	4487.16	10950.60	3031.77	1920.75	2080.92
IMDB9	\	1788.57	2648.48	4106.52	4471.32	10950.60	3031.77	1920.75	2088.84
IMDB10	\	1799.11	2654.19	4131.24	4496.04	11303.80	3031.71	1920.75	2093.28

Table 9
Nine steganalysis methods "training time" comparison of mixed texts. Bold means the shortest time cost. (Unit is second(s)).

Methods	[16]	[17]	[20]	[18]	[21]	[22]	[23]	[24]	V-A3tS
IMDB8	\	160.63	919.18	134.64	312.59	1031.51	1366.37	4958.88	**85.83**
IMDB9	\	154.26	928.37	139.05	318.93	1023.91	1285.82	4962.42	**89.82**
IMDB10	\	172.39	899.16	133.26	282.33	1051.89	1281.71	4981.13	**85.34**

Table 10
Ten steganalysis methods "accuracy" comparison of short texts. Bold means the best performance.

Movie / Stego	Payload	Index	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	*Acc*	0.514	0.577	0.785	0.797	0.800	0.798	**0.817**	0.803	0.783	**0.817**
	2bpw	*Acc*	0.560	0.713	0.829	0.868	0.872	0.879	0.878	0.843	0.851	**0.886**
	3bpw	*Acc*	0.610	0.840	0.863	0.897	0.890	0.907	0.915	0.880	0.896	**0.921**
	4bpw	*Acc*	0.623	0.909	0.909	0.947	0.945	0.951	0.953	0.930	0.934	**0.954**
	5bpw	*Acc*	0.636	0.909	0.945	0.967	0.965	0.974	**0.977**	0.949	0.966	0.976
Twitter / Stego	Payload	Index	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	*Acc*	0.509	0.538	0.723	0.756	0.738	0.764	0.759	0.736	0.765	**0.767**
	2bpw	*Acc*	0.542	0.544	0.876	0.914	0.893	0.928	0.931	0.932	0.883	**0.935**
	3bpw	*Acc*	0.547	0.577	0.941	0.968	0.945	0.969	0.969	0.970	0.949	**0.971**
	4bpw	*Acc*	0.650	0.729	0.975	0.985	0.980	0.994	0.993	0.987	0.979	**0.995**
	5bpw	*Acc*	0.702	0.850	0.975	0.991	0.981	0.990	0.992	0.992	0.988	**0.995**

Table 11
Ten steganalysis methods "parameter size" comparison of short texts.(Unit is 1×10^4).

Movie / Stego	Payload	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	\	\	\	91.11	946.19	128.10	492.90	84.81	1920.75	64.83
	2bpw	\	\	\	102.04	957.12	153.72	518.52	97.83	1920.75	77.64
	3bpw	\	\	\	118.04	973.12	191.22	556.02	120.93	1920.75	96.39
	4bpw	\	\	\	133.12	988.20	226.56	591.36	138.60	1920.75	114.06
	5bpw	\	\	\	152.24	1007.32	271.38	636.18	165.90	1920.75	136.47
Twitter / Stego	Payload	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	\	\	\	123.93	979.01	205.02	569.82	131.67	1920.75	103.29
	2bpw	\	\	\	127.00	982.08	212.22	577.02	133.95	1920.75	106.89
	3bpw	\	\	\	139.31	994.39	241.08	605.88	148.02	1920.75	121.32
	4bpw	\	\	\	157.39	1012.47	283.44	648.24	170.46	1920.75	142.50
	5bpw	\	\	\	179.04	1034.12	334.20	699.00	196.50	1920.75	167.88

4.5. Comparative experiment of short texts

4.5.1. Steganalysis accuracy

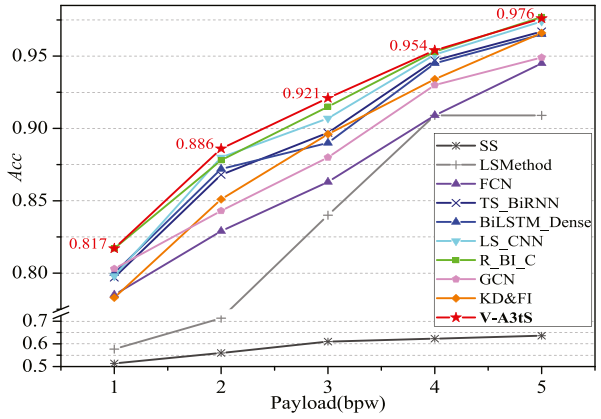
The results from Table 10 and Fig. 8 show that the proposed method has higher detection accuracy than existing related methods of short texts. Among them, when Payload=2bpw, the accuracy of V-A3tS is 32.6%, 17.3%, 5.7%, 1.8%, 1.4%, 0.7%, 0.8%, 4.3%, 4.3% and 3.5% higher than that of other methods on the Movie.

4.5.2. Parameter size

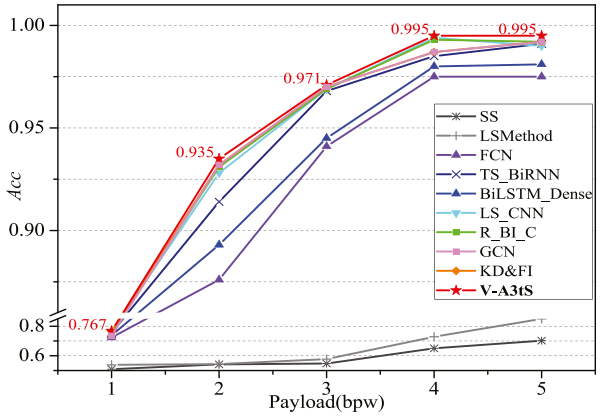
It is obtained from Table 11 and Fig. 9 that V-A3tS has a smaller number of parameters of short texts, ranging from 11.4 times to 29.6 times smaller than KD&FI [24].

Table 12
Ten steganalysis methods "training time" comparison of short texts. Bold means the best performance. (Unit is second(s)).

Movie Stego	Payload	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	\	\	\	46.38	58.60	30.70	56.29	24.83	1236.20	**21.51**
	2bpw	\	\	\	46.41	59.19	31.18	63.42	24.37	1240.33	**22.27**
	3bpw	\	\	\	47.27	63.76	32.40	66.36	25.71	1241.32	**23.44**
	4bpw	\	\	\	48.70	60.05	32.28	65.51	24.90	1272.32	**22.28**
	5bpw	\	\	\	49.82	61.94	32.45	66.93	25.84	1241.12	**23.59**
Twitter Stego	Payload	[33]	[34]	[16]	[17]	[20]	[18]	[21]	[23]	[24]	V-A3tS
Fang	1bpw	\	\	\	65.90	97.06	55.37	102.28	45.22	2744.34	**41.95**
	2bpw	\	\	\	69.59	102.17	57.77	106.12	44.36	2750.69	**41.63**
	3bpw	\	\	\	82.07	108.45	58.56	114.99	43.83	2748.77	**41.07**
	4bpw	\	\	\	85.93	110.37	58.41	115.23	44.01	2754.80	**40.67**
	5bpw	\	\	\	85.48	116.95	58.53	115.65	44.06	2755.76	**41.25**

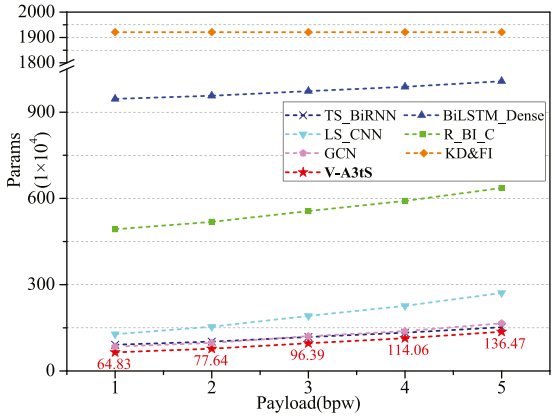


(a) The "accuracy" comparison of Movie

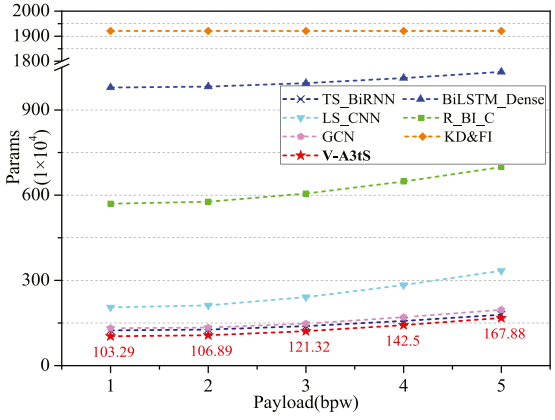


(b) The "accuracy" comparison of Twitter

Fig. 8. Ten steganalysis methods "accuracy" comparison of short texts. The horizontal axis represents the payloads, and the vertical axis represents the detection accuracy.



(a) The "parameter size" comparison of Movie



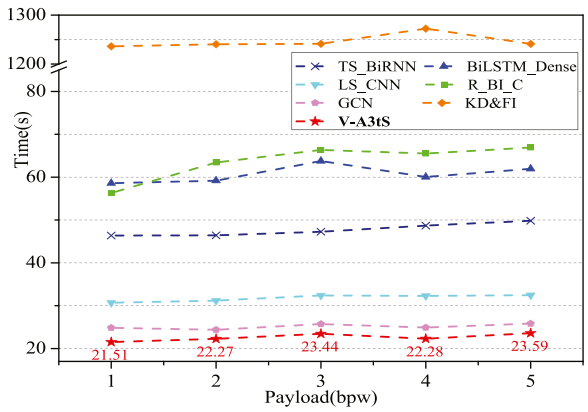
(b) The "parameter size" comparison of Twitter

Fig. 9. Ten steganalysis methods "parameter size" comparison of short texts. The horizontal axis represents the payloads, and the vertical axis represents the parameter size.

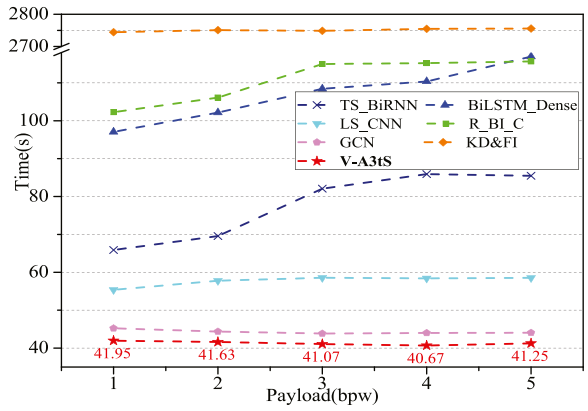
4.5.3. Training time

Since SS [33] and LSMethod [34] are traditional methods and their accuracy is significantly lower than those based on deep learning, we do not compare their parameter size and training time of theirs. The results from Table 12 and Fig. 10 show that V-A3tS method is more efficient than the existing related methods of short texts. When Payload=4bpw, the efficiency of V-A3tS method is 2.1 times, 2.7 times, 1.4 times, 2.83 times, 1.08 times and 67.7 times higher than that of TS_BiRNN [17], BiLSTM_Dense [20], LS_CNN [18], and other methods on the Twitter dataset, which greatly improves the training speed, avoiding unnecessary time cost.

The comparison experiments on the above three kinds of datasets illustrate that V-A3tS method can quickly and effectively detect stego texts with distinct lengths, and has higher detection accuracy and efficiency than existing methods. At the same time, V-A3tS method has a smaller amount of training parameters. And we also discovered that existing methods are more conducive to detecting short stego texts while existing methods perform poorly for long and mixed stego texts. This is because the syntactic structure of short and long texts is distinct, and the mechanism of extracting steganalysis features is also different. Therefore, the variable parameters-based method proposed in



(a) The "training time" comparison of Movie



(b) The "training time" comparison of Twitter

Fig. 10. Ten steganalysis methods "training time" comparison of short texts. The horizontal axis represents the payloads, and the vertical axis represents the training time.

this paper adapts to stego texts with distinct lengths and can effectively capture a moderate amount of dependency.

5. Conclusion

The existing text steganalysis based on deep learning is either too simple in design, which leads to further improvement of detection accuracy; or required fine-tuning of large-scale models, which leads to too long training time. It is difficult to satisfy both higher detection accuracy and lower training time. Therefore, this paper proposes V-A3tS method, a novel text steganalysis method with variable parameters of the multi-head self-attention mechanism by text length. V-A3tS method maps words into the semantic space with position information and extracts dependencies using the self-attention layer designed in this paper to adapt to distinct text lengths. Ultimately, the steganalysis features through the residual linear layer are input to the classifier for the detection of stego texts. Comparative experiments of long texts, mixed texts, and short texts indicate that V-A3tS method can effectively learn steganalysis features to improve the accuracy and efficiency of steganalysis with the smaller size of parameters.

Last but not least, features extracted based on large-scale pre-training models may have better detection effects, such as BERT, T5 and GPT-2, etc. How to make better use of large-scale pre-training models to greatly improve detection accuracy will be our next work.

Declaration of competing interest

The authors declare no conflict of interest.

Funding

This work was supported by the National Natural Science Foundation of China (grant numbers U21B2020, U1936216).

References

[1] Shannon Claude Elwood. Communication theory of secrecy systems. Bell Syst Tech J 1949;28(4):656–715.
[2] Yang Zhongliang, Guo Xiaoqing, Chen Ziming, et al. RNN-Stega: Linguistic steganography based on Recurrent Neural Networks. IEEE Trans Inf Forensics Secur 2019;14(5):1280–95.
[3] Fang Tina, Jaggi Martin, Argyraki Katerina. Generating steganographic text with LSTMs. 2017, arXiv preprint arXiv:1705.10742.
[4] Xu Guanshuo, Wu Hanzhou, Shi Yunqing. Structural design of Convolutional Neural Networks for steganalysis. IEEE Signal Process Lett 2016;23(5):708–12.
[5] Ye Jian, Ni Jiangqun, YangYi. Deep learning hierarchical representations for image steganalysis. IEEE Trans Inf Forensics Secur 2017;12(11):2545–57.
[6] Boroumand Mehdi, Chen Mo, Fridrich Jessica. Deep residual network for steganalysis of digital images. IEEE Trans Inf Forensics Secur 2019;14(5):1181–93.
[7] Zhang Ru, Zhu Feng, Liu Jianyi, et al. Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis. IEEE Trans Inf Forensics Secur 2019;15:1138–50.
[8] Dai Falcon Z, Cai Zheng. Towards near-imperceptible steganographic text. In: Proceedings of the 57th annual meeting of the association for computational linguistics. 2019, p. 4303–8.
[9] Yang Zhongliang, Zhang Siyu, Hu Zhiwen, et al. VAE-Stega: Linguistic steganography based on variational auto-encoder. IEEE Trans Inf Forensics Secur 2021;16:880–95.
[10] Zhang Siyu, Yang Zhongliang, Yang Jinshuai, et al. Provably secure generative linguistic steganography. In: Proceeding of the international joint conference on natural language processing (ACL-IJCNLP). 2021.
[11] Zhou Xuejing, Peng Wanli, Yang Boya, et al. Linguistic steganography based on adaptive probability distribution. IEEE Trans Dependable Secure Comput 2021;1.
[12] Zhang Siyu, Yang Zhongliang, Yang Jinshuai, et al. Linguistic steganography: From symbolic space to semantic space. IEEE Signal Process Lett 2021;28:11–5.
[13] Ziegler Zachary M, Deng Yuntian, Rush Alexander M. Neural linguistic steganography. 2019, arXiv:1909.01496.
[14] Shen Jiaming, Ji Heng, Han Jiawei. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In: Proceeding of empirical methods natural lang. process. 2020, p. 303–13.
[15] Mahato S, Yadav DK, Khan DA. A minesweeper game-based steganography scheme. J Inf Secur Appl 2017;32:1–14.
[16] Yang Zhongliang, Huang Yongfeng, Zhang Yujin. A fast and efficient text steganalysis method. IEEE Signal Process Lett 2019;26(4):627–31.
[17] Yang Zhongliang, Wang Ke, Li Jian, et al. TS-RNN: Text steganalysis based on Recurrent Neural Networks. IEEE Signal Process Lett 2019;26(12):1743–7.
[18] Wen Juan, Zhou Xuejing, Zhong Ping, et al. Convolutional neural network based text steganalysis. IEEE Signal Process Lett 2019;26(3):460–4.
[19] Yang Zhongliang, Huang Yongfeng, Zhang Yujin. TS-CSW: text steganalysis and hidden capacity estimation based on convolutional sliding windows. Multimedia Tools Appl 2020;79(25–26):18293–316.
[20] Yang Hao, Bao YongJian, Yang Zhongliang, et al. Linguistic steganalysis via densely connected LSTM with feature pyramid. In: Proceeding of the ACM workshop on information hiding and multimedia security. IWDW, 2020, p. 5–10.
[21] Niu Yan, Wen Juan, Zhong Ping, et al. A hybrid R-BILSTM-C neural network based text steganalysis. IEEE Signal Process Lett 2019;26(12):1907–11.
[22] Zou Jiajun, Yang Zhongliang, Zhang Siyu, et al. High-performance linguistic steganalysis, capacity estimation and steganographic positioning. In: Proceeding of the international workshop on digital watermarking. 2021, p. 80–93.
[23] Wu Hanzhou, Yi Biao, Ding Feng, et al. Linguistic steganalysis with graph neural networks. IEEE Signal Process Lett 2021;28:558–62.
[24] Peng Wanli, Zhang Jinyu, Xue Yiming, et al. Real-time text steganalysis based on multi-stage transfer learning. IEEE Signal Process Lett 2021;28:1510–4.
[25] Yang Jinshuai, Yang Zhongliang, Zhang Siyu, et al. SeSy: Linguistic steganalysis framework integrating semantic and syntactic features. IEEE Signal Process Lett 2022;29:31–5.
[26] Xue Yiming, Yang Boya, Deng Yaqian, et al. Domain adaptational text steganalysis based on transductive learning. In: Proceeding of the ACM workshop on information hiding and multimedia security (IH & MMSec). 2022.
[27] Wen Juan, Zhang Ziwei, Yang Yu, et al. Few-shot text steganalysis based on attentional meta-learner. In: Proceeding of the ACM workshop on information hiding and multimedia security (IH & MMSec). 2022.
[28] Xiang Lingyun, Sun Xingming, Luo Gang, et al. Linguistic steganalysis using the features derived from synonym frequency. Multimedia Tools Appl 2014;71(3):1893–911.
[29] Hsieh Ku-Sung, Wang Chung-Ming. Constructive image steganography using example-based weighted color transfer. J Inf Secur Appl 2022;65(103126).

[30] Taskiran Cuneyt M, Topkara Umut, Topkara Mercan, et al. Attacks on lexical natural language steganography systems. In: Proceeding of the security, steganography, and watermarking of multimedia contents VIII. vol. 6072, 2006, p. 1–9.
[31] Xiang Lingyun, Sun Xingming, Luo Gang, et al. Linguistic steganalysis using the features derived from synonym frequency. Multimedia Tools Appl 2014;71(3):1893–911.
[32] Yang Yu, Lei Min, Wang Jianfeng, et al. A SVM based text steganalysis algorithm for spacing coding. China Commun 2014;11(1):108–13.
[33] Yang Hao, Cao Xianbin. Linguistic steganalysis based on meta features and immune mechanism. Chin J Electron 2010;19(4):661–6.
[34] Meng Peng, Hang Liusheng, Yang Wei, et al. Linguistic steganography detection algorithm using statistical language model. In: Proceeding of the information technology and computer science. 2009, p. 540–3.
[35] Vaswani Ashish, Shazeer Noam, Parmar Niki, et al. Attention is all you need. 2017, arXiv preprint arXiv:1706.03762.
[36] Devlin Jacob, Chang Mingwei, Lee Kenton, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018, arXiv preprint arXiv:1810.04805.
[37] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition. In: Proceeding of the IEEE conference on computer vision and pattern recognition. CVPR, 2016, p. 770–8.
[38] Maas Andrew L, Daly Raymond E, Pham Peter T, et al. Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. 2011, p. 142–50.
[39] Radford Alec, Wu Jeffrey, Child Rewon, et al. Language models are unsupervised multitask learners. OpenAI Blog 2019;1(8):9.
[40] Raffel Colin, Shazeer Noam, Roberts Adam, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019, p. 1–65, arXiv preprint arXiv:1910.10683.
[41] Kingma Diederik P, Ba Jimmy Lei. ADAM: A method for stochastic optimization. 2014, arXiv:1412.6980.