DNA Steganalysis Using Deep Recurrent Neural Networks

Ho Bae [1], Byunghan Lee [2, 3], Sunyoung Kwon [2, 4] and Sungroh Yoon [1, 2, 5,*]

[1] *Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul 08826, Korea*
[2] *Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea*
[3] *Electronic and IT Media Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea*
[4] *Clova AI Research, NAVER Corp., Seongnam 13561, Korea*
[5] *ASRI and INMC, Seoul National University, Seoul 08826, Korea*
E-mail: sryoon@snu.ac.kr

Recent advances in next-generation sequencing technologies have facilitated the use of deoxyribonucleic acid (DNA) as a novel covert channels in steganography. There are various methods that exist in other domains to detect hidden messages in conventional covert channels. However, they have not been applied to DNA steganography. The current most common detection approaches, namely frequency analysis-based methods, often overlook important signals when directly applied to DNA steganography because those methods depend on the distribution of the number of sequence characters. To address this limitation, we propose a general sequence learning-based DNA steganalysis framework. The proposed approach learns the intrinsic distribution of coding and non-coding sequences and detects hidden messages by exploiting distribution variations after hiding these messages. Using deep recurrent neural networks (RNNs), our framework identifies the distribution variations by using the classification score to predict whether a sequence is to be a coding or non-coding sequence. We compare our proposed method to various existing methods and biological sequence analysis methods implemented on top of our framework. According to our experimental results, our approach delivers a robust detection performance compared to other tools.

Keywords: Deep recurrent neural network, DNA steganography, DNA steganalysis, DNA watermarking

1. Introduction

Steganography serves to conceal the existence and content of messages in media using various techniques, including changing the pixels in an image[1]. Generally, steganography is used to achieve two main goals. On the one hand, it is used as digital watermarking to protect intellectual property. On the other hand, it is used as a covert approach to communicating without the possibility of detection by unintended observers. In contrast, steganalysis is the study of detecting hidden messages. Steganalysis also has two main goals, which are detection and decryption of hidden messages[1,2].

Among the various media employed to hide information, deoxyribonucleic acid (DNA) is appealing owing to its chemical stability and, thus is a suitable candidates as a carrier of

concealed information. As a storage medium, DNA has the capacity to store large amounts of data that exceed the capacity of current storage media[3]. For instance, a gram of DNA contains approximately 10^{21} DNA bases (108 terabytes), which indicates that only a few grams of DNA can store all information available[4]. In addition, with the advent of next-generation sequencing, individual genotyping has become affordable[5], and DNA in turn has become an appealing covert channels.

To hide information in a DNA sequence, steganography methods require that a reference target sequence and a message to be hidden[6]. A naïve example of a substitution-based method for watermarking that exploits the preservation of amino acids is shown in Fig. 1 (see the caption for details). The hiding space of this method is restricted to exon regions using a complementary pair that does not interfere with protein translation. However, most DNA steganography methods are designed without considering the hiding spaces, and they change a sequence into a binary format utilizing well-known encryption techniques.

In this regard, Clelland et al.[7], first proposed DNA steganography that utilized the microdot technique. Yachie et al.[8], demonstrated that living organisms can be used as data storage media by inserting artificial DNA into artificial genomes and using a substitution cipher coding scheme. This technique is reproducible and successfully inserts four watermarks into the cell of a living organism[9]. Several other encoding schemes have been proposed [10,11]. The DNA-Crypt coding scheme [12] translates a message into 5-bit sequences, and the ASCII coding scheme[13] translates words into their ASCII representation, converts them from decimals to binary, and then replaces 00 with adenine (`A`), 01 with cytosine (`C`), 10 with guanine (`G`), and 11 with thymine (`T`).

With the recent advancements with respect to steganography methods, various steganalysis studies have been conducted using traditional storage media. Detection techniques that are based on statistical analysis, neural networks, and genetic algorithms[14] have been developed for common covert objects such as digital images, video, and audio. For example, Bennett[1] exploits letter frequency, word frequency, grammar style, semantic continuity, and logical methodologies. However, these conventional steganalysis methods have not been applied to DNA steganography.

In this paper, we show that conventional steganalysis methods are not directly applicable to DNA steganography. Currently, the most commonly employed detection schemes, i.e., a statistical hypothesis testing methods, are limited with respect to the number of input queries in order to estimate distribution to perform statistical test[15]. To overcome the limitations of these existing methods, we propose a DNA steganalysis method based on learning the internal structure of unmodified genome sequences (*i.e.*, intron and exon modeling[16,17]) using deep recurrent neural networks (RNNs). The RNN-based classifier is used to identify modified genome sequences. In addition, we enhance our proposed model using unsupervised pre-training of a sequence-to-sequence autoencoder in order to overcome the restriction of the robustness of detection performance. Finally, we compare our proposed method to various machine learning-based classifiers and biological sequence analysis methods that were implemented on top of our framework.

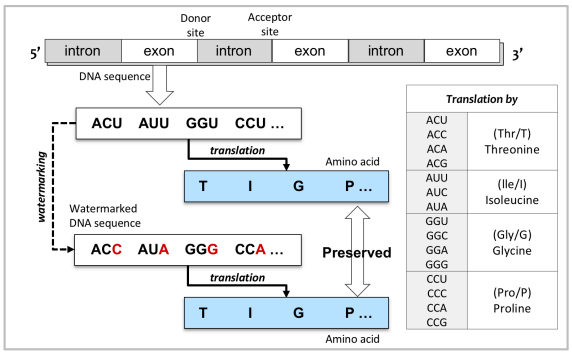

Fig. 1. DNA hiding scheme using synonymous codons. A watermark is a scheme used to deter unauthorized dissemination by marking hidden symbols or texts. For the conservation of amino acids, DNA watermarking can be changed to one of the synonymous codons.

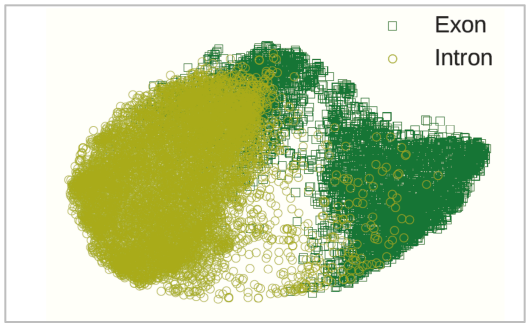

Fig. 2. Learned representation of DNA sequences. The learned representations for each coding and non-coding region projected into a two-dimensional (2-D) space using t-SNE.[18] The representation is based on sequence-to-sequence learning using an autoencoder and stacked RNNs.

2. Background

We use the standard terminology of information hiding[19] to provide a brief explanation of the related background. For example, two hypothetical parties, (i.e., a sender and a receiver) wish to exchange genetically modified organisms (GMOs) protected by patents. A third party detects watermark sequence from the GMOs for unauthorized use. Both the sender and receiver use the random oracle[20] model, which posits existing steganography schemes, to embed their watermark message, and the third party uses our proposed model to detect the watermarked signal. A random oracle model posits the randomly chosen function H, which can be evaluated only by querying the oracle that returns $H(m)$ given input m.

2.1. *Notations*

The notations used in this paper are as follows: $\mathbf{D} = \{D_1, \cdots, D_n\}$ is a set of DNA sequences of n species; $\hat{\mathbf{D}} = \{\hat{D}_1, \cdots, \hat{D}_n\}$ is a set of DNA sequences of n species and the hidden messages are embedded for some species $\hat{D}_i$; $m \in \{\texttt{A,C,G,T}\}^\ell$ is the input sequence where ℓ is the length of the input sequence; $\hat{m} \in \{\texttt{A,C,G,T}\}^\ell$ is the encrypted value of m where ℓ is the length of the encrypted sequence; E is an encryption function, which takes input m and returns the encrypted sequence $E(m) \rightarrow \hat{m}$; $\mathbf{M}_{D_i}$ is a trained model that takes target species D_i as training input; $\overline{y}$ is an averaged output score y; $\hat{y}$ is a probability output given by the trained model $\mathbf{M}_{D_i}(\hat{m}) \rightarrow \hat{y}$ given input $\hat{m}$, where $\hat{m} \in \hat{D}_i$; $\mathcal{A}$ is a probabilistic polynomial-time adversary. The adversary[21] is an attacker that queries messages to the oracle model; ϵ is the standard deviation value of score y.

2.2. *Hiding Messages*

The hiding positions of a DNA sequence segment are limited compared to those of the covert channel because the sequences are carried over after the translation and transcription processes in the exon region. For example, assume that `ACGGTTCCAATGC` is a reference sequence, and

01001100 is the message to be hidden. The reference sequence is then translated according to any coding schemes. In this example, we apply the DNA-crypt coding scheme[12], which converts the DNA sequence to binary replacing `A` with 00, `C` with 01, `G` with 10, and `T` with 11. The reference sequence is then translated to 000110101111010100000111001 and divided into key bits that are defined by the sender and receiver. Assume that the length of the key is 3, the reference sequence can be expressed as 000, 110, 101, 111, 010, 100, 001, 110, 01, and the message is concealed at the first position. The DNA sequence with the concealed messages are then represented as 0000, 1110, 0101, 0111, 1010, 1100, 0001, 0110, 01. Finally, the sender transmits the transformed DNA sequence of `AATGCCCTGGTAACCG`. The recipient can extract the hidden message using the pre-defined key.

2.3. *Determination of Message-Hiding Regions*

Genomic sequence regions (i.e., exons and introns) are utilized depending on whether the task is data storage or transport. Intron regions are suitable for transportation since they are not transcribed and are removed by splicing[22,23] during transcription. This property of introns provides large sequence space for concealing data, creating potential covert channels. In contrast, data storage (watermarking) requires data to be resistant to degradation or truncation. Exons are a suitable candidate for storage because underlying DNA sequence is conserved after the translation and transcription processes[24]. These two components of internal structure components in eukaryote genes are involved in DNA steganography as the payload (watermarking) or carrier (covert channels). Fig. 2 shows the learned representations of introns and exons which are calculated by softmax function. The softmax function reduces the outputs of intron and exons to range between 0 and 1. The 2D projection position of introns and exons will change if hidden messages are embedded without considering shared patterns between the genetic components (e.g., complementary pair rules). Thus, the construction of a classification model to enable a clear separation axis of these shared patterns is an important factor in the detection of hidden messages.

3. Methods

Our proposed method uses RNNs[25] to detect hidden messages in DNA. Fig. 3 shows our proposed steganalysis pipeline. The pipeline comprises of training and detection phases. In the model training phase, the model learns the distribution of unmodified genome sequences that distinguishes between introns and exons (see Section 3.2 for the model architecture). In the detection phase, we obtain a prediction score exhibiting the distribution of introns and exons. By exploiting the obtained prediction score, we formulate a detection principle. The details of the detection principle are described in Section 3.1.

3.1. *Proposed DNA Steganalysis Principle*

The security of the random oracle is based on an *experiment* E involving an adversary $\mathcal{A}$, as well as $\mathcal{A}$'s indistinguishability of the encryption. Assume that we have the random oracle that acts like a current steganography scheme S with only a negligible success probability.

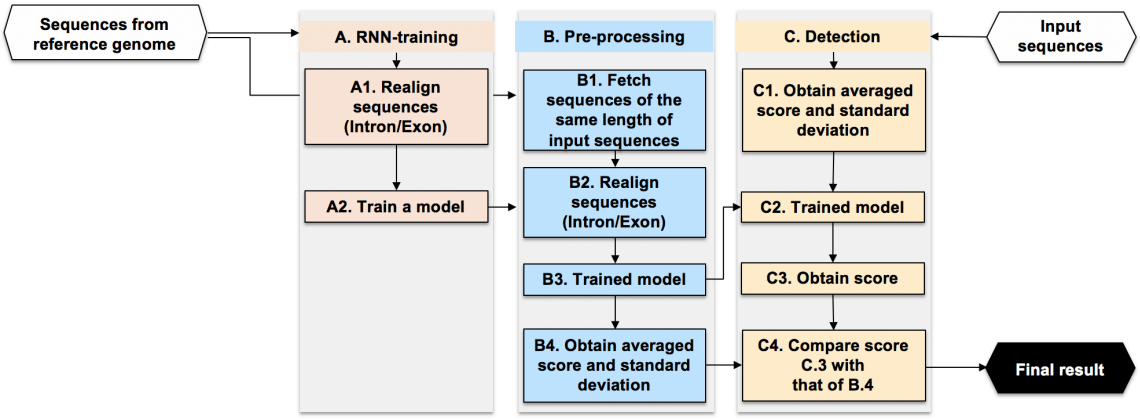

Fig. 3. Flowchart of proposed DNA steganalysis pipeline.

The experiment E can be defined for any encryption scheme S over message space $\mathbf{D}$ and for adversary $\mathcal{A}$. We describe the proposed method to detect hidden messages using the random oracle. For the E, the random oracle chooses a random steganography scheme S. Scheme S modifies or extends the process of mapping a sequence with length n input to a sequence with length ℓ with a random sequence as the output. The process of mapping sequences can be considered as a table that indicates for each possible input m the corresponding output value $\hat{m}$. With chosen scheme S, $\mathcal{A}$ chooses a pair of sequences $m_0, m_1 \in D_i$. The random oracle which posits the scheme S selects a bit $b \in \{0, 1\}$ and sends encrypted message $S(m_b) \rightarrow \hat{m}$ to the adversary. The adversary outputs a bit b'. Finally, the output of the E is defined as 1 if $b' = b$, and 0 otherwise. $\mathcal{A}$ succeeds in the E in the case of distinguishing m_b. Our methodology using E is described as follows:

(i) We construct M_{D_i} (Fig. 3-A) that runs on a random oracle where selected species $D_i \in \mathbf{D}$. Note that a model M can be based on any classification model, but the key to select a model is to reduce the standard deviation. Our proposed model M is described in Section 3.2.
(ii) $\mathcal{A}$ computes y (Fig. 3-B4) using $M_{D_i}(m)$ given $m \in D_i$.
(iii) $\mathcal{A}$ computes the standard deviation ϵ of y (Fig. 3-B).
(iv) $\mathcal{A}$ computes $\hat{y}$ (Fig. 3-C3) using $M_{D_i}(\hat{m})$ given $\hat{m} \in \hat{D}_i$.
(v) $\hat{m}$ is successfully detected (Fig. 3-C4) if

$$|\overline{y} - \hat{y}| > \epsilon. \tag{1}$$

This gives two independent scores y and $\hat{y}$ from M_{D_i}. The score y will have the same range of the unmodified genome sequences whereas the score $\hat{y}$ will have a different range of modified genome sequences. If the score difference between y and $\hat{y}$ is larger than the standard deviation of the unmodified genome sequence distribution, it may be that the sequence has been forcibly changed. Fig. 4 shows the histogram of the final score of y and $\hat{y}$ returned from softmax of the neural network. If the message is hidden, we can see that the final score from softmax of the neural network differs over the range $\overline{y} \pm \epsilon$. From Eq. (1) below, we show that detection is possible using information theoretical proof based on entropy H (Ref.[26]).

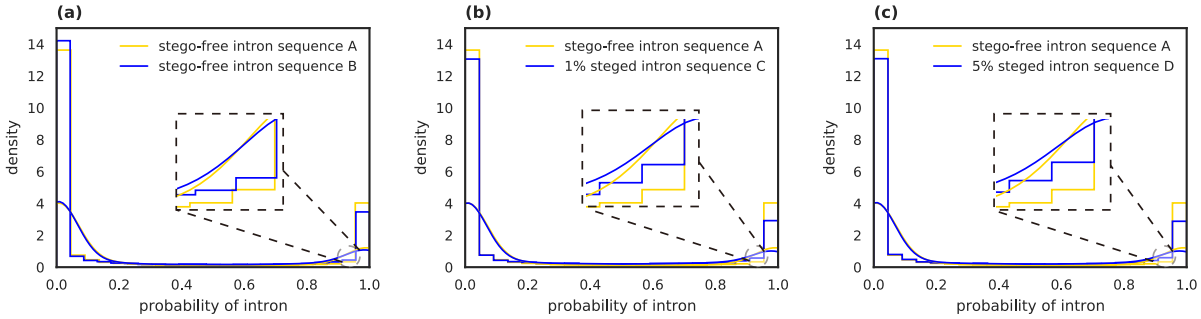

Fig. 4. Final score of intron/exon sequence obtained from the softmax of the neural network (best viewed in color). (a) kernel density differences between two stego-free intron sequences (b) kernel density differences between stego-free and 1% perturbed steged intron sequences. (c) kernel density differences between stego-free and 5% perturbed steged intron sequences.

Lemma 1. *A DNA steganography scheme is not secure if* $H(\mathbf{D}) \neq H(\hat{\mathbf{D}}|\mathbf{D})$.

Proof. The mutual joint entropy $H(\mathbf{D}, \hat{\mathbf{D}}) = H(\mathbf{D}) + H(\hat{\mathbf{D}}|\mathbf{D})$ is the union of both entropies for distribution $\mathbf{D}$ and $\hat{\mathbf{D}}$. According to Gallager at el[27], the mutual information of $I(\mathbf{D}; \hat{\mathbf{D}})$ is given as $I(\mathbf{D}; \hat{\mathbf{D}}) = H(\mathbf{D}) - H(\mathbf{D}|\hat{\mathbf{D}})$. It is symmetric in $\mathbf{D}$ and $\hat{\mathbf{D}}$ such that $I(\mathbf{D}; \hat{\mathbf{D}}) = I(\hat{\mathbf{D}}; \mathbf{D})$, and always non-negative. The conditional entropy between two distribution is 0 if and only if the distributions are equal. Thus, the mutual information must be zero to define secure DNA steganography schemes:

$$I(\mathbf{C}; (\mathbf{D}, \hat{\mathbf{D}})) = H(\mathbf{C}) - H(\mathbf{C}|(\mathbf{D}, \hat{\mathbf{D}})) = 0. \tag{2}$$

where $\mathbf{C}$ is message hiding space and it follows that:

$$H(\mathbf{C}) = H(\mathbf{C}|(\mathbf{D}, \hat{\mathbf{D}})). \tag{3}$$

Eq. (2) indicates that the amount of entropy $H(\mathbf{C})$ must not be decreased based on the knowledge of $\mathbf{D}$ and $\hat{\mathbf{D}}$. It follows that the secure steganography scheme is obtained if and only if:

$$\forall_i \in \mathbb{N}, m_i \in \mathbf{D}, \hat{m}_i \in \hat{\mathbf{D}} : m_i = \hat{m}_i. \tag{4}$$

Note that for $m_i = \hat{m}_i$ it is not possible to distinguish between the original sequence and the stego sequence. Considering that the representations of $\hat{m}$ are limited to {`A,C,G,T`}, it is nearly impossible to satisfy the condition because current steganography schemes are all based on the assumption of addition or substitution. Because $\mathbf{C}$ is independent of $\mathbf{D}$, the amount of information will increase over distribution $\mathbf{D}$ if hidden messages are inserted over distribution $\hat{\mathbf{D}}$. We can conclude that the schemes are not secure under condition $H(\mathbf{C}) > H(\mathbf{C}|(\mathbf{D}, \hat{\mathbf{D}}))$. □

3.2. *Proposed Steganalysis RNN Model*

The proposed model is based on sequence-to-sequence learning using an autoencoder and stacked RNNs[28], where the model training consists of two main steps: 1) unsupervised pre-training of sequence-to-sequence autoencoder for modeling an overcomplete case, and 2) supervised fine-tuning of stacked RNNs for modeling patterns between canonical and non-canonical

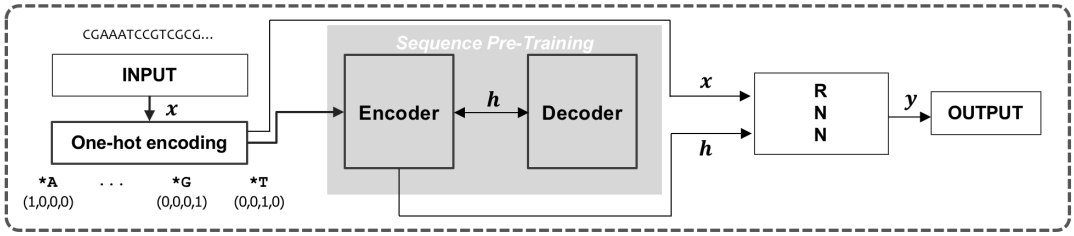

Fig. 5. Overview of proposed RNN methodology.

splice sites (see Fig. 5). In the proposed model, we use a set of DNA sequences labeled as introns and exons. These sequences are converted into a binary vector by orthogonal encoding[29]. It employs n_c-bit one-hot encoding. For $n_c = 4$, {`A,C,T,G`} is encoded by

$$\langle [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1] \rangle. \tag{5}$$

For example, the sequence `ATTT` is encoded into a 4×4 dimensional binary vector $\langle [1,0,0,0], [0,0,0,1], [0,0,0,1], [0,0,0,1] \rangle$. The encoded sequence is a tuple of a four-dimensional (4D) dense vector, and is connected to the first layer of an autoencoder, which is used for the unsupervised pre-training of sequence-to-sequence learning. An autoencoder is an artificial neural network (ANN) that is used to learn meaningful encoding for a set of data in a case involving unsupervised learning. An autoencoder consists of two components, namely an encoder and decoder.

The encoder RNN encodes $\mathbf{x}$ to the representation of sequence features $\mathbf{h}$, and the decoder RNN decodes $\mathbf{h}$ to the reconstructed $\hat{\mathbf{x}}$; thus minimizing the reconstruction errors of $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$, where $\mathbf{x}$ is one-hot encoded input. Through unsupervised learning of the encoder-decoder model[30], we obtain representations of inherent features $\mathbf{h}$, which are directly connected to the second activation layer. The second layer is RNNs layer used to construct the model. The model in turn is used to determine patterns between canonical and non-canonical splice signals. We then obtain the tuple of fine-tunned $\mathbf{h} =< \mathbf{h_1}, \cdots, \mathbf{h_d} >$, where $\mathbf{h}$ is the representation of sequence features learned by features, which is a representation of introns and exons in hidden layers, and $\mathbf{d}$ is the dimension of a vector.

The features $\mathbf{h}$ learned from the autoencoder are connected to the second stacked RNN layer, which consists of our proposed architecture for outputting a classification score for the given sequence $D_i \in \mathbf{D}$. For the fully connected output layer, we use the sigmoid function as the activation. The activation score is given by $\Pr(y = i|\mathbf{h}) = \frac{1/(1+\exp(-\mathbf{w}_i^T \mathbf{h}))}{\sum_{k=0}^{1} 1/(1+\exp(-\mathbf{w}_k^T \mathbf{h}))}$, where y is the label that indicates whether the given region contains introns ($y = 1$) or exons ($y = 0$). For our training model, we use a recently proposed optimizer of multi-class logarithmic loss function Adam[31]. The objective function $\mathcal{L}(\mathbf{w})$ that must be minimized is defined as follows:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N} \sum_{n=1}^{N} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \tag{6}$$

where N is the mini-batch size. A model $\mathbf{M}_{D_i}$ has a possible score of p_i for one species, where p_i is the score of given non perturbed sequences.

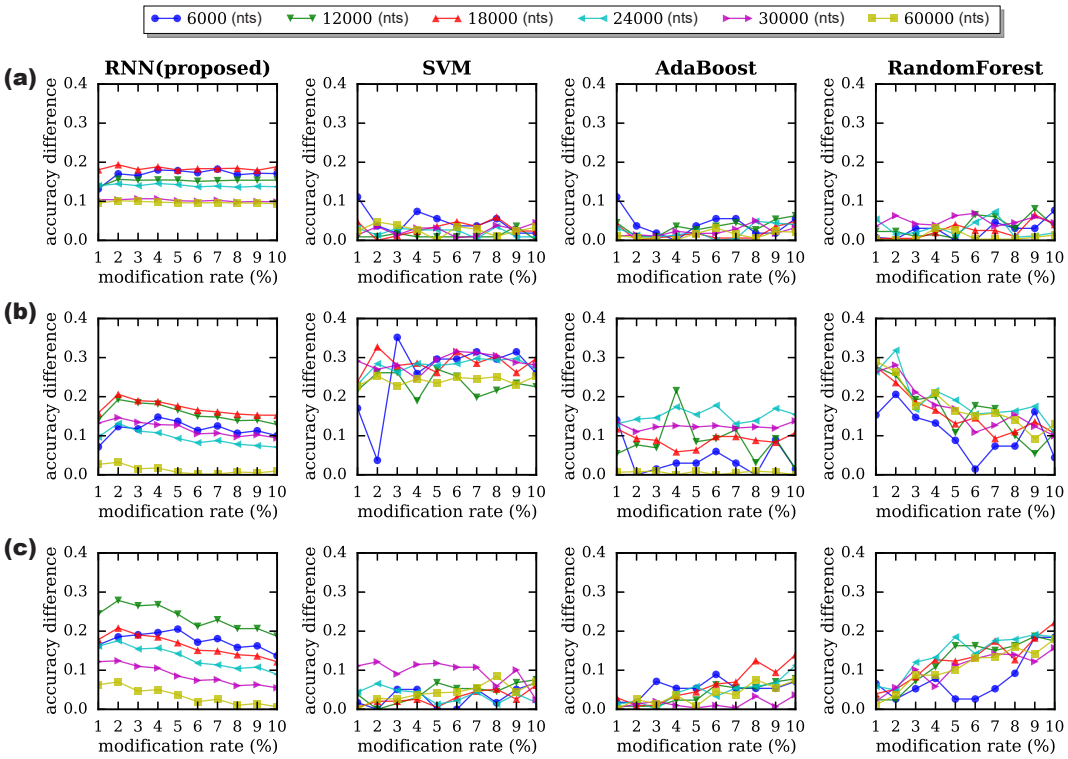

Fig. 6. Comparison of learning algorithms with random hiding algorithms (best viewed in color). (a) differences in accuracy for intron region (b) differences in accuracy for exon region (c) difference in accuracy for both region. [The performances of four supervised learning algorithms when detecting hidden messages are shown for six variable lengths of nucleotides (nts).]

4. Results

4.1. *Dataset*

We simulated our approach using the Ensembl human genome dataset and human UCSC-hg38 dataset[32], which include sequences from 24 human chromosomes (22 autosomes and 2 sex chromosomes). The Ensembl human genome dataset has a two-class classification (coding, and non-coding) and the UCSC-hg38 dataset has a three-class classification (donor, acceptor, and non-site).

4.2. *Input Representation*

The machine learning approach typically employs a numerical representation of the input for downstream processing. Orthogonal encoding, such as one-hot coding[29], is widely used to convert DNA sequences into a numerical format. It employs n_c-bit one-hot encoding. For $n_c = 4$, {`A,C,T,G`} is encoded as described in Eq. (5). According to Lee et al.[17], the vanilla one-hot encoding scheme tends to limit generalization because of the sparsity of its encoding (75% of the elements are zero). Thus, our approach encodes nucleotides into a 4D dense vector that follows the direct architecture of a normal neural network layer[33], which is trained by the gradient decent method.

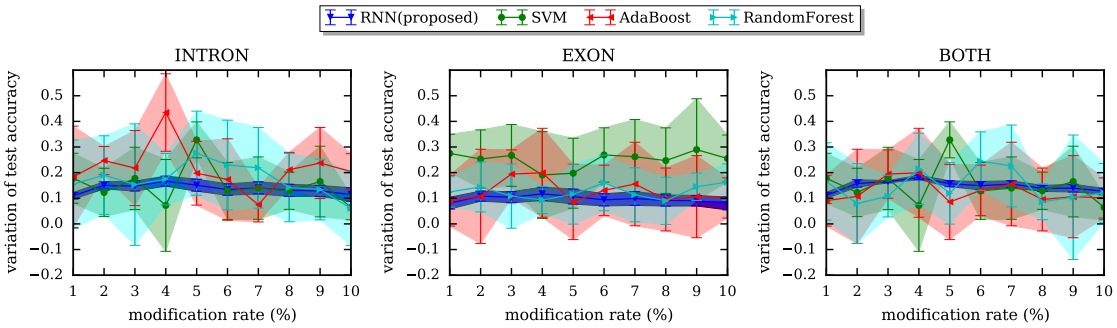

Fig. 7. Comparison of learning algorithms in terms of robustness (best viewed in color). Mean and variance of accuracy are measured for the fixed DNA sequence length of 6000 for 500 cases by changing one percent of the hidden message. The shaded line represents the standard deviation of the inference accuracy.

4.3. *Model Training*

The proposed RNN-based approach uses unsupervised training for the autoencoder and supervised training for the fine-tuning. The first layer of unsupervised training uses 4 input units, 60 hidden RNNs units with 50 epochs and 4 output units that are connected to the second layer. The second layer of supervised training uses 4 input units that are connected to stacked LSTM layers with full version including forget gates and peephole connections. The 4 input layers are used for 60 hidden units with 100 epochs, and the 4 output units are a fully connected output layer containing K units for K-class prediction.

In our experiment, we used $K = 2$ to classify sequences (coding or non-coding). For the fully connected output layer, we used the softmax function to classify sequences and the sigmoid function to classify sites for the activation. For our training model, we used a recently proposed optimizer of multi-class logarithmic loss function Adam[31]. The objective function $\mathcal{L}(\mathbf{w})$ that has to be minimized is as described in Eq (6). We used a batch size of 100 and followed the batch normalization[34]. We initialized weights according to a uniform distribution as directed by Glorot and Bengio[35]. The training time was approximately 46 hours and the running time was less than 1 second (Ubuntu 14.04 on 3.5GHz i7-5930K and 12GB Titan X).

4.4. *Evaluation Procedure*

For evaluation of performance, we used the score obtained from the softmax of the neural network. We exploited the state-of-the-art algorithm[2] to embed hidden messages for the message hiding. We randomly selected DNA sequences from the validation set using the Ensembl human genome dataset. We obtained the score of the stego-free sequence from the validation set. In the next step, we embedded hidden messages to a selected DNA sequence from the validation set, and we obtained the score. Using the score distribution of the stego-free and steged sequences, we evaluated the different scores for the range $\overline{y} \pm \epsilon$. The output from softmax of the neural network is expected to have a similar score distribution as the unmodified genome sequences. However, the score distribution changes if messages are embedded. As shown in Fig. 4(b) and Fig. 4(c), modified sequences are distinguishable using our RNNs model.

Table 1. Detection performance of sequence alignment and denoising tools.

	Both Region (%)	**Intron Region (%)**	**Exon Region (%)**
RNN (proposed)	**99.93**	**99.96**	**99.94**
BLAST[36]	84.00	85.00	85.00
Coral[37]	0.00	0.00	0.00
Lighter[38]	0.00	0.00	0.00

4.5. *Performance Comparison*

We evaluated the performance of our proposed method based on four supervised learning algorithms (RNNs, SVM, random forests, and adaptive boosting) to detect hidden messages. For the performance metric, we used the differences in accuracy.[a] Using the prediction performance data, we evaluated learning algorithms with respect to the following three regions; introns dedicated, exons dedicated, and both regions together.

For each algorithm, we generated simulated data for different lengths of DNA sequences (6000, 12000, 18000, 24000, 30000, and 60000) using the UCSC-hg38 dataset[32]. We also randomly selected 1000 cases for the fixed DNA sequence length for the modification rate 1 to 10%. Using selected DNA sequences, we obtained the average prediction accuracy of different numbers of samples against non-perturbed DNA sequences for 1000 randomly selected cases. In the next step, we obtain the prediction accuracy for the modified data generated according to the hiding algorithms. Using the averaged prediction accuracy for both the perturbed and non-perturbed cases, we evaluated the differences between the prediction accuracy rates for varying different numbers of samples. We carried out five-fold cross-validation to obtain the mean/variance of the differences in accuracy.

Fig. 6 shows an experiment for each algorithm using six variable DNA sequence lengths. Each algorithm was compared to three different regions based on the six variable DNA sequence lengths. The experiments were conducted by changing from one to then percent of the hidden message. SVM showed good detection performance in the exon region, but showed inferior performance in the intron as well as both regions category. In the case of adaptive boosting, the detection performance was similar in both regions and in intron only categorie, but performed poorly in exon regions. In the case of the random forest, the cases with the exon and both regions showed good performance except for some modification rates. In the intron regions, the detection performance was similar to that of other learning algorithms. Notably, our proposed methodology based on RNNs outperformed all of the existing hidden messages detection algorithms for all genomic regions evaluated.

In addition, we examined our proposed methodology based on denoising methods using Coral[37] and Lighter[38]. The UCSC-hg38 dataset was used to preserve local base structures and perturbed data samples were used as random noise. As shown in Table 1, the results showed that both Coral and Lighter missed detection for all modification rates in all regions. In addition, the sequence alignment method performed poorly. The results suggest that there is a 15 to 16% chance that hidden messages may not be detected in all three regions.

[a]Accuracy $= (TP + TN)/(TP + TN + FP + FN)$, where TP, FP, FN, and TN represent the numbers of true positives, false positives, false negatives, and true negatives, respectively.

To validate the learning algorithms with respect to robustness, we tested them with a fixed DNA sequence length of 6000 with 500 cases for each modification rate to measure the mean and variance of the test accuracy. Fig. 7 shows how the performance measures (mean and variance of accuracy differences) change for modification rates ranging from 1 to 10 in the intron, exon, and both regions categories. The plotted entries represents the the averaged mean over the 500 cases, and shade lines show the average of the variances over the 500 cases. The results indicate that hidden messages may not be detected if the prediction difference is less than the variance. The overall analysis with respect to the robustness showed that the learning algorithms of SVM, random forests and adaptive boosting performed poorly.

5. Discussion

The development of next-generation sequencing has reduced the price of personal genomics[39], and the discovery of the CRIPSPR-Cas9 gene has provided unprecedented control over genomes of many species[40]. While the technology is yet to be applied to simulations involving artificial DNA, human DNA sequences may become an area in which we can apply DNA watermarking. Our experiments using the real UCSC-hg38 human genome implicitly consider that unknown relevant sequences are also detectable because of the characteristics of similar patterns in non-canonical splice sites. The number of donors with `GT` pairs and acceptors with `AG` pairs were found to be 86.32% and 84.63%, respectively[16]. Existing steganography techniques modify several nucleotides. Considering few single nucleotide modifications, we can transform DNA steganography to the variant calling problem. In this regard, we believe that our methodology can be extended to the field of variant calling.

Although there are many advantages to using machine learning techniques to detect hidden messages[41–43], the following improvements are required: parameter tuning is dependent on the steganalyst, e.g., the training epochs, learning rate, and size of the training set; the failure to detect hidden messages cannot be corrected by the steganalyst. However, we expect that the future development of such techniques will resolve the limitations. According to Alvarez and Salzmann[44], the numbers of layers and neurons of deep networks can be determined using an additional class of methods, sparsity regularization, to the objective function. The sizes of vectors of grouped parameters of each neuron in each layer incur penalties if the loss converges. The affected neurons are removed if the neurons are assigned a value of zero.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) [2014M3C9A3063541, 2018R1A2B3001628], and the Brain Korea 21 Plus Project in 2018.

References

1. K. Bennett (Citeseer, 2004).
2. B. A. Mitras and A. Abo, *International Journal of Information Technology and Business Management* **14**, 96 (2013).
3. M. B. Beck, E. C. Rouchka and R. V. Yampolskiy, 204 (2012).

4. A. Gehani, T. LaBean and J. Reif, 167 (2003).
5. H. J. Cordell and D. G. Clayton, *The Lancet* **366**, 1121 (2005).
6. S. Katzenbeisser and F. Petitcolas (Artech house, 2000).
7. C. T. Clelland, V. Risca and C. Bancroft, *Nature* **399**, 533 (1999).
8. N. Yachie, K. Sekiyama, J. Sugahara, Y. Ohashi and M. Tomita, *Biotechnology progress* **23**, 501 (2007).
9. D. G. Gibson, J. I. Glass, C. Lartigue, V. N. Noskov, R.-Y. Chuang, M. A. Algire, G. A. Benders, M. G. Montague, L. Ma, M. M. Moodie *et al.*, *science* **329**, 52 (2010).
10. S. Brenner, S. R. Williams, E. H. Vermaas, T. Storck, K. Moon, C. McCollum, J.-I. Mao, S. Luo, J. J. Kirchner, S. Eletr *et al.*, *Proceedings of the National Academy of Sciences* **97**, 1665 (2000).
11. K. Tanaka, A. Okamoto and I. Saito, *Biosystems* **81**, 25 (2005).
12. D. Heider and A. Barnekow, *BMC bioinformatics* **8**, p. 176 (2007).
13. S. Jiao and R. Goutte, (2008).
14. I. K. Maitra, *Journal of Global Research in Computer Science* **2** (2011).
15. K. Grosse, P. Manoharan, N. Papernot, M. Backes and P. McDaniel, *arXiv preprint arXiv:1702.06280* (2017).
16. T. Lee and S. Yoon *International Conference on Machine Learning* 2015.
17. B. Lee, T. Lee, B. Na and S. Yoon, *arXiv preprint arXiv:1512.05135* (2015).
18. L. v. d. Maaten and G. Hinton, *Journal of Machine Learning Research* **9**, 2579 (2008).
19. R. Anderson (Springer Science & Business Media, 1996).
20. R. Canetti, O. Goldreich and S. Halevi, *Journal of the ACM (JACM)* **51**, 557 (2004).
21. M. Bellare and P. Rogaway, 62 (1993).
22. H. Keren, G. Lev-Maor and G. Ast, *Nature Reviews Genetics* **11**, 345 (2010).
23. D. J. Lockhart and E. A. Winzeler, *Nature* **405**, 827 (2000).
24. B. Shimanovsky, J. Feng and M. Potkonjak, 373 (2002).
25. J. Schmidhuber, *Neural networks* **61**, 85 (2015).
26. R. E. Blahut (Addison-Wesley Longman Publishing Co., Inc., 1987).
27. R. G. Gallager, *Information theory and reliable communication* (Springer, 1968).
28. S. M. Peterson, J. A. Thompson, M. L. Ufkin, P. Sathyanarayana, L. Liaw and C. B. Congdon, *Frontiers in genetics* **5**, p. 23 (2014).
29. P. Baldi and S. Brunak, *Bioinformatics: the machine learning approach* (MIT press, 2001).
30. N. Srivastava, E. Mansimov and R. Salakhutdinov, 843 (2015).
31. D. Kingma and J. Ba, *arXiv preprint arXiv:1412.6980* (2014).
32. W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler and D. Haussler, *Genome research* **12**, 996 (2002).
33. F. Chollet *et al.*, *URL: https://keras. io/k* **7** (2015).
34. S. Ioffe and C. Szegedy, *arXiv preprint arXiv:1502.03167* (2015).
35. X. Glorot and Y. Bengio, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.
36. S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman, *Journal of molecular biology* **215**, 403 (1990).
37. L. Salmela, *Bioinformatics* **26**, 1284 (2010).
38. L. Song, L. Florea and B. Langmead, *Genome biology* **15**, p. 509 (2014).
39. S. C. Schuster, *Nature methods* **5**, p. 16 (2008).
40. P. D. Hsu, E. S. Lander and F. Zhang, *Cell* **157**, 1262 (2014).
41. S. Lyu and H. Farid, **5306**, 35 (2004).
42. S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, *Pattern Recognition* **58**, 121 (2016).
43. S. Min, B. Lee and S. Yoon, *Briefings in bioinformatics* **18**, 851 (2017).
44. J. M. Alvarez and M. Salzmann, in *Advances in Neural Information Processing Systems*, 2016.