Using Artificial Neural Networks to Model Errors in Biochemical Manipulation of DNA Molecules

Alan J.X. Guo and Hao Qi

***Abstract*—In recent years, the non-biological applications of DNA molecules have made considerable progress; most of these applications were performed *in vitro*, involving biochemical operations such as synthesis, amplification and sequencing. Because errors may occur with specific sequence patterns or experimental instruments, these biochemical operations are not completely reliable. Modeling errors in these biochemical procedures is an interesting research topic. For example, researchers have proposed several methods to avoid the known vulnerable sequence patterns in the study of storing binary information in DNA molecules. However, there are few end-to-end methods to evaluate these biochemical errors with regard to the DNA sequences. In this article, based on the data generated by a DNA storage research, we use artificial neural networks to predict whether a DNA sequence tends to cause errors in biochemical operations. Through comparative experiments and hyperparameter optimization, we analyze the known and potential problems in the research process. As a result, an end-to-end method to model the biochemical errors of DNA molecules *in vitro* through a computer system is proposed.**

***Index Terms*—Sequence Analysis, DNA storage, Artificial Neural Network**

I. Introduction

Since the structure of DNA molecules was revealed in [1], many studies have been conducted in related fields. In the past few decades, the non-biological applications of DNA molecules have received extensive attention, to name a few, DNA storage [2], DNA nanotechnology [3], *etc*.

Storing information, which is one of the most important jobs of DNA molecules *in vivo*, has attracted much attention for its advantages of high storage density, low maintenance cost, and fast parallel replication [2], [4]. Storing binary information through DNA molecules usually consists of five steps, which are encoding binary data into four bases (A,T,G,C), DNA synthesis, polymerase chain reaction (PCR) amplification, molecular sequencing, and decoding the DNA sequences into binary data [2]. During these steps, the biochemical processes may produce several kinds of errors or noises. In practice, some patterns are hard to be synthesized or sequenced and some patterns are prone to vary in amplification [5].

Since the landmark works of Church and Goldman in the field of large-scale DNA storage [6], [7], researchers have been trying to alleviate or bypass the abovementioned problems. In [6], Church *et al.* mapped both two bases A, C (resp. G, T) to one binary digit 0 (resp. 1), so the homopolynucleotides can be avoided. At the same time, the author also used about 3000-fold coverage and paired-end sequencing to reduce errors in biochemical procedures. In [7], Goldman *et al.* encoded the information into bases depending on their previous bases to avoid homopolynucleotides and used redundant DNA molecules (each molecule has the last 75% bases overlapped with another molecule) to increase the chance of retrieving the corrected information from the sequencing data. Logical redundancy has also been applied to correct or judge the errors in the sequencing data. Grass *et al.* introduced the Reed-Solomon code to encode the information into bases that able to do error correction [8]. In [9], Blawat *et al.* counted the error rates of swap, insertion, and deletion from synthesis to sequencing. Further, they designed the forward error correction schemes to tackle these errors. In [10], Erlich *et al.* designed a DNA fountain along with the Reed-Solomon code. The DNA fountain enables recovering data from enough droplets, which are the encoded sequences, hence they were able to screen the redundant weak droplets before synthesis. In their work, the droplets were evaluated by the GC content and homopolymer runs. Recently, Chen *et al.* [11] developed statistical models for the copy unevenness problem and further explored the trade-off between the bias and redundancy in DNA storage research.

Accurately modeling the errors in the biochemical manipulation of each given DNA sequence is very interesting to the non-biological applications of DNA technology. It enables the researchers to avoid using weak DNA molecules in their subsequent work. In the field of DNA storage, a straightforward application is to design more effective rules to screen out weak sequences before synthesis. The aforementioned applications alleviate the shortcomings caused by the biochemical errors. However, the description of these biochemical errors with regard to a specific DNA sequence is not clear enough. Researchers usually consider rough characteristics of vulnerable sequences and use imprecise general techniques to tackle these issues. In the study of DNA storage, these settings usually introduce considerable redundancy in the coding and reduce the coding capacity.

Artificial neural networks (ANN) use computer systems to simulate biological neurons to perform intelligent works. They have made extraordinary achievements in many research fields, such as object detection, natural language processing (NLP), *etc.* [12]. In order to process sequence data such as text, audio, *etc.*, plenty of methods based on ANN have been studied. Milestones include the recurrent neural network (RNN) [13] and its variation long short-termed memory (LSTM) [14],

A. Guo is with Center for Applied Mathematics, Tianjin University, Tianjin, China. E-mail: jiaxiang.guo@tju.edu.cn

H. Qi is with School of Chemical Engineering and Technology, Tianjin University, Tianjin, China, and Key Laboratory of Systems Bioengineering (Ministry of Education), Tianjin University, Tianjin, China.

This work was supported by the National Key Research and Development Program of China under Grant 2020YFA0712100, and the National Natural Science Foundation of China under Grant 11801409.

attention mechanism [15], [16], *etc.*. These methods are successful in solving NLP problems, for the ANN-based models can present not only the feature from the fixed position but also the contextual features from the global data [16].

ANN or deep learning methods have also been applied in many works involving DNA sequences [17]. In [18], Angermueller *et al.* used a deep learning method called DeepCpG to predict methylation states in a single cell. In [19], Washburn *et al.* proposed deep learning methods for predicting relative transcript abundance from DNA sequence. In [20], Umarov *et al.* applied deep learning models for promoter analysis and prediction. Other works on genomic prediction or gene function include [21]–[24].

In this article, we propose an ANN-based method that models the biochemical errors in the DNA sequence during synthesis, amplification, and sequencing. In order to explore the capacity of the ANN-based methods on this task, We conduct experiments with different ANN-based models and hyperparameters. In addition, we analyze the advantages and disadvantages of ANN-based methods in modeling the errors from DNA sequences. To the best of our knowledge, such a study has not been approached so far.

The remainder of this paper is structured as follows. Section II clarifies the tasks and the data of DNA sequences for training, validating, and testing. Section III introduces the proposed ANN. Section IV shows the performance of the proposed method, the effect of hyperparameters introduced by the proposed method, and the comparison between the proposed method and other popular ANN-based models. Section V analyzes the issues that raises in this work and the further research directions. Section VI provides some concluding remarks.

II. Evaluating Task and Data Preprocessing

In this paper, we propose an ANN-based methods to distinguish the vulnerable positions from the stable ones in the encoded sequences. It is assumed that the probability of a biochemical error occurring at a particular position in the nucleotide sequence is affected by its neighbors with distance-related decay. Under this assumption, the evaluating task can be mathematically expressed as finding an ANN-based function f that maps a nucleotide fragment $\boldsymbol{x}$ with the proper lengh to the probability of a biochemical error

$$y = f(\boldsymbol{x}) \tag{1}$$

occurring at its central position.

The data produced by a DNA storage research [25] was used for training and testing. In their research, the isothermal DNA amplification was recruited as a low-bias amplification of DNA oligo pool for robust DNA storage. The binary data, which was going to be stored in DNA molecules, was compressed by a computer and then encoded into short sequences from alphabet $\{A, T, G, C\}$ of fixed length. Glued with functional sequences, the result sequences were ready for synthesis. These sequences to be synthesized are usually called the *reference*s. After synthesis and amplification, the information was retrieved by sequencing the DNA molecules. The retrieved sequences are usually called the *read*s. To fulfill the procedure of DNA storage, the stored binary data was obtained by decoding the reads. Although the DNA molecules were amplificated by the low-bias isothermal DNA amplification method, some references still lead to noisy reads. In our work, we purposed to model the errors in the reads based on the original references.

In the DNA storage research [25], where the experimental data was acquired from, two files A and B were turned into references with length $150\mathrm{nt}$, seperately. Let us use $S = \{\boldsymbol{s_i}\}$ to denote these references, where the $\boldsymbol{s}_i \in \{A, T, G, C\}^{150}$. After synthesis, amplification, and sequencing, the retrieved reads, denoted by $\tilde{S} = \{\tilde{\boldsymbol{s}}_j\}$, were expected to meet the $\boldsymbol{s}_i$s. However, three kinds of common errors, which are the swap error, the insertion error, and the deletion error [9], had the opportunity to occur at any position in any sequences from $\tilde{S}$. In order to depict these errors in our work, the reads were seperated into subsets $\tilde{S}^i = \{\tilde{\boldsymbol{s}}_j^i\}$, where each subset originated from a specific reference $\boldsymbol{s}_i$, by comparing the reads and references via alignment algorithms [26], [27]. Later, for each position of the reference $\boldsymbol{s}_i$, the rates of the three kinds of errors were recorded by comparing each $\tilde{\boldsymbol{s}}_j^i$ with $\boldsymbol{s}_i$. The pipeline of the data preprocessing, which assigned each reference a position-wised error vector, is presented in Fig. 1.

Disregarding the reference $\boldsymbol{s}_i$, the swap errors, insertion errors, and deletion errors globally happened in chances of 0.477%, 0.030%, and 0.146%, respectively. Note that these numbers may vary with different equipments in biochemical experiments. Among these three kinds of errors, the swap error had the most cases and was simplest to depict. In view of this, the swap error was chosen as the researching object. Regarding to a reference $\boldsymbol{s} = (s_1, s_2, \ldots, s_{150})$, let $\boldsymbol{e} = (e_1, e_2, \ldots, e_{150})$ be the vector whose element e_i recorded the ratio of swap errors happened at s_i in the related reads. Firstly, a threshold hyperparameter K was used to separate the positive positions from negative ones. To be precise, the positive positions had error ratios larger than threshold $e_i \geq K$, while the negative positions had no errors happened, $e_i = 0$. Secondly, the reference $\boldsymbol{s}$ was truncated into fragments $\{\boldsymbol{x}_i\}$ by windows sliding of size L. Each fragment $\boldsymbol{x}_i$ could be labeled positive with $y_i = 1$ or negative with $y_i = 0$ according to its central position's label. In the end, the fragments and their labels $(\boldsymbol{x}_i, y_i)$ were separated into training data, validation data, and testing data. In order to avoid overlapping between training data and testing data, both the training data and validation data were related to the original binary file A, while the testing data was related to the file B. The positive samples were usually far less than the negative samples, we duplicated the positive samples and randomly dropped some negative samples to maintain label balance.

The two hyperparameters, which are the threshold K and the window size L, affected the performance of the proposed method. With a smaller threshold K, the difference between positive samples and negative samples decreased, so the separation between positive and negative samples would be more difficult. While with a larger threshold K, the number of positive samples was declined and that may cause overfitting issues. As for the window size L, a smaller L may omit the key pattern that actually determines the stability of the

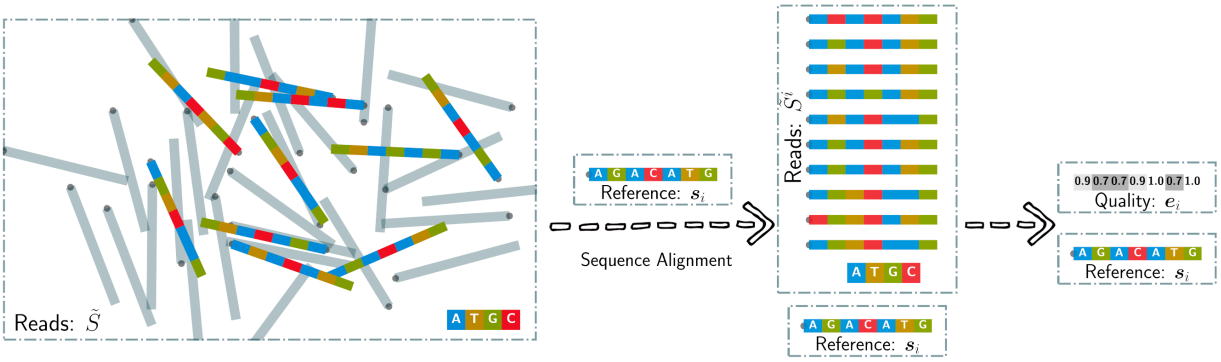

Fig. 1. Pipeline of data preprocessing. The reference $\boldsymbol{s}_i$ was one of the references generated by encoding the binary data, while the set of reads $\tilde{S}$ was obtained by sequencing the stored DNA molecules, from the research [25]. In our work, the reads originated from reference $\boldsymbol{s}_i$ were gathered into $\tilde{S}^i$ by algorithm of sequence alignment [26], [27] with reference $\boldsymbol{s}_i$. Counting the fraction of errors that happened on each position of reference $\boldsymbol{s}_i$, a position-wise error vector $\boldsymbol{e}_i$ was assigned to the reference $\boldsymbol{s}_i$ for the later research.

center position, while a larger L would cause computational burden. All these issues may decline the performance of the proposed model. We analyze the relationship between the hyperparameters and the model performance in detail, and give balanced values of K and L, in Section IV-B.

III. Proposed Artificial Neural Network

As a key part of the proposed method, the function f in (1) takes the sequence of nucleotide $\boldsymbol{x}$ with length L as input and outputs the predicted stability $\hat{y}$ for $\boldsymbol{x}$. Among the various choices of ANN-based models, a model based on the attention mechanism was employed for the function f. For convenience, notation f was used to denote the engaged neural network. As shown in Fig. 2, the neural network f consisted of several layers, and each layer used simple formulas to process the input data to its output.

The arrows in Fig. 2 show how data flowed in the proposed network. Firstly, five 1D convolutional operations were independently performed on the input data with different sizes of convolution kernels from 1 to 5. Different from the conventional 2D convolutional neural networks [28], the 1D convolution is limited to the only free dimension in the sequence data. Applying these convolutional operations, the input data was transformed to hierarchical features which had higher dimensions. For example, the convolution with kernel size 1 outputs the information from each nucleotide in the inputed sequence, while the convolution with kernel size 5 outputs the information from a window of length 5 sliding along the inputed sequence. Similar structures called "group convolution" were firstly introduced in [28] to accelerate the model by deploying the convolutions on different GPUs. The convoluted information regarding the same position of $\boldsymbol{x}$ was concatenated before sent to the next operation. Secondly, the data flowed through two kinds of multi-head attentions [15], [16], which were the global attention and local attention. The attentions helped to draw the dependencies between the stability and the input sequences, both globally and locally. The implementation of global attention followed [16], which is also called Scaled Dot-Product Attention. Given queries $\boldsymbol{Q} = (\boldsymbol{q}_i)$, values $\boldsymbol{V} = (\boldsymbol{v}_i)$ of the same length, and a key $\boldsymbol{k}$, the attention function uses the key $\boldsymbol{k}$ and the queries $\boldsymbol{Q}$ to compute the weights of values $\boldsymbol{V}$, and outputs the weighted sum on the values $\boldsymbol{V}$, which could be formulated as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{k}, \boldsymbol{V}) = \text{softmax}\left(\frac{\{\boldsymbol{q}_i \cdot \boldsymbol{k}\}}{\sqrt{d_k}}\right) \cdot \boldsymbol{V}, \quad (2)$$

where d_k is the dimension of $\boldsymbol{k}$. If we use $\boldsymbol{K} = (\boldsymbol{k}_i)$ to denote a arrays of keys, the formula is written as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right) \boldsymbol{V}. \quad (3)$$

While applying the global attention in the proposed method, the input data $\boldsymbol{X}$ of the attention layers was firstly transformed to queries, keys and values by independent learnable matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$, and the output of the attention was computed by

$$\text{Attention}(\boldsymbol{X}\boldsymbol{W}^Q, \boldsymbol{X}\boldsymbol{W}^K, \boldsymbol{X}\boldsymbol{W}^V). \quad (4)$$

Considering the input data for the attentions was obtained by the convolution layers with different receptive fields, the attentions were expected to express the relations between different subsequences. The local attention works similar to the global ones, but only the local values are considered. In the proposed model, the local attentions were realized by two 1D convolutional operations. We also employed shortcut connections [29] to introduce low-level features before the final prediction. In Fig. 2, the shortcut connections is illustrated by the "Add & BN", where the "Add" operation is element-wised addition on two sources and "BN" implies the operation of batch normalization [30] to improve the stability of the model. Finally, the data flowed through several fully connected layers. The first two of them were position-wised, which were fulfilled by 1D convolutions with kernel-size 1, while the last two of them were conventional fully connected layers. All the convolutional and fully connected layers were followed with activations ReLU, except for the last layer which used $\mathrm{softmax}$ function to output the predicted distribution over positive and negative labels of $\boldsymbol{x}$. After setting a threshold T on the outputs of the last layer, the inputs $\boldsymbol{x}$ could be classified into positive against negative.

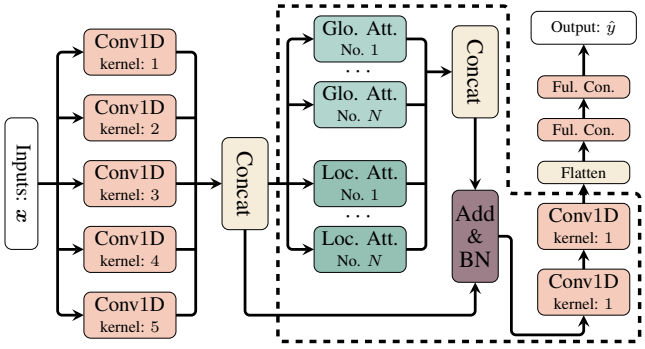

Fig. 2. Structure of the proposed model. Five 1D convolutions with different kernel sizes were used as the first part of the proposed model. The different convolutional kernel could output hierarchical features on the input sequence by their different sized receptive fields. Later, these features were sent into a attention module consisted of local attentions and global attentions. Finally, the output of attention module flowed through two fully connected layers, and a final prediction was given by the softmax function.

The structure in Fig. 2 indicates how the network f is built. During this procedure, undetermined parameters $\boldsymbol{\theta}$, such as the weights and biases, were introduced by the convolutional operation, attention, and fully connected layers. To find proper $\hat{\boldsymbol{\theta}}$ that enabled the function f fulfilling the proposed task, a cost function $\mathcal{L}$ was introduced on the outcome of f and the groundtruth labels on training data. By minimizing the cost function $\mathcal{L}$, the parameters $\boldsymbol{\theta}$ were optimized to $\hat{\boldsymbol{\theta}}$. If the neural network f had good generalization ability, the optimized f could predict the label y' of a new sample $\boldsymbol{x}'$ well. The cross-entropy cost between the groundtruth label y and $f(\boldsymbol{x})$ is the most commonly used in classification tasks. In this paper, the parameters $\boldsymbol{\theta}$ were optimized by minimizing the cross-entropy cost under the Adam [31] algorithm.

Different network architectures were also considered. We performed ablation studies and comparative experiments by replacing the attentions in Fig. 2 with identical mapping, simple RNN, and LSTM, in Section IV-C.

IV. Experiments

A. Evaluation metrics

As shown in Section II, most of the nucleotide fragments were stable during synthesis, amplification, and sequencing. In addition, after dividing the nucleotide fragments $\boldsymbol{x}$ into positive samples and negative samples with a threshold K, the imbalance between positive and negative samples was further aggravated. Therefore, common metrics, such as accuracy, precision, or recall, may be skewed in evaluating the performance.

In the experiments, the balanced accuracy (BA) at threshold $T = 0.5$ was engaged to evaluate the performance. The metric BA is calculated by the average of true positive rate (TPR) and true negative rate (TNR). It could also be interpreted as the accuracy when the positive samples and the negative samples have the same cardinalities. Also, the receiver operating characteristic (ROC) curve was used to show the comparison between different settings. The ROC curve plots

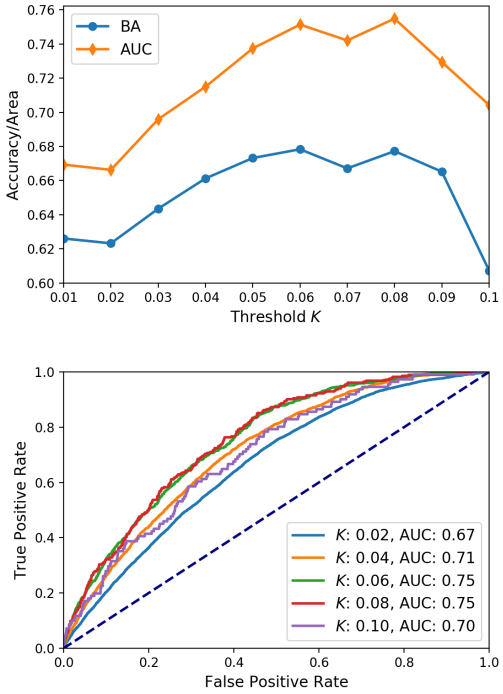

Fig. 3. The ROC curves and metrics BA and AUC with different thresholds K. As shown in this figure, the performance of proposed model increased when the thresold K was gained from a low value. The peak performance was obtained when $0.06 \leq K \leq 0.08$. When K was further increased, the metrics BA and AUC of the model decreased sharply.

the TPR against false positive rate (FPR) at various thresholds Ts. An ROC curve starts from $(0, 0)$ and ends to $(1, 1)$, the more north-west the curve locates, the better performance the classifier has. Along with the ROC curve, the area under curve (AUC) is another metric that is calculated by the area under the ROC curve. In most cases, better classifiers have higher AUC scores.

B. Hyperparameter optimization

In this subsection, we focus on how the hyperparameters K and L affected the performance of the proposed method.

As mentioned in Section II, the nucleotide fragments whose central position had error rates larger than threshold K were marked positive. The threshold K controlled the number of positive samples and also the discrimination of positive samples from negative ones. Hence, it greatly affected the performance of the model. After experiments with different Ks and fixed $L = 15$, the relation between the model performance and the threshold K is shown in Fig. 3.

It can be seen that, when the threshold K was relatively small, the performance will improve when K was increased. When the threshold K was greater than 0.07, the performance decreased when K was increased. As stated in Section II, K was the criterion of error rates that positive samples had. A higher K meant that the chosen positive samples had higher error rates, which also meant the positive samples may have more significant features that lead to errors. As a result, classifying the samples with a larger K was easier to a model.

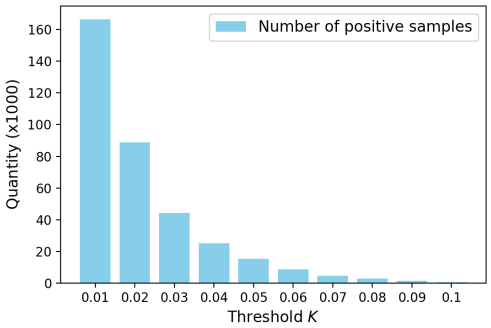

Fig. 4. The number of positive samples with different thresholds K. Different threshold on the error rates controlled the number of positive samples. As shown in this figure, the number of positive samples with $K = 0.01$ was hundreds times greater compared to the number with $K = 0.1$.

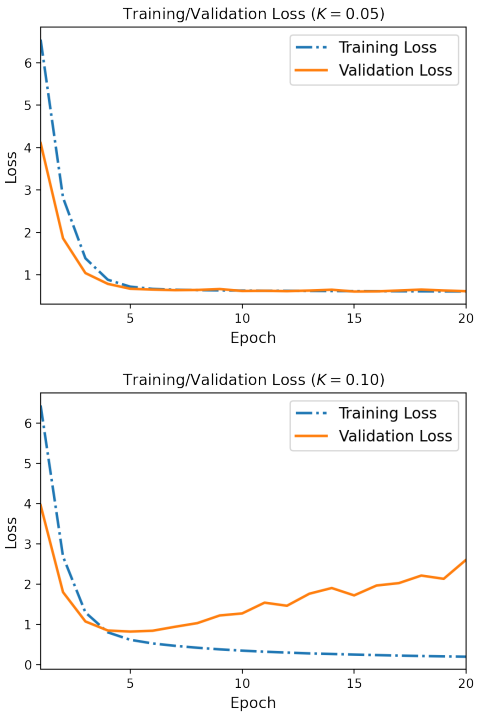

Fig. 5. The trainning and validation loss when $K = 0.05$ and $K = 0.10$. When $K = 0.05$, the validation loss was almost the same with training loss after each run of epoch, which indicated the model fitted the data well. When $K = 0.10$, the validation loss was significantly higher than the training loss after 5 epochs training, which indicated that the model overfitted on the training data.

However, the accuracy declined when $K > 0.07$. Another effect on the training samples of K was that K also controlled the number of positive samples. In Fig. 4, the relation between the number of positive samples and the threshold K is shown. It can be seen that, the number of positive samples with $K = 0.01$ was hundred times larger than $K = 0.1$. Small training data usually induces the overfitting issue on an ANN-based model, which means the model fits the training data well, but loses the generalization and fails in testing data. The training loss and validation loss were recorded in Fig. 5 with $K = 0.05$ and $K = 0.1$ to verify our conjecture. The validation loss was calculated at the end of each training epoch on the validation data, which was chosen independently from training data. Big validation loss and small training loss usually means the model is overfitted. In Fig. 5, we could see the model with $K = 0.1$ overfitted the training data at epoch 5, while the model $K = 0.05$ had less overfitting issue. Also, a large K that sieved most of the interesting samples out of the research contradicts the motivation of this work.

The size L of window sliding also played a key role in the proposed method. It determined the amount of information considered in the nucleotide fragment $\boldsymbol{x}$. The optimized L was determined by both the biochemical properties of the *in vitro* nucleotide sequence and the inherent ability of the proposed method to model the relationships or features between the nucleotide and its neighbors. It is straightforward that increasing L enabled the model to acquire more information for judging, which was verified by the experiments with L floating from 3 to 25. In Fig. 6, the performance increased along with the L from 3 to 9 and reached a plateau with $L > 9$. These experiments indicated that under the circumstance of the proposed method, a nucleotide fragment of length 9 had already contained most of the reachable information that determines the stability of its central position. More powerful models may take advantage of the complicated features from a larger neighborhood. However, the more complicated the model is, the more overfitting issue it has on the same amount of training samples. Regarding the quantity of data we got from the biochemical experiments, the proposed method was a good choice. Comparisons between different model structures could be found in Section IV-C.

C. Comparative experiments

In this subsection, we conducted comparative experiments by replacing the attention part of the proposed model, which is enclosed by the dashed lines in Fig. 2, with other well-known network structures.

Firstly, the network structure that was obtained by deleting the attention structure from the proposed model was considered. Notation Abs. (an abbreviation of absent) is used to denote this model in the following text. Secondly, the commonly-used network structures, which are plain RNN and LSTM, were tested. These two kinds of network structures perform well on time series data, for their ability to express features along the time axis. In our experiments, the bidirectional RNN and the bidirectional LSTM [32] were employed to enable the network to express the features from both sides. For fairness, the same experimental settings were applied on all these experiments except for the difference in the model structures. In detail, the hyperparameters were $K = 0.05$, $L = 15$, the learning rate was 0.0001, the optimizer was the Adam optimizer, and the experiments were early stopped by monitoring the validation loss. All the experiments had 5 runs, whose results were recorded in TABLE I by their average values and standard deviations in $\mathrm{mean}\pm\mathrm{std}$. The ROC curves were plotted by a random single run of the experiment.

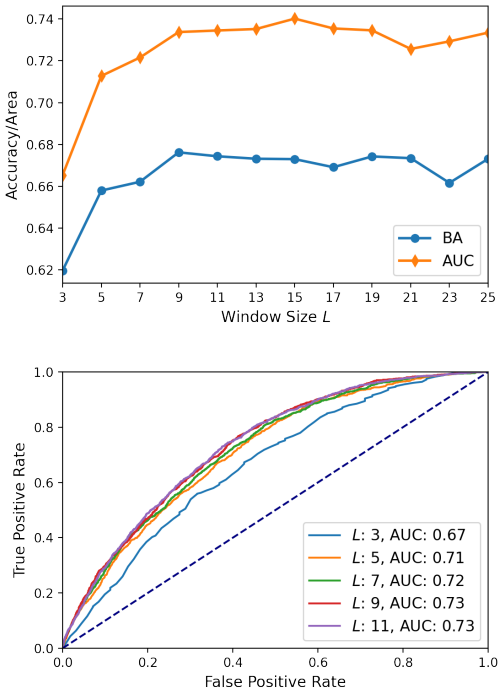

Fig. 6. The ROC curves and metrics BA and AUC with different window sizes L. The performance of the proposed model increased when the window size L was increased from $L = 3$ to $L = 9$. With a L greater than $L = 9$, the metrics reached a plateau. This indicated the extra information gained by a larger L than $L = 9$ was weakly linked to the error rate of the central position, or could not be captured and used by the proposed model.

In TABLE I and Fig. 7, we list the performance of the comparative experiments and the proposed model, which proposed model was marked with Att.. One could see that different network structures showed similar accuracies and AUCs. To be precise, the proposed model performed the best among the comparative experiments, while the model removed the attention structure had the worst performance. Moreover, the amount of parameters in each model is reported in the first row of TABLE I. The running time of each model is reported by timing 10 epochs training in the second row of the same figure, on a machine equipped with CPU of Intel Xeon Silver 4116 @2.1GHz and GPU of NVIDIA RTX2080ti.

Finally, a more complex network stacked LSTM and attentions (LSTM+Att.) was tested. Stacking RNN and attentions is common and effective in natural language processing [15]. In this paper, we performed experiments on LSTM+Att. to show that a complex model may not further improve the performance and was prone to overfit with the data from the biochemical experiments. To reveal the overfitting issues on complex models, the models were trained until 50 epochs. The metrics of the models after training with 20 epochs and 50 epochs are reported in TABLE II. These numbers show that the performance of the simple structured model Att. was slightly better than the complex structured model LSTM+Att. Moreover, the comparison of the performance between different training epochs shows that the simple model was more stable than the complex one. The reason may be

TABLE I
PERFORMANCE OF ABLATION STUDY AND COMPARATIVE EXPERIMENTS.
(AVERAGED OVER 5 RUNS, REPORTED IN mean $\pm$ std.)

	Att.	Abs.	RNN	LSTM
Params	105.63k	156.45k	154.53k	240.93k
Time(s)	107.1 ± 3.4	31.5 ± 0.5	106.5 ± 2.4	211.4 ± 4.03
BA	67.40 ± 0.14	65.35 ± 0.42	66.39 ± 0.50	65.63 ± 0.35
AUC	73.21 ± 0.35	70.71 ± 0.14	72.03 ± 0.18	71.10 ± 0.54

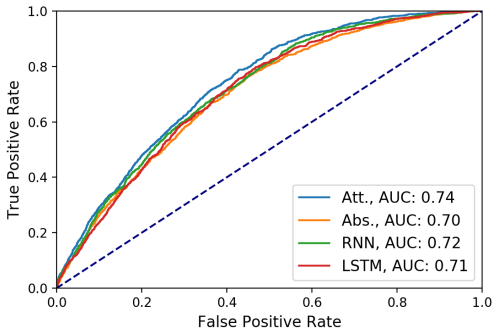

Fig. 7. The ROC curves of comparative experiments. The comparative methods resulted in similar ROC curves. Among these models, the proposed model (Att.) showed the best performance, while the model (Abs.) obtained by deleting the attention structure from the proposed model had the worst performance.

that complex models usually have more overfitting issues and hence result in a bad performance on testing data. The relations between training loss and validation loss, shown in Fig. 8, confirm our conjecture. When the number of training epochs was increased, the gap between the validation and training loss was greater on LSTM+Att. than Att., which indicates the overfitting happened.

V. KNOWN ISSUES AND RESEARCH DIRECTION

As one could see, a binary classifier with performance $\mathrm{ROC} = 0.74, \mathrm{BA} = 68\%$ is not good enough. In Section IV, different hyperparameters and several kinds of network structures were tried, but no further improvement was achieved. Moreover, the model overfitted before improving performance on complex model structures. We speculated that this was mainly caused by the soft label [33] used in this work. Most classification tasks are performed on hard labels, which label the samples into the classes they belong to. However, the label in our task was defined by the ratio of errors. How to define the bad guy from good ones? In this paper, we brutely assigned the positive labels on the position which had an error rate larger than $K = 5\%$. This setting retained most of the information about vulnerable fragments in positive samples, but it would

TABLE II
PERFORMANCE OF PROPOSED METHOD AND LSTM+ATT..
(AVERAGED OVER 5 RUNS, REPORTED IN mean $\pm$ std.)

	Att. (20 epochs)	Att. (50 epochs)	LSTM+Att. (20 epochs)	LSTM+Att. (50 epochs)
BA	67.12 ± 0.44	66.25 ± 0.37	65.99 ± 0.58	64.32 ± 0.73
AUC	73.57 ± 0.09	72.59 ± 0.46	72.23 ± 1.21	70.36 ± 0.23

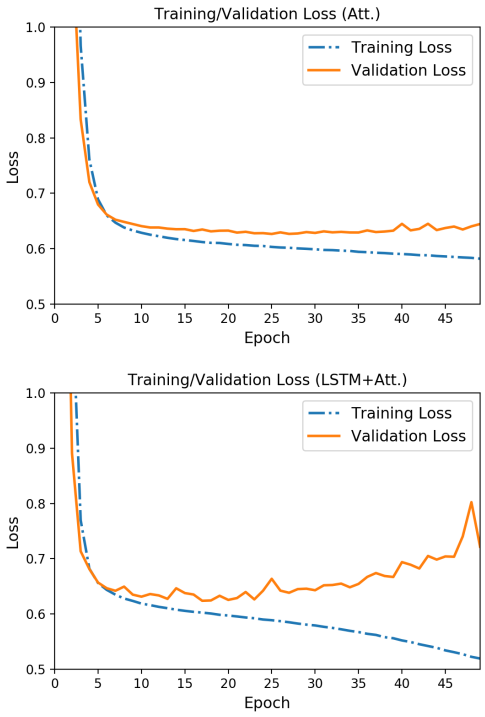

Fig. 8. The training and validation loss of Att. and LSTM+Att.. By comparing the curves of training loss and valication loss, it was indictated that the complicate model LSTM+Att. had greater overfitting effect than the proposed method Att..

also undoubtedly introduce considerable noises. Noise is one of the most important negative factors that degrade the model performance.

In order to overcome the aforementioned issues, a more precise description of the errors that happened during synthesis, amplification, and sequencing will be a good research direction. However, instead of the coarse target, modeling fine targets usually requires more data to support. In this paper, the data was not big enough to support a precise description of the errors, in view of the shortage of training samples shown in Fig. 4 and the overfitting issues shown in Figs. 5 and 8. In our opinion, fine researches on the errors, large scale sequencing data, and corresponding complex network may further improve the performance.

VI. Conclusion

Based on the sequencing data from a DNA storage research, an ANN-based method was proposed to model errors in biochemical manipulation of DNA molecules *in vitro*. The nucleotide sequences were first cut into nucleotide fragments by window sliding, each nucleotide fragment was labeled by its central position's error rate in the sequencing data. Later, an ANN-based model with the attention mechanism was built to fulfill a classification task that classifies the nucleotide fragments into the vulnerable and the stable classes. In the experimental part, we verified different hyperparameter values, explored different network structures, and further analyzed the effects of the complex model structures and overfitting issues. Based on the experiments, the proposed method was proved to be effective. By further analysis on the experiments and the experimental data, the potential issues and further research directions were talked.

References

[1] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid," *Nature*, vol. 171, no. 4356, pp. 737–738, Apr 1953.

[2] Y. Dong, F. Sun, Z. Ping, Q. Ouyang, and L. Qian, "DNA storage: research landscape and future prospects," *National Science Review*, vol. 7, no. 6, pp. 1092–1107, 01 2020.

[3] N. C. Seeman and H. F. Sleiman, "DNA nanotechnology," *Nature Reviews Materials*, vol. 3, no. 1, p. 17068, Nov 2017.

[4] Z. Ping, D. Ma, X. Huang, S. Chen, L. Liu, F. Guo, S. J. Zhu, and Y. Shen, "Carbon-based archiving: current progress and future prospects of DNA-based data storage," *GigaScience*, vol. 8, no. 6, 06 2019, giz075.

[5] T. P. Niedringhaus, D. Milanova, M. B. Kerby, M. P. Snyder, and A. E. Barron, "Landscape of next-generation sequencing technologies," *Analytical Chemistry*, vol. 83, no. 12, pp. 4327–4341, 06 2011.

[6] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012.

[7] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[8] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, pp. 2552–2555, 2015.

[9] M. Blawat, K. Gaedke, I. Hütter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011 – 1022, 2016, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

[10] Y. Erlich and D. Zielinski, "DNA Fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.

[11] Y.-J. Chen, C. N. Takahashi, L. Organick, C. Bee, S. D. Ang, P. Weiss, B. Peck, G. Seelig, L. Ceze, and K. Strauss, "Quantifying molecular bias in DNA data storage," *Nature Communications*, vol. 11, no. 1, p. 3264, 2020.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008.

[17] H. Li, S. Tian, Y. Li, Q. Fang, R. Tan, Y. Pan, C. Huang, Y. Xu, and X. Gao, "Modern deep learning in bioinformatics," *Journal of Molecular Cell Biology*, 06 2020, mjaa030.

[18] C. Angermueller, H. J. Lee, W. Reik, and O. Stegle, "DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning," *Genome Biology*, vol. 18, no. 1, p. 67, 2017.

[19] J. D. Washburn, M. K. Mejia-Guerra, G. Ramstein, K. A. Kremling, R. Valluru, E. S. Buckler, and H. Wang, "Evolutionarily informed deep learning methods for predicting relative transcript abundance from DNA sequence," *Proceedings of the National Academy of Sciences*, vol. 116, no. 12, pp. 5542–5549, 2019.

[20] R. Umarov, H. Kuwahara, Y. Li, X. Gao, and V. Solovyev, "Promoter analysis and prediction in the human genome using sequence-based deep learning models," *Bioinformatics*, vol. 35, no. 16, pp. 2730–2737, 01 2019.

[21] H. Wang, E. Cimen, N. Singh, and E. Buckler, "Deep learning for plant genomics and crop improvement," *Current Opinion in Plant Biology*, vol. 54, pp. 34 – 41, 2020, genome studies and molecular genetics.

[22] T. Pook, J. Freudenthal, A. Korte, and H. Simianer, "Using local convolutional neural networks for genomic prediction," *bioRxiv*, 2020.

[23] E. H. Mahood, L. H. Kruse, and G. D. Moghe, "Machine learning: A powerful tool for gene function prediction in plants," *Applications in Plant Sciences*, vol. 8, no. 7, p. e11376, 2020.

[24] J. Wang, A. Ma, Y. Chang, J. Gong, Y. Jiang, H. Fu, C. Wang, R. Qi, Q. Ma, and D. Xu, "scGNN: a novel graph neural network framework for single-cell RNA-Seq analyses," *bioRxiv*, 2020.

[25] Y. Gao, X. Chen, H. Qiao, Y. Ke, and H. Qi, "Low-bias manipulation of DNA oligo pool for robust data storage," *ACS Synthetic Biology*, vol. 9, no. 12, pp. 3344–3352, 2020, pMID: 33185422.

[26] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucleic Acids Research*, vol. 16, no. 22, pp. 10 881–10 890, 11 1988.

[27] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon, "Biopython: freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 03 2009.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[32] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602 – 610, 2005, iJCNN 2005.

[33] A. Galstyan and P. R. Cohen, "Empirical comparison of "hard" and "soft" label propagation for relational classification," in *International Conference on Inductive Logic Programming*. Springer, 2007, pp. 98–111.