NUCLEIC TRANSFORMER: DEEP LEARNING ON NUCLEIC ACIDS WITH SELF-ATTENTION AND CONVOLUTIONS

Shujun He
Department of Chemical Engineering
Texas A&M University
College Station, TX
shujun@tamu.edu

Baizhen Gao
Department of Chemical Engineering
Texas A&M University
College Station, TX
baizhen@tamu.edu

Rushant Sabnis
Department of Chemical Engineering
Texas A&M University
College Station, TX
rushantsabnis@tamu.edu

Qing Sun
Department of Chemical Engineering
Texas A&M University
College Station, TX
sunqing@tamu.edu

ABSTRACT

Much work has been done to apply machine learning and deep learning to genomics tasks, but these applications usually require extensive domain knowledge and the resulting models provide very limited interpretability. Here we present the Nucleic Transformer, a conceptually simple but effective and interpretable model architecture that excels in a variety of DNA/RNA tasks. The Nucleic Transformer processes nucleic acid sequences with self-attention and convolutions, two deep learning techniques that have proved dominant in the fields of computer vision and natural language processing. We demonstrate that the Nucleic Transformer can be trained in both supervised and unsupervised fashion without much domain knowledge to achieve high performance with limited amounts of data in *E. coli* promoter classification, viral genome identification, and degradation properties of COVID-19 mRNA vaccine candidates. Additionally, we showcase extraction of promoter motifs from learned attention and how direct visualization of self-attention maps assists informed decision making using deep learning models.

***K*eywords** Genomics · mRNA Vaccine Degradation · Deep learning · Natural Language Processing · Neural Network · Bioinformatics · COVID-19 mRNA

1 Introduction

DNA and RNA are essential components of life. Information stored in DNA and RNA provides instructions on the complex functions should in biological organisms [1]. However, understanding DNA and RNA, including the stored information and degradation properties, has always been challenging because of the complex and large potential sequence space of DNA and RNA. Especially, the functions and properties of many coding and non-coding DNA/RNA sequences still remain poorly understood [2, 3].

Deep learning is a class of data-driven modeling approaches that have found much success in many fields including image recognition [4], natural language processing [5, 6], and computation biology [7]. It has allowed researchers to efficiently predict the function, origin, and properties of DNA/RNA sequences by training neural networks on large datasets [8, 9, 10, 11, 12, 13]. The sequential nature of DNA/RNA has made recurrence (RNN/LSTM/GRU) appealing for models dealing with such data [14, 15, 16, 17, 18]. However, the sequential computation is difficult to parallelize and objects at long distances suffer from vanishing gradients. Transformers, on the other hand, is a recently proposed

architecture that solely relies on attention mechanisms that can model dependencies regardless of the distance in the input or output sequences [5]. Although transformer has been adopted in many natural language processing tasks, we have found few studies using transformer in biological systems such as DNA/RNA sequences.

In this study, we propose a model architecture Nucleic Transformer that utilizes convolution and self-attention to capture both local and global dependencies, which enable the model to achieve high accuracy and provide interpretability for three DNA/RNA tasks. The Nucleic Transrformer formulates DNA understanding as natural language processing tasks, and RNA degradation as molecular prediction tasks. First, Nucleic Transformer is trained to classify short pieces of DNA sequence (81 bp) as either an *E. coli* (Escherichia coli) promoter sequence or non-promoter sequence. Nucleic Transformer learns from labeled promoter/non-promoter sequences, and the classification accuracy is compared with other state-of-the-art promoter identification models [19, 20, 21]. Second, Nucleic Transformer is tested on longer DNA sequences (300 bp) for classification of viral and non-viral sequences. We show that Nucleic Transformer predicts both viral and non-viral sequences with improved accuracy compared with a previous best computational model [22]. In addition, we demonstrate that a Nucleic Transformer variant can be trained to predict the RNA degradation rates at each position of a given sequence, a task that few computational models have tackled but of great importance especially under current circumstances due to the recent COVID-19 pandemic. All training code for reproducible results is released at `https://github.com/Shujun-He/Nucleic-Transformer`.

2 Methods

The Nucleic Transformer architecture combines convolutions and self-attention to capture both local and global dependencies (Figure 1), and can be trained in both supervised and unsupervised fashion to make predictions per sequence and per nucleotide. Its flexible nature also allows injection of information from biophysical models. The self-attention mechanism explicitly models pairwise interactions and provides interpretability that allows more informed decision making. In this section, we first describe the benchmark datasets used, then explain our architecture design, and finally detail the training procedures.

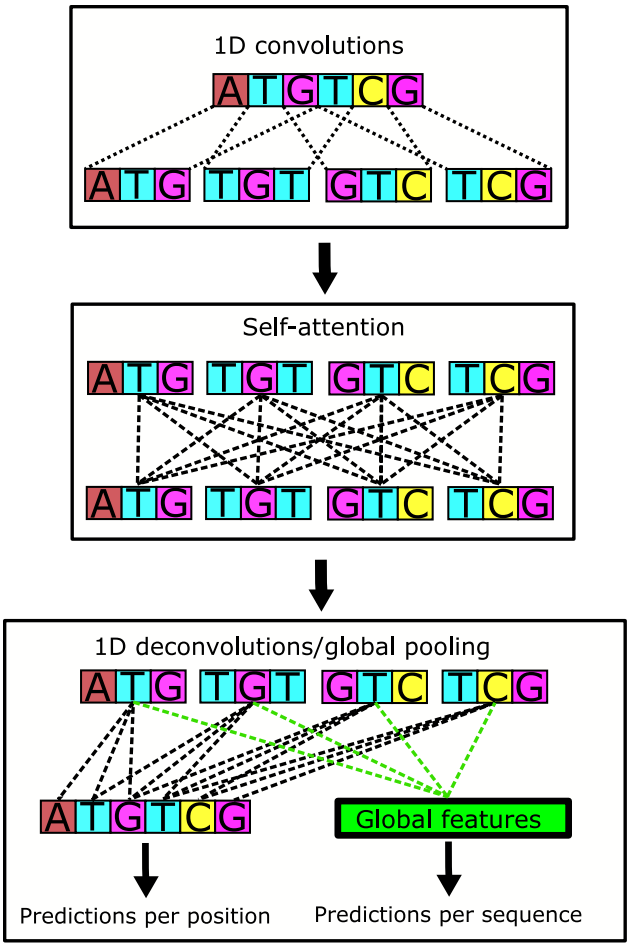

Figure 1: Overview of the Nucleic Transformer architecture.

2.1 Benchmark datasets

High quality and accessible datasets are imperative in advancing our understanding of DNA and RNA, not only because of the useful information they provide but also because they establish benchmarks where fair comparisons between different models can be made. In this work, we benchmark with 3 high quality and easily accessible datasets, which include a labeled *E. coli* promoter/non-promoter dataset, a labeled viral/non-viral genome dataset, and an RNA dataset of COVID-19 vaccine candidates with experimentally measured degradation properties. The *E. coli* promoter/non-promoter dataset and the labeled viral/non-viral genome dataset have been well studied in the literature [19, 20, 23, 24, 22], while the RNA dataset is extremely novel and little existing work has dealt with RNA degradation properties except for very recent works [25].

2.1.1 *E. coli* promoter/non-promoter dataset

The *E. coli* promoter/non-promoter dataset is an experimentally confirmed benchmark dataset widely used in the literature to model and evaluate DNA promoter sequences [26]. All DNA sequences in the dataset were collected from RegulonDB, and sequences were screened by CD-HIT based on redundant sequence identity [27]. This dataset consists of 2,860 promoter sequences and 2,860 non-promoter sequences. All promoter sequences were experimentally confirmed and collected from RegulonDB (version 9.3) [28]. The non-promoters sequences were extracted randomly from the middle regions of long coding sequences and convergent intergenic regions in *E. coli* K-12 genome [29, 30]. This dataset can be freely downloaded at github. Model performance on this dataset is evaluated using 5-fold cross validation, where the data is split using iterativte stratification [31]. The metrics used for this dataset are accuracy, sensitivity, specificity, and Matthews correlation coefficient (MCC). In addition to cross-validation, we also use an independent test set composed of experimentally verified *E. Coli* promoters recently added to RegulonDB.

2.1.2 Viral/non-viral genome dataset

The viral/non-viral genome dataset is same as the one used to trained viraminer [22], which consists of 19 different NGS experiments analyzed and labeled by PCJ-BLAST [32] following de novo genome assembly algorithms. This dataset is publicly available at `https://github.com/NeuroCSUT/ViraMiner`. DNA sequences included in this dataset were cut to 300 bp segments with remaining portions smaller than 300 bp discarded. Further, all sequences that contain "N" (unknown with equal probability to any of the four nucleotides) were removed as well. This dataset has approximately 320,000 DNA sequences in total. The main challenge with this dataset is the class imbalance, where only 2% of sequences are of viral origin. The dataset is split into training, validation, and test, where hypertuning was done with the training and validation set, and model performance evaluated on the test set. The metric used for this dataset is AUC (Area Under the Receiver Operating Characteristic Curve).

2.1.3 OpenVaccine challenge dataset

The COVID-19 pandemic has led to a chaotic 2020, and development of effective vaccines remains a challenge of the highest priority. mRNA is the leading candiate for COVID-19, but they face significant limitations, one of which is the spontaneous degradation of mRNA. The OpenVaccine challenge [33] hosted by the Eterna community sought to rally the data science expertise of Kaggle competitors to develop models that could accurately predict degradation of mRNA. During the 21-day challenge, competitors were provided with 2400 107-bp mRNA sequences with the first 68 base pairs labels with 5 degradation properties at each position. These properties are reactivity, deg_pH10, deg_Mg_pH10 , deg_50C , and deg_Mg_50C. More details on these properties can be found at `https://www.kaggle.com/c/stanford-covid-vaccine/data`.

Like most Kaggle competitions, the test set was divided into a public test set and a private test set. Results on the public test set was available during the competition, while private test set results were hidden. The final evaluation was done on a portion of the private test set consisting of 3005 130-bp mRNA sequences, whose degradation measurements were conducted during the 21-day challenge and revealed at the end. The test set was subjected to screening based on three criteria:

1. Minimum value across all 5 degradation properties must be greater than -0.5
2. Mean signal/noise across all 5 degradation properties must be greater than 1.0. [Signal/noise is defined as mean(measurement value over 68 nts)/mean(statistical error in measurement value over 68 nts)]
3. Sequences were clustered into clusters with less than 50% sequence similarity and chosen from clusters with 3 or fewer members

After screening, only 1172 sequences remained on the test set. Final evaluation was done on 3 of the 5 properties (reactivity, deg_Mg_pH10, and deg_Mg_50C). Unlike the training set, the test set has longer mRNA sequences with more sequence diversity and more measurements (first 91 positions) per sequence; in fact, more predictions had to be made for the test set than there were training samples. The OpenVaccine challenge is thus the most challenging among the tasks tackled in this paper. The metric used for ranking in the competition is MCRMSE (mean columnwise root mean squared error):

$$\text{MCRMSE} = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij}) - \hat{y}_{ij})^2}, \tag{1}$$

where N_t is the number of columns, n the number of positions predicted, y the ground truth, and $\hat{y}$ the predicted value. In addition, we also use R2 score (coefficient of determination) during further analysis.

2.2 Architecture design

In this section, we discuss the theoretical considerations in architecture design. For DNA tasks, the Nucleic Transformer is akin to a language model, where kmers are encoded using 1D convolutions and a transformer encoder stack is used to model pairwise global dependencies. For the RNA task, we modified the Nucleic Transformer by leveraging additional inputs from biophysical models and adding deconvolution layers, which enable predictions per nucleotide.

2.2.1 DNA as a language & RNA as a molecule

Any DNA seqeuence is a series of nucleotides, each of which can be one of A (adenosine), C (cytidine), G (guanosine), and T (thymine). Therefore, if we consider DNA a langauge that uses only four different characters to encode information, we can model it in similar fashion as a natural langauge. This idea then allows us to agnostically apply Natural Language Processing (NLP) techniques in the domain of deep learning without injection of domain specific knowledge in bioinformatics.

Although there are similarities between DNA sequences and natural language, there are also some important differences. Take the English language for example, all words in the vocabulary are combinations of the 26 letters in the alphabet; similarly, a DNA sequence is a sequence of 4 nucleotides. Nevertheless, the English langauge not only contains letters but also spaces that separate words and commas and periods that seperate sentences, whereas comparatively a DNA sequence is simply a sequence of 4 nucleotides. Further, when humans interpret a sentence, words are discretely recognized and each word can be considered a discrete object. As a result, state of art NLP models evaluate languages as collections of words (and punctuations) instead of letters. Since a DNA sequence does not have punctuations, we need to find a way to transform a DNA sequence into "words", which we will discuss in detail in the next section.

Unlike DNA which is double stranded and relatively stable, RNA is single-stranded and highly promiscuous. While double-stranded DNA forms hydrogen bonds between its complementary bases, single-stranded RNA forms secondary structures by itself. Therefore, it is reasonable to regard DNA as an *information storage device*, while RNA should be seen as a *molecule* in the context of mRNA degradation predictions. This simple realization suggests we can use existing biophysical models to inject biophysical knowledge into our deep learning models.

2.2.2 K-mers with 1-D convolutions

A common method to process DNA sequences in the field of bioinformatics is to transform them into k-mers. For instance, consider a short DNA sequence ATGC. The k-mers in this sequence are subsequences of length k, so ATGC has three 2-mers AT, TG, and GC, two 3-mers ATG and TGC, and one 4-mer ATGC. Converting a DNA sequence into k-mers is analogous to separating a language sentence into words and allows for more efficient extraction of information from DNA.

The extraction of k-mers from a DNA sequence can be considered a sliding window of size k taking snapshots of the sequence while moving one position at a time from one end of the sequence to the other, which is conceptually identical to the convolution operation used in deep learning. Consider a simple example of convolution involving a vector $S \in \mathcal{R}^l$, where l is the length of the vector, and a convolution kernel $K \in \mathcal{R}^3$, which convolves over the vector S. If the convolutional kernel strides one position at a time, an output vector of dot products $O \in \mathcal{R}^{l-2}$ is computed,

$$O_p = \sum_{i \in \{0,1,2\}} K_i S_{p+i}, \tag{2}$$

where p denotes the position in the output vector. In this case, the convolution operation aggregates local information with 3 positions at a time, so if S is a sequence of nucleotides, the convolution operation is essentially extracting 3-mers from the DNA sequence S. Consequently, it is conceptually clear that a convolution kernel of size k can be used to transform a DNA sequence into k-mers. In the next paragraph, we will explain further how this works mathematically.

Since our model takes sequences of DNA/RNA nucleotides as input, we first transform each nucleotide into embeddings of fixed size d_{model}. So now for each sequence we have a tensor $I \in \mathcal{R}^{l \times d_{model}}$, where l is the length of the sequence. Because we are using the transformer encoder architecture, which is permutation invariant, we need to add positional encoding, same as the implementation in [5]:

$$PE_{(pos,2i)} = sin(pos/5000^{2i/d_{model}}) \tag{3}$$

$$PE_{(pos,2i+1)} = sin(cos/5000^{2i/d_{model}}), \tag{4}$$

where pos is the position and i is the channel dimension. Now to create k-mers we perform convolutions on the tensor I without padding and stride = 1. When the convolution operation with kernel size k is performed over I, a new tensor $K_k \in \mathcal{R}^{(l-k+1) \times d_{model}}$ representing the sequence of k-mers is generated. Now we obtain a collection of k-mers, where each k-mer is represented by a feature vector of size d_{model}. The 1D convolution layers are always followed by a layer normalization layer [34].

Indeed our representation of k-mers deviates from conventional representation of words in deep learning, where each word in the vocabulary directly corresponds to a feature vector in a look up table. The disadvantage of using look up

tables for k-mers is that a very small percentage of all possible k-mers are present in any dataset, and there is no way for the network to generalize to unseen k-mers. For instance, if a common promoter motif TATAAT appears in the dataset but a similar motif TATATT does not, there is no way for the network to generalize to the unseen motif TATATT, but by using convolutions, we make it easy for the network to recognize that there is not much difference between TATAAT and TATATT, leading to better generalization. Additionally, embeddings of k-mers of larger sizes require a prohibitively large amount of parameters, since the total possible amount of k-mers for a given k is 4^k.

2.2.3 Transformer encoder

For the encoder, we implement the vanilla transformer encoder [5], which uses the multi-head self-attention mechanism. Unlike a single-head attention function, the multi-head self-attention function linearly projects d_{model}-dimensional keys, values and queries into lower dimensional representations. Then the multi-head attention function directly operates on the entire sequence. It has been posited that the multi-head mechanism allows different heads to learn different hidden representations of the input, leading to better performance. The multi-head self-attention mechanism can be summarized in a few equations:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \tag{6}$$

$$\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{7}$$

Since we are only using the transformer encoder, Q, K, V come from the same sequence of feature vectors (hence the name self-attention), each of which represents a k-mer with positional encoding.

The self-attention mechanism enables each k-mer to attend to all k-mers (including itself), so global dependencies can be drawn between k-mers at any distance. Contrary to recurrence and convolutions, both of which enforce sparse local connectivity, transformers allow for dense or complete global connectivity. The ability to draw global dependencies of transformers is a huge advantage over recurrence and convolutions, both of which struggle with long sequences.

The self-attention function is followed by a position-wise feedforward network applied separately and identically to each position:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b1)W_2 + b_2. \tag{8}$$

The position-wise feedforward network is basically two linear transforms with a ReLU activation in between. Conventionally, the combination of self-attention and position-wise feedforward network is referred to as the transformer encoder layer, and a stack of transformer encoder layers is referred to the transformer encoder.

2.2.4 Incorporating biophysical models

Although we found it sufficient to simply use sequence information for DNA tasks, predicting degradation of RNA requires more than just sequence information. Firstly, we include the predicted structure per position, which describes whether a nucleotide is paired or unpaired with another one via hydrogen bonding. Also, we include the predicted loop type assigned by bpRNA. Additionally, we directly add a modified version of the base-pairing probability matrix into the attention function:

$$\text{Attention}(Q, K, V, M_{bpp}) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + \gamma M_{bpp})V \tag{9}$$

where γ is a learnable parameter and M_{bpp} is the modified base-pairing probability matrix. The original base-pairing probability matrix contains the probabilities for every possible base pair in an RNA sequence and has been used for many RNA informatics tasks. Here in addition to base-pairing probabilities, we also stack inverse, inverse squared, and inverse cubed pairwise distance matrices on top of the original base-pairing probability matrix, where the distance is the the number of covalent bonds between the pair of nucleotides (this can also be considered the path length in an RNA graph where the only edges are the covalent bonds). The inverse distance matrices encode some information about the relative distance between pairs of nucleotides, since pairs of nucleotides with a small number of covelent bonds in between are likely to be closer to each other spatially. Because the distance matrix already encodes information about position, we do not use positional encoding for mRNA.

Because 1-D convolution operation used in the Nucleic Transformer does not use padding, the convolution product ends up with reduced dimensionality in the L dimension when the convolution kernel size is bigger than 1. As a result,

the base pairing probability matrix cannot be directly added to self-attention matrix. To circumvent this, we also do 2D convolution with the same kernel size as the 1D convolution on the modified base pairing probability matrix without padding, so the dimensionality of the feature map becomes $C \times (L-k+1) \times (L-k+1)$. The attention function now is:

$$\text{Attention}(Q, K, V, M_{bpp}) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}} + \gamma \text{conv2d}(M_{bpp}))V. \tag{10}$$

Conceptually, instead of a base pair to base pair interaction mapping, the 2D convolution product of the modified base pairing probability matrix can be seen as a kmer to kmer pairwise interaction mapping with matching dimensionality to the 1D convolution kmer products. Aside from matching dimensionality, the 2D convolution operation also makes up for some missing information regarding the geometry of mRNA folding. To illustrate this, we visualize an mRNA sequence in the OpenVaccine dataset to explain the physical and mathematical reasoning behind the 2D convolution operation (Figure 2). While inspecting the interaction between A-20 (A at position 20), G-21, C-40, and U-41, we can visually see that A-20 and C-40 are quite close to each other and imagine that there is some degree of interaction between them, despite A-20 and C-40 not forming hydrogen bonds. However, looking at the portion of BPP matrix and distance matrix corresponding to the 2x2 connection between A-20 (A at position 20), G-21, C-40, and U-41, we see that neither the BPP matrix nor the distance matrix convey this information, as the component (40,20) has zero or close to zero values on both the BPP matrix and the distance matrix. When a 2x2 convolution kernel operates on the BPP matrix and distance matrix (for illustration purposes here we simply draw a kernel with all values set to unity), it essentially fuses the 4 connections between A-20, G-21, C-40, and U-41, and creates a strong connection between the 2 2mers (A-20, G-21 and C-40, U-41). Now it becomes much easier for the network to learn the interaction between A-20 and G-40 (as well as for G-21 and U-41).

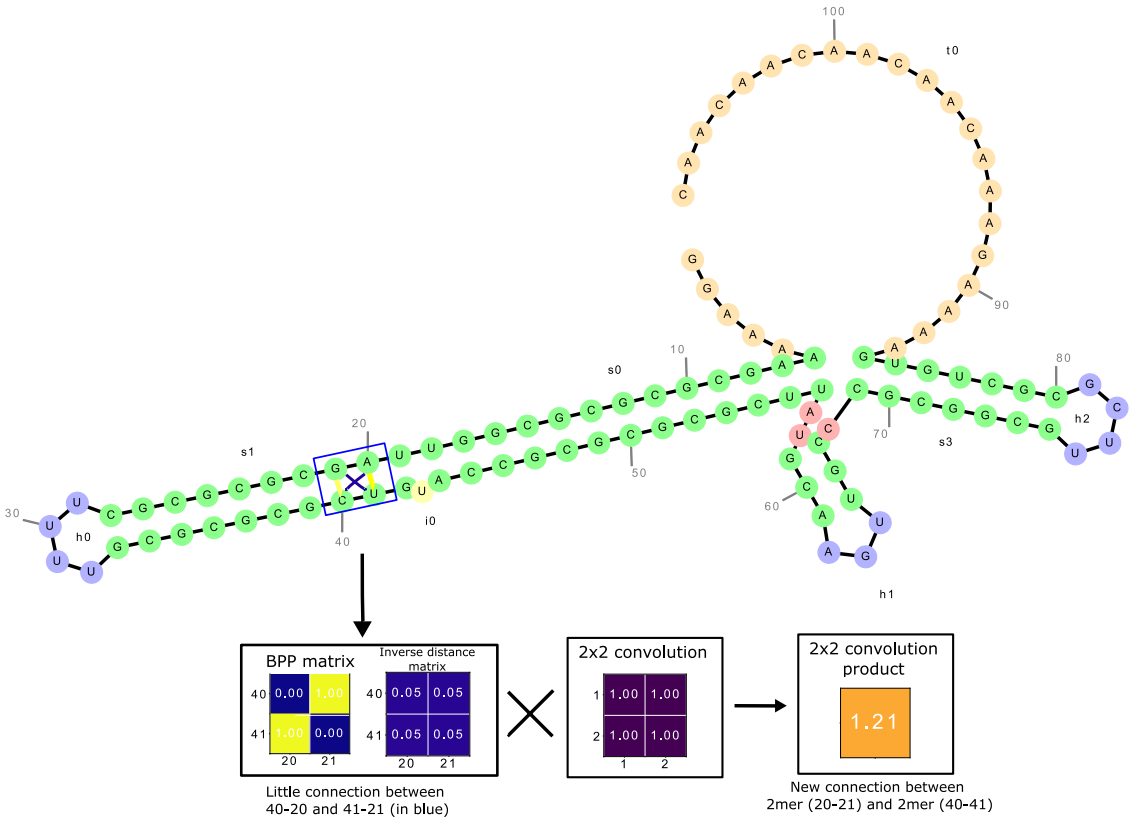

Figure 2: Physical meaning of 2D convolution on BPP matrix illustrated. Here we visualize the folding of sequence id_0049f53ba.

The combination of convolution and self-attention cannot produce nucleotide position wise predictions, since it generates kmer encodings instead single base pair encoding. In order to make predictions per nucleotide position, we introduce

additional deconvolution layers to retrieve full dimensional encodings, which allow residual connections of both 1D and 2D encodings before and after the transformer encoder. As a result, both the single nucleotide embeddings and the modified BPP matrix go through deep transforms before outputting predictions.

Now we can summarize the modified Nucleic Transformer architecture used for the RNA task (Figure 3), which can be seen as a special case of a series of multiple Nucleic Transformers with a single transformer encoder layer followed by a deconvolution layer. Also, because the OpenVaccine challenge requires making predictions at each position of the RNA sequence, it is important for the last transformer encoder layer right before outputting predictions to operate on single nucleotide encodings instead of kmer encodings. With these considerations in mind, we choose a simple strategy to construct the stack of Nucleic Transformers with two main hyperparameters k and n_{layer} set equal to each other (Figure3). The first single layer Nucleic Transformer has $k = n_{layer}$ and we decrease the size of the convolution kernel by 1 for the next single layer Nucleic Transformer. Therefore, when we get to the last Nucleic Transformer in the stack, k becomes 1 and the last Nucleic Transformer is simply a transformer encoder layer with an added bias from the BPP feature map.

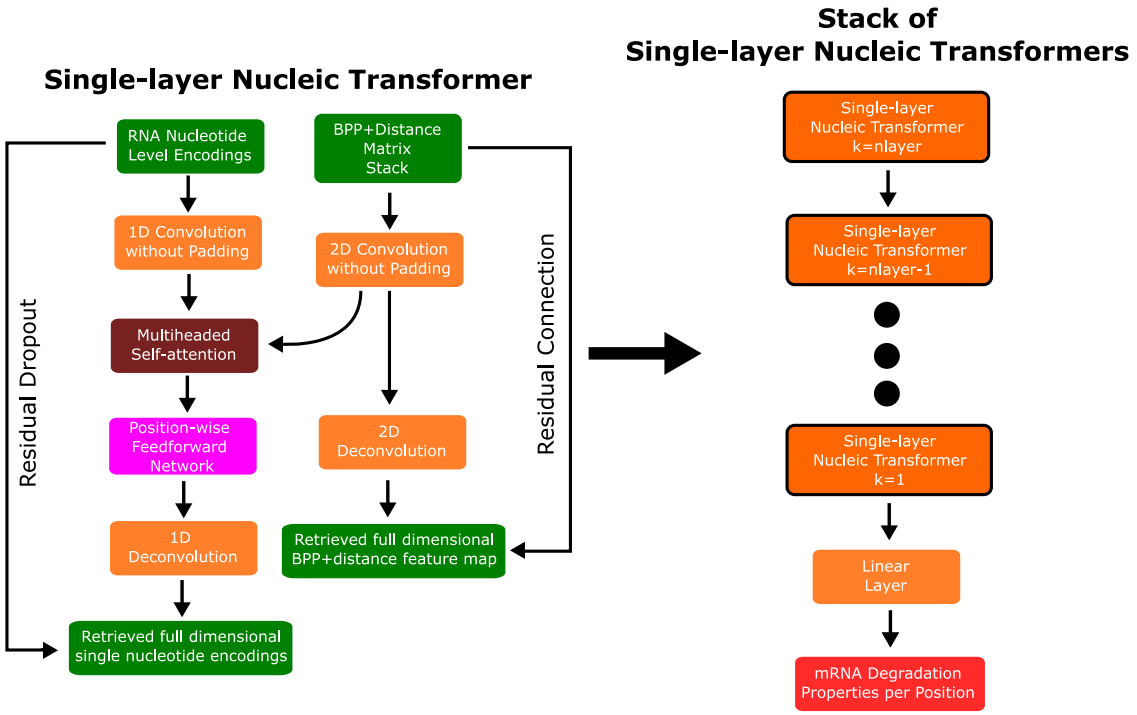

Figure 3: Nucleic Transformer stack which takes advantage of additional input information from biophysical models.

2.3 Training details

Training of transformers can be tricky and wrong selection of training schedule and hyperparamters can be lead to diverged training [35], so here we detail the training process.

2.3.1 Optimizer and training schedule

For DNA classification tasks, we choose Adam [36], a commonly used optimizer in deep learning with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 1e-8$. Weight decay is set to 1e-5. Our learning rate schedule is a stepwise inverse square root decay schedule with warm up. Since we use relatively small batch sizes during training, we adjust the learning rate by a scaling factor C:

$$\text{learning rate} = C \cdot d_{model}^{-0.5} \cdot \min(\text{step_num}^{-0.5}, \text{step_num} \cdot \text{warmup_steps}^{-1.5}). \tag{11}$$

In our experiments, C is set 0.1 and warmup_steps is set to 3200. Additionally, we use dropout [37] of probability 0.1 in all attention layers, fully connected layers, and positional encoding.

For the RNA task, we found Adam to be underfitting. Therefore, we switched to a more recent and powerful optimizer, Ranger, which uses gradient centralization from `https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer` [38]. As for the training schedule, we used flat and anneal, where training starts with a flat learning rate of 1e-3 and then 75% through all the epochs training proceeds with cosine annealing schedule reducing learning rate down to 0 at the end of training. Weight decay is set to 0.1.

2.3.2 Error weighted loss

Because the OpenVaccine dataset came from experimental measurements that had errors, we adjusted the losses based on the error for each measurement during supervised training:

$$\text{error adjusted loss} = \text{loss} \times (\alpha + e^{\beta \times \text{error}}), \tag{12}$$

where α and β are tunable hyperparameters. If α is set to 1 and β to infinity, then the loss values stay the same; otherwise gradients from measurements with large errors would be lowered to prevent the neural network from overfitting to experimental errors. For the OpenVaccine dataset, we use $\alpha = 0.5$ and $\beta = 5$.

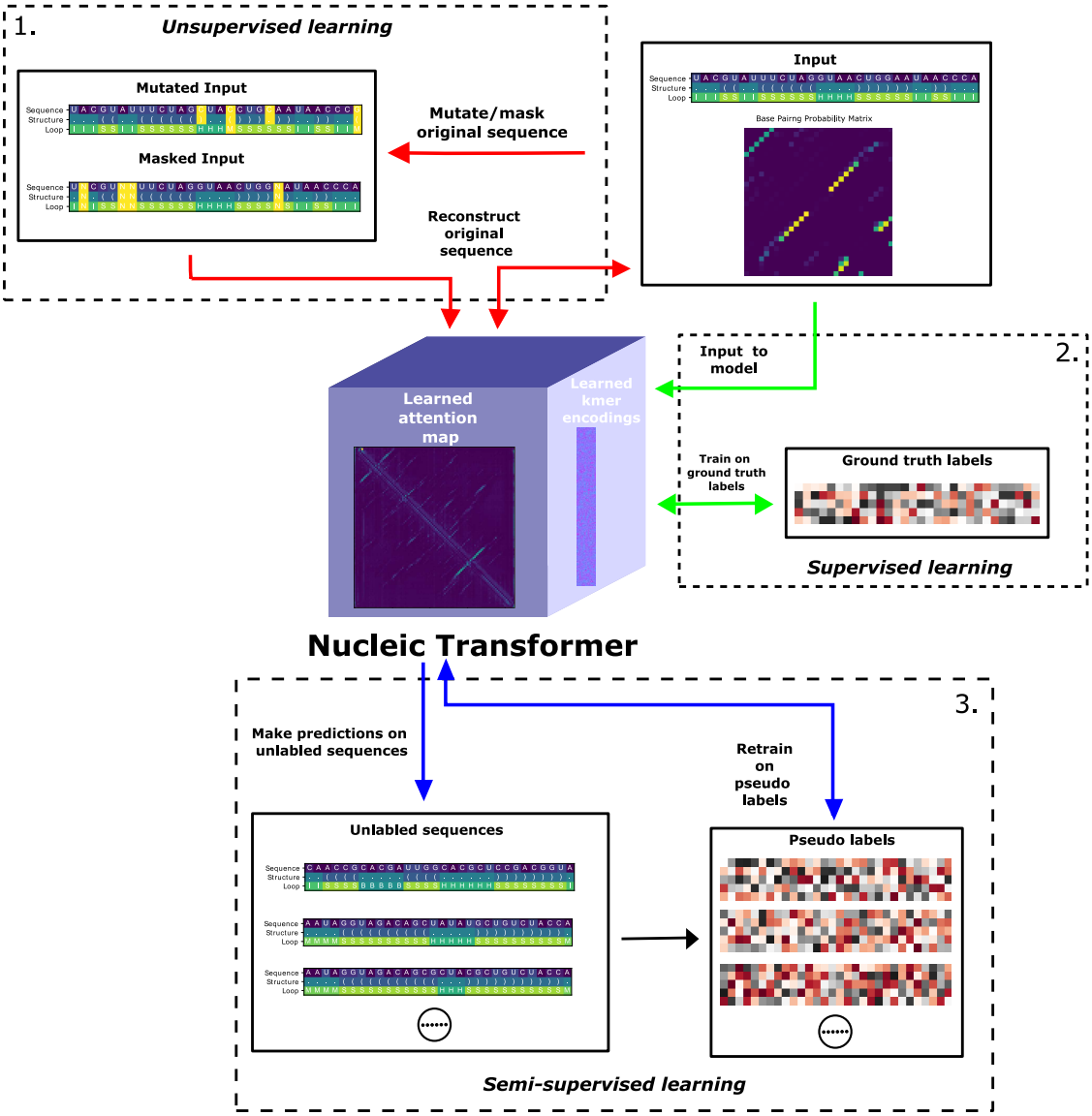

Figure 4: The Nucleic Transformer can be trained in unsupervised, supervised, and semi-supervise fashion.

2.3.3 Learning

Since in the OpenVaccine challenge, the number of samples in the test set exceeds that in the training set, we use a combination of learning methods (Figure 4). Here we describe those learning methods.

Multitasking learning: during pretraining, mutated/masked sequence, structure, and predicted loop type are inputted into the Nucleic Transformer and then the Nucleic Transformer is trained with crossentropy loss to retrieve the correct sequence, structure, and predicted loop type simultaneously at each position of the RNA sequence. During training on ground truth labels and pseudo labels, the Nucleic Transformer is trained to predict 5 different degradation properties simultaneously at each measured position of the RNA sequence.

Unsupervised learning: although we did not use pretraining for our DNA tasks, we used all available sequences in the OpenVaccine challenge dataset to pretrain our network on randomly mutated and masked (with NULL token) sequence retrieval loss (basically softmax to retrieve correct nucleotide/structure/loop). During pretraining, the Nucleic Transformer learns the rules of RNA structure, guided by biophysical knowledge provided by biophysical models.

Supervised/semi-supervised learning: during supervised learning, the Nucleic Transformer is trained on target values of classification classes or RNA degradation properties. Following RNA supervised learning, the Nucleic Transformer was retrained in semi-supervised fashion on pseudo labels generated by an ensemble of Nucleic Transformers with different depths. Similar to previous work with semi-supervised learning [39], we retrain the models first using pseudo labels at a flat learning rate and then finetune with ground truth labels in the training set with cosine anneal schedule.

2.3.4 Usage of biophysical models

We used secondary structures predicted by an ensemble of biophysical models including RNAsoft [40], rnastructure [41], CONTRAfold [42], EternaFold [43], NUPACK [44], and Vienna [45]. Arnie `https://github.com/DasLab/arnie` is used as a wrapper to generate secondary structure predictions. For each sequence, we generated secondary structure predictions at 37 and 50 Celsius, since two of the scored degradation properties were measured at different temperatures. Although we also need to make predictions for a degradation property at pH10, none of the biophysical models used could generate predictions at different pH's. With 6 packages, we ended with up with 12 secondary structure predictions for each sequence. During training, we randomly select one of the 12 secondary structure predictions for each sample during a forward and backward propagation pass. During validation and testing, we use the averaged predictions made used all 12 secondary structure predictions.

2.3.5 Random mutations during training

There is no question that the transformer architecture is extremely powerful. In fact, it has been shown that the transformers can learn from the English Wikipedia which contains billions of words in an unsupervised fashion and achieve state-of-the-art results by finetuning on task specific data [6]. It is known that deep learning models can perfectly memorize completely random data and labels, so to combat the memorization effect, we inject noise artificially by randomly mutating positions in the source DNA/RNA sequence before forward and backward propagation during training, similar to bert's pretraining [6]. This injection of random noise is done during DNA supervised learning and RNA unsupervised learning. Note that in all our experiments, we simply randomly mutate nucleotides in randomly selected positions, and because we do not ensure nucleotide at each selected position is changed, the average amount of mutations is 3/4 of n_{mute}. It is true that these random mutations could be non-label-preserving; however, deep learning algorithm is robust to massive label noise, so the non-label-preserving mutations should simply be ignored by the network during training [46]. The number of positions to randomly mutate is a hyperparamter that we denote n_{mute}, with which we experiment to find the best value.

3 Results & discussion

3.1 Promoter/non-promoter classification

In this section, we first compare the performance of the Nucleic Transformer with the best hyperparameters on *E. coli* promoter classification with other top results in the literature. Then we explore how a few important hyperparameters affect the accuracy of the results, followed by visualization and analysis of attention weights.

3.1.1 Comparison with top results in the literature

We found the Nucleic Transformer to outperform previous results in the literature in *E. coli* promoter classification (Table 1). Compared to non-deep learning approaches which use sophisticated hand crafted features, the Nucleic Transformer leads in accuracy by at least 1.6% or more. A more recent model, iPromoter-BnCNN, which also employs structural property properties of DNA such as stability, rigidity, and curvature, is similar in performance to the Nucleic Transformer, although the Nucleic Transformer directly makes predictions from sequence information. The best results were obtained using $k = 7$, $n_{mute} = 15$, six transformer encoder layers, $d_{model} = 256$, $n_{head} = 8$, and batch size is 24.

	Accuracy	MCC	Sensitivity	Specificity
Nucleic Transformer	**0.8831**	**0.7665**	**0.8935**	0.8727
iPromoter-BnCNN [19]	0.882	0.763	0.883	**0.88**
MULTiPly [20]	0.8668	0.7224	0.8656	0.8668
iPromoter-2L2.0 [23]	0.8498	0.6998	0.8413	0.8584
iPromoter-2L1.0 [24]	0.8168	0.6343	0.792	0.8416

Table 1: Performance of Nucleic Transformer against top results in the literature based on accuracy, MCC, sensitivity, and specificity.

On the independent test set (Table 2), which includes recently released experimentally verified promoter samples, the Nucleic Transformer is more accurate than MULTiPly and iPromoter-2L, while only slightly more accurate (by 1 sample) than iPromoter-BnCNN.

	TP	FN
Nucleic Transformer	246	10
iPromoter-BnCNN	245	11
MULTiPly	238	18
iPromoter-2L	238	18

Table 2: Performance of Nucleic Transformer on the independent test set against top results in the literature.

3.1.2 K-mers of different lengths

To experiment with k-mers of different lengths, first we set a baseline with no k-mer aggregation, where only single nucleotides with positional encoding are inputted into the transformer. Then we incrementally increase k, and evaluate its effect on cross validation accuracy (Figure 5). Additionally, we also run experiments without the transformer encoder to see the improvement the transformer encoder gives, in which case the k-mer aggregation layer is simply followed a global-maxpooling and fully connected layer. Here we used $n_{mute} = 15$, six transformer encoder layers, $d_{model} = 256$, $n_{head} = 8$, and batch size is 24.

The model's performance improves initially upon using longer k-mers, and then saturates to a point where using longer k-mers no longer increases performance. We see that aggregating larger kmers (k=8,9) actually led to a decrease in performance, likely due to overfitting resulting from large convolution kernels. Further, we note that the transformer encoder consistently gives a significant boost in accuracy compared to the counterpart without transformer encoder.

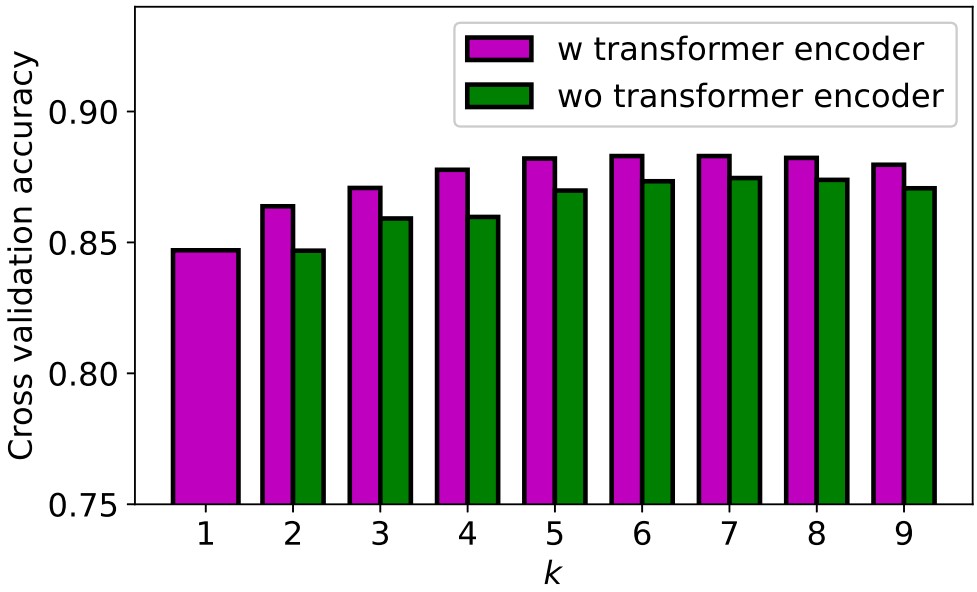

Figure 5: Bar chart of accuracy vs the parameter k.

3.1.3 Number of random mutations during training

Much like most tunable hyperparameters in deep learning, we see from our experiments that there is an optimal value for the number of random mutations during training (Figure 6). Here we used six transformer encoder layers, $d_{model} = 256$, $n_{head} = 8$, and batch size is 24. For experiments without the transformer encoder, we simply took out the transformer encoder while keeping everything else the same. Our experiments show that the model tends to overfit to the training data when no random mutations are added during training. Without mutations, the model quickly converges to 1.0 accuracy and close to 0 loss on the training set, indicating that the model has simply memorized the training data. Additionally, too many mutations lead to underfitting, as there is too much random noise during training leading to poor convergence.

3.1.4 Model behavior on randomly generated DNA sequences

It has been experimentally shown that approximately 10% random sequences can function as promoters [47]. Here we screened one million randomly generated DNA sequences of length 81, and classified them with our trained model (Figure 7). We found that despite a balanced distribution of positive and negative examples in the promoter/non-promoter dataset, our model recognized approximately 65% of random DNA sequences as non-promoters. This suggests that our model has some mechanism to guard against random sequences and hence is likely able to generalize well.

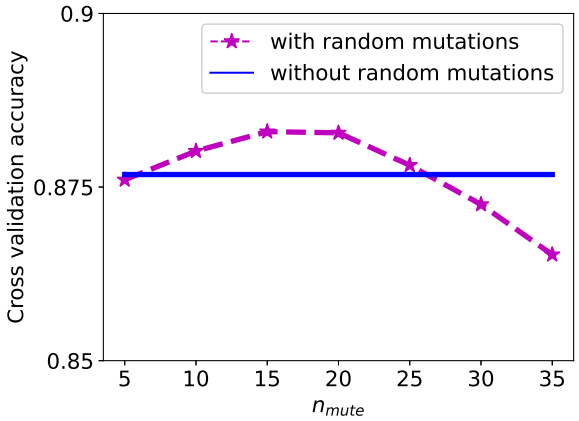

Figure 6: Cross validation accuracy vs number of random mutations during training.

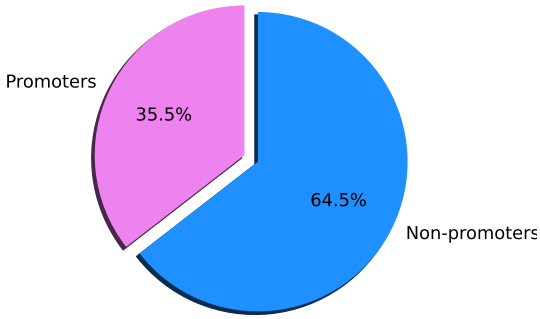

Figure 7: Percent of one million randomly generated sequences classified as promoters/non-promoters by the Nucleic Transformer.

3.1.5 Analysis of attention weights and motif extraction

By visualizing the attention weight matrix, we see that the Nucleic Transformer often focuses on kmers that resemble consensus promoter motifs (Figure 8). We can also extract motifs the Nucleic Transformer considers the most characteristic of promoters (Figure 9). For each correct prediction of promoters in the validation sets of 5-fold cross validation, we take the column-wise sum of the attention weight matrix and rank the kmers. Then we simply count the amount of times each kmer receives top 3 most attention in all correct predictions. We found the kmers that frequently appear in top 3 resemble the consensus promoter motif TATAAT. In fact, two of the 10 most frequently 7-mers have the exact consensus motif TATAAT, while 5 others contain the motif TAAAAT, TATCAT, TATTAT, TATATT, TATACT, all of which are one mutation away from the consensus motif TATAAT. Deep learning has been criticized as black boxes that cannot be interpreted; however, here we demonstrate that the Nucleic Transformer can be interpreted and useful motifs can even be derived from attention weights directly.

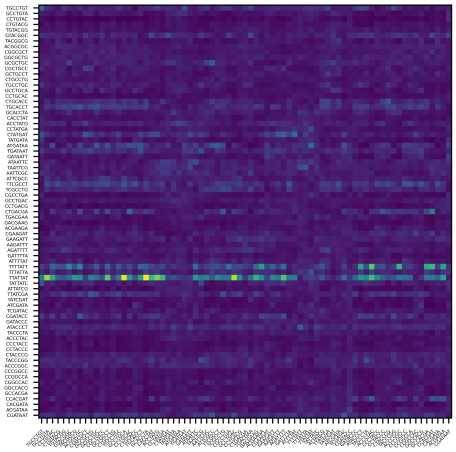

Figure 8: Visualization of an attention weight right before outputting a prediction. A right band can be seen for the 7-mer TTATTAT, which is one mutation away from TATAAT.

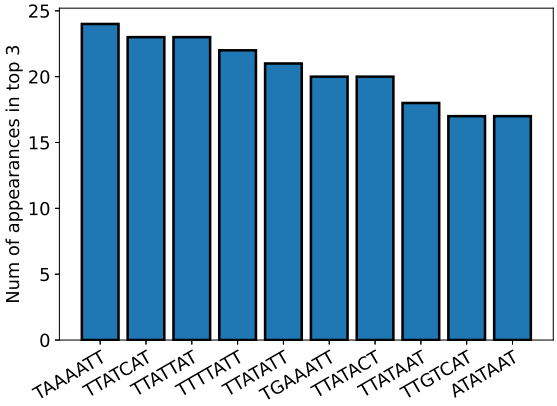

Figure 9: Most important 7-mers in promoter classification based on analysis of attention weights.

3.2 Viral/non-viral classification

To further demonstrate the effectiveness of the Nucleic Transformer architecture, we trained Nucleic Transformer models on a viral/non-viral genome dataset previously used to train Viraminer and compare the performance the 2 models [22].

3.2.1 Comparison with Viraminer

When trained end to end, the Nucleic Transformer significantly outperforms the Viraminer counterpart by 3.4% AUC score. When trained with a two-stage training process combining two branches with different hyperparamters and pooling schemes, the Viraminer performed significantly better compared to its end to end counterpart, but the Nucleic Transformer still leads in accuracy even with just one end to end model ($k = 13$). It is widely known that even just simply averaging results from a few machine learning models trained with slightly different hyperparameters usually leads to better performance, so it is no surprise that Viraminer's strategy to combine two branches with different setups prove successful. For better comparison, we also trained the Nucleic Transformer 2 and 3 times with different ks and averaged the test set predictions. The AUC improved upon by 0.3% upon averaging 2 models ($k = 11, 13$), but with 3 models averaged ($k = 11, 13, 15$), the AUC only improved slightly by 0.2%. Here we used $n_{mute} = 40$, six transformer encoder layers, $d_{model} = 512$, and $n_{head} = 8$.

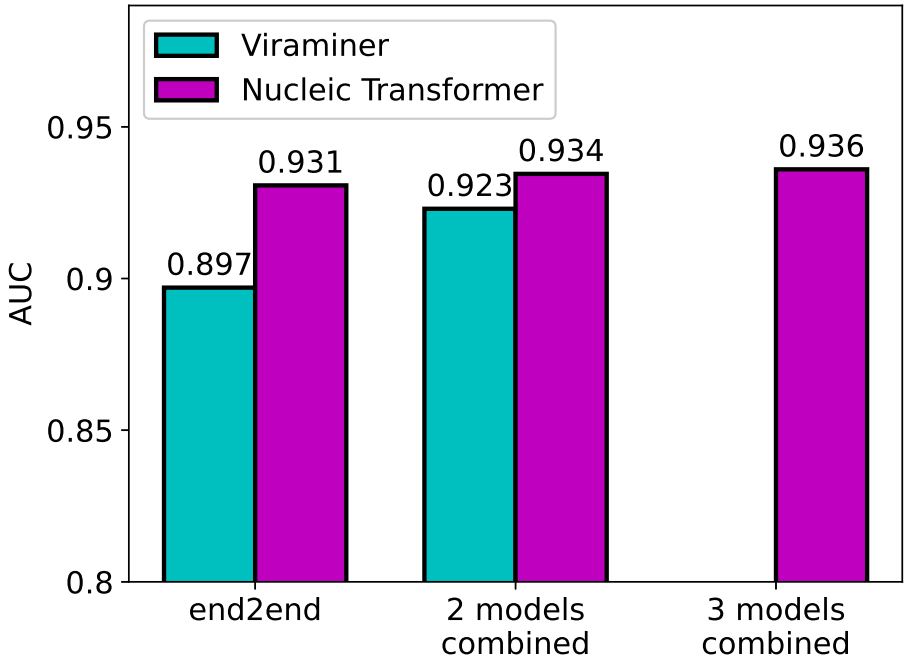

Figure 10: Comparison of test set performance between Nucleic Transformer and Viraminer.

3.2.2 Model width and depth

Previous works utilizing genomics usually used relatively shallow networks and did not explore the effects of varying the depth or width of neural networks. When hypertuning the Nucleic Transformer, we found that increasing the depth and width of the transformer encoder generally led to better performance on the viral/non-viral dataset (Figure 11, 12), although we do see that increasing the model width to 1024 actually led to worse results, mostly due to overfitting. Notably, we found that increasing model depth beyond 6 transformer encoder layers actually led to diverged training on the viral genome dataset, probably because of the huge class imbalance. Although increasing the depth and width both lead to better performance, an argument can be made that it is more efficient to increase the depth of the transformer encoder, because computational complexity and the number of parameters increase quadratically with the width of the transformer encoder.

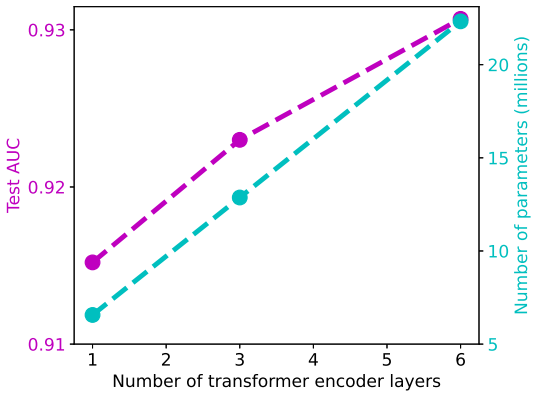

Figure 11: The effect of the number of transformer encoder layers on test AUC performance.

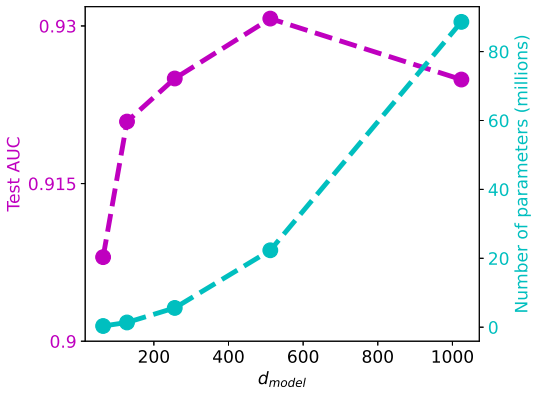

Figure 12: The effect of the width of transformer encoder layers on test AUC performance.

3.3 OpenVaccine challenge

To further test our ideas, we used an adaption of the Nucleic Transformer in the recent 21-day OpenVaccine challenge, hosted by Das Lab at Stanford University [33]. We first show our results using unsupervised learning that led to a 7th finish in 1636 teams of machine learning experts from all over the world and how interpretibility helped our decision making in model selection. Then we demonstrate using a semi-supervised learning approach the Nucleic Transformer can give better results than even the top finish, which used a variety of feature engineering and more than 50 models. We also compare predictions made using secondary structure information from different biophysical models. Finally, we show that pretraining on randomly generated RNA sequences result in very similar performance as pretraining on test set sequences.

3.3.1 Interpretiblility of self-attention helps decision making

The biggest challenge in OpenVaccine was that the private test set had sequences that were longer and more diverse. In machine learning competitions, competitors would normally have to rely on local cross validation to evaluate model performance. Nevertheless, the training set had noisy training samples and limited sequence diversity, which meant that even local cross validation may not be robust enough. During the course of the competition, each team could make up to 5 submissions per day, and score on the public test set would be displayed for each submission. The final ranking was to be evaluated with a private test set that consisted of more diverse and longer mRNA vaccine sequences and each team could select up to 2 submissions for final evaluation. The score on the private test set would not be shown until competition deadline, ensuring a robust testing process.

By the end of the competition, we had trained two sets of models to use in the final submissions, one that was trained directly on short sequences with labels and one that was pretrained with all available sequences before training on on short sequences with labels. Our initial experiments with pretraining did not result in improvements in cross validation or on the public test set, so it was difficult to make a decision purely based on model performance.

In order to robustly select submissions, we visualized and evaluated the learned attention weights from the transformer encoder (Figure 13). Since we add the BPP matrix and distance matrix as a bias, both learned attention distributions of pretrained and non-pretrained models resembled the BPP and distance matrix, but there were also some key differences. The non-pretrained model paid heavy attention indiscriminately to pairs of positionally close nucleotides, as indicated by the bright stripes parallel and close to the diagonal of the attention matrix. This indicates the non-pretrained model thought that positionally close nucleotides were always important when making predictions on mRNA degradation properties, which seemed highly unlikely. On the other hand, the pretrained model did not show the same bias towards pairs of positionally close nucleotides, and was able to recognize the weak BPP connections which were barely visible on the original BPP matrix. In this case, the model seemed to make more effective use of the BPP matrix generated by biophysical models. Because of these considerations, we favored pretrained models in our final submissions, where one submission was an average of 20 pretrained models, and the other was an average of 20 pretrained models and 20 non-pretrained models.

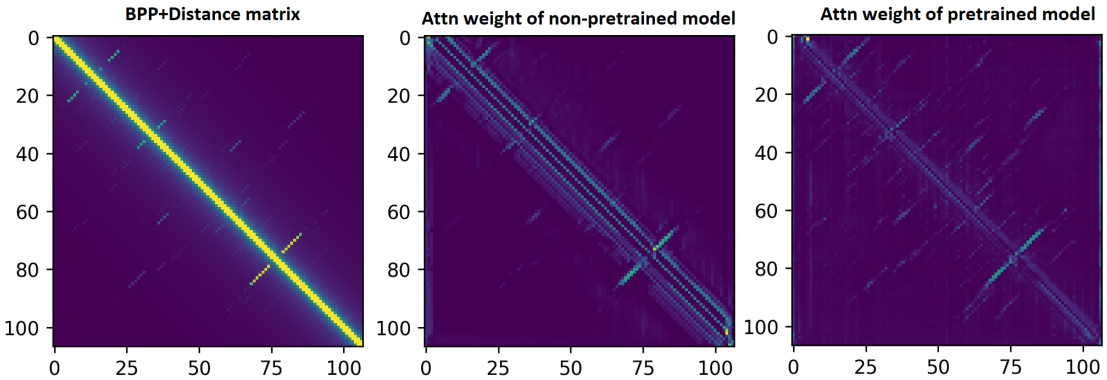

Figure 13: Visualization of BPP+distance matrix, attention weights of a non-pretrained Nucleic Transformer, and attention weights of a pretrained Nucleic Transformer

Results on the private test set validated our selection based on visual inspection of attention weights (Figure 14). Pretrained models performed significantly better than non-pretrained models on both public test set and private test set. The non-pretrained models would have placed us at 39th/1636 instead of 7th/1636. Notably, the predictions on the private test set have much higher error, likely both due to longer sequence length and more sequence diversity in the private test set. For models trained during the competition, n_{head} is set to 32, d_{model} is set to 256, dropout is set to 0.1, and conv2d filter size is set to 32, α and β are both 1. Half of the models were trained with sequences with signal to noise greater than 1 and half were trained with signal to noise greater than 0.5.

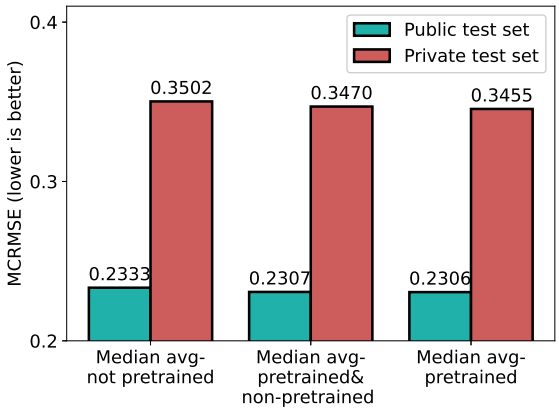

Figure 14: Comparison of pretrained and non-pretrained models trained during the OpenVaccine competition.

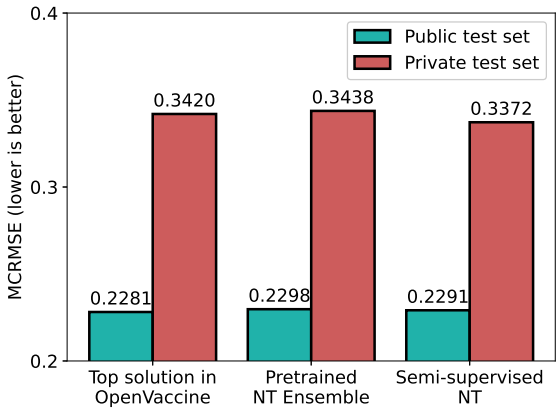

Figure 15: Performance comparison of semi-supervised learning on the OpenVaccine dataset.

3.3.2 Semi-supervised learning leads to more accurate predictions

During post competition experiments, we found that results are better when penalizing measurements with high error more by reducing α to 0.5 and increasing β to 5. Although using more models more different conditions can give better results, for better reproducibility we only trained 5 models with $k = n_{layer} = 3, 4, 5, 6, 7, 8$ for each experiment hereinafter. Here n_{head} is set to 32, d_{model} is 256, dropout is set to 0.1, and conv2d filter size is set to 8. We also only use sequences with signal to noise greater than 0.25 for training and signal to noise greater than 1 for 10-fold cross validation.

Although all top 3 solutions in OpenVaccine employed semi-supervised learning approaches, we did not do so during the competition. Following the competition, we found that using predictions generate by an ensemble of 5 Nucleic Transformers as pseudo-labels, we could reduce the test set MCRMSE to 0.33722, compared to the top solution's 0.34198 (Figure 15). This is somewhat surprising given the that the predictions used as pseudo-labels could only score 0.3438 on the private test set. When we compare the R2 scores of supervised only, unsupervised, and semi-supervised

learning, we see that unsupervised and semi-supervised resulted in significant improvements over the supervised only (Figure 16). Also, we notice that predictions for deg_Mg_pH10 have significantly larger errors for the other 2 properties. This is expected because the biophysical models used are incapable of generating different secondary structure predictions at different pH's.

It is important to note that the semi-supervised learning approach requires pseudo-labeling and knowing the test set distribution before hand. Therefore, while it is effective under the competition setting, it is a risky approach and the performance gain may not transfer in real life applications.

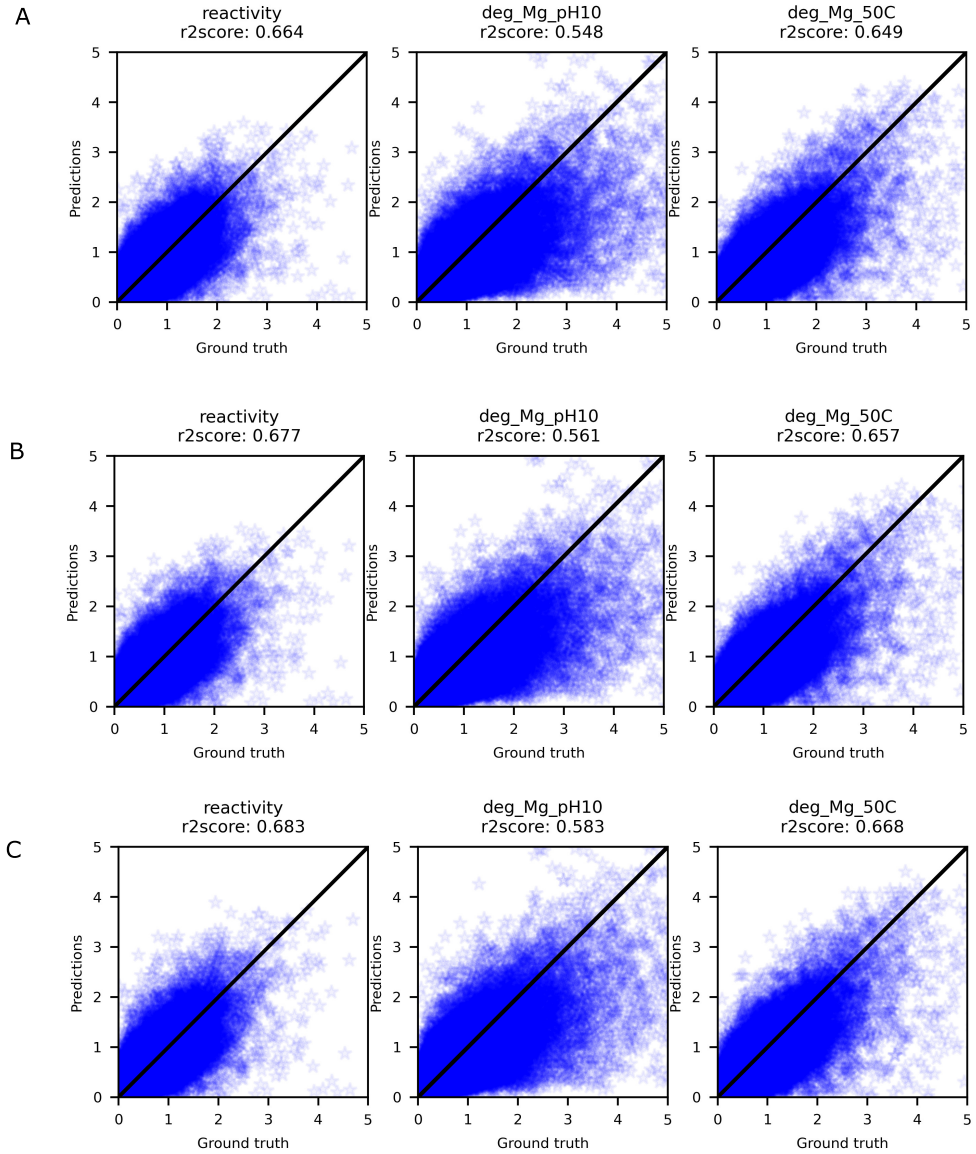

Figure 16: R2 score on the OpenVaccine private test set of the Nucleic Transformer. A: supervised only, B: unsupervised, C: semi-supervised

3.3.3 Comparison of different biophysical models

Different biophysical models often produce somewhat different secondary structure predictions, so here we compare the MCRMSE of the predictions made using inputs from different biophysical models. Previous study has ranked these biophysical models using high-throughput experiments and found CONTRAfold and RNAsoft to be the most accurate

Package	Public MCRMSE	Private MCRMSE	Public MCRMSE semi-supervised	Private MCRMSE semi-supervised
RNAsoft	**0.23154**	0.34521	0.23101	**0.33841**
rnastructure	0.23637	0.34639	0.23502	0.33989
EternaFold	0.23158	0.34613	**0.23081**	0.33878
Contrafold_2	0.23245	0.34858	0.23123	0.34111
NUPACK	0.23679	0.34955	0.23587	0.34292
Vienna	0.23492	**0.34515**	0.2337	0.33864
avg of all	0.22976	**0.34375**	0.22914	**0.33722**

Table 3: Performance using inputs generated by different biophysical models to predict mRNA degradation in the OpenVaccine dataset.

[43]. Here we find that overall RNAsoft performs the best across different conditions. On the private test set, we see that predictions made using secondary structure information from Vienna and RNAsoft have the lowest MCRMSE without semi-supervised learning. With semi-supervised learning, RNAsoft gives most accurate predictions, while EternaFold and Vienna come close. Additionally, simply the averaging all predictions made using all available packages on the private test set always results in lower MCRMSE than using any single package, indicating that ensembling different biophysical models is a good strategy that would give better predictions than any single model.

3.3.4 Learning from random sequences

To explore whether the pretraining performance gain results from knowing the test set distribution, we also generated additional completely random sequences to use for pretraining. The sequences in the test set have more sequence diversity than the training set, and generated sequences are even more diverse (Figure 17). We repeated the pretraining process with random sequences using the same hyperparamters, our experiments show that pretraining with random sequences instead of test sequences results in almost identical performance and increasing the amount of random sequences for pretraining leads to slightly better performance on both test sets (Figure 18). These results suggest that pretraining improves test set error not because of information leak from knowing the test sequences but rather the model learning the generalized rules of mRNA secondary structure formation.

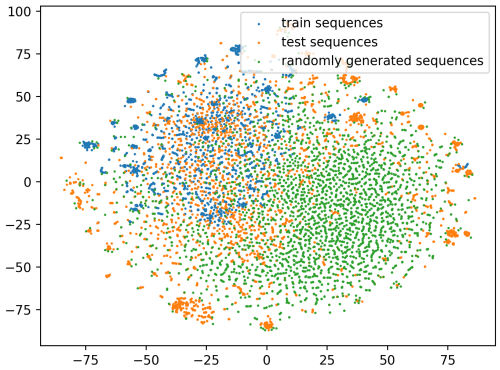

Figure 17: Tsne plot of RNA sequences in the training set, test set, and randomly generated set

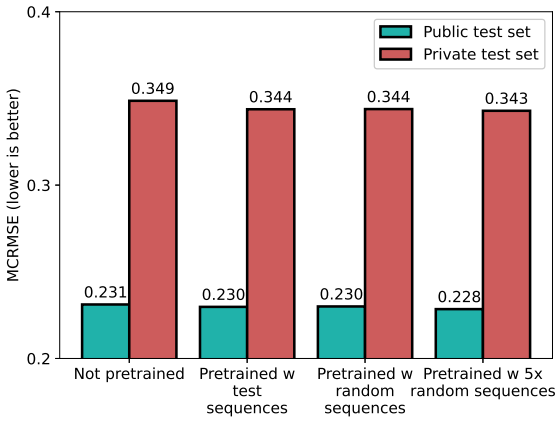

Figure 18: Test set performance of OpenVaccine challenge using different pretraining procedures.

Additionally, we also pseudo-labeled random sequences and retrained our models in semi-supervised fashion. In this case, pseudo-labeling with completely random sequences did not lead to significant improvements (Figure 19). It is likely that due to the large mutation distance of random sequences compared to train and test sequences, the pseudo-labels on the random sequences are not close to ground truth values at all. Therefore, the semi-supervised learning process ended up feeding mostly noise to the model during training, although it did lead to very slight improvement on the private test set over the model without semi-supervised learning.

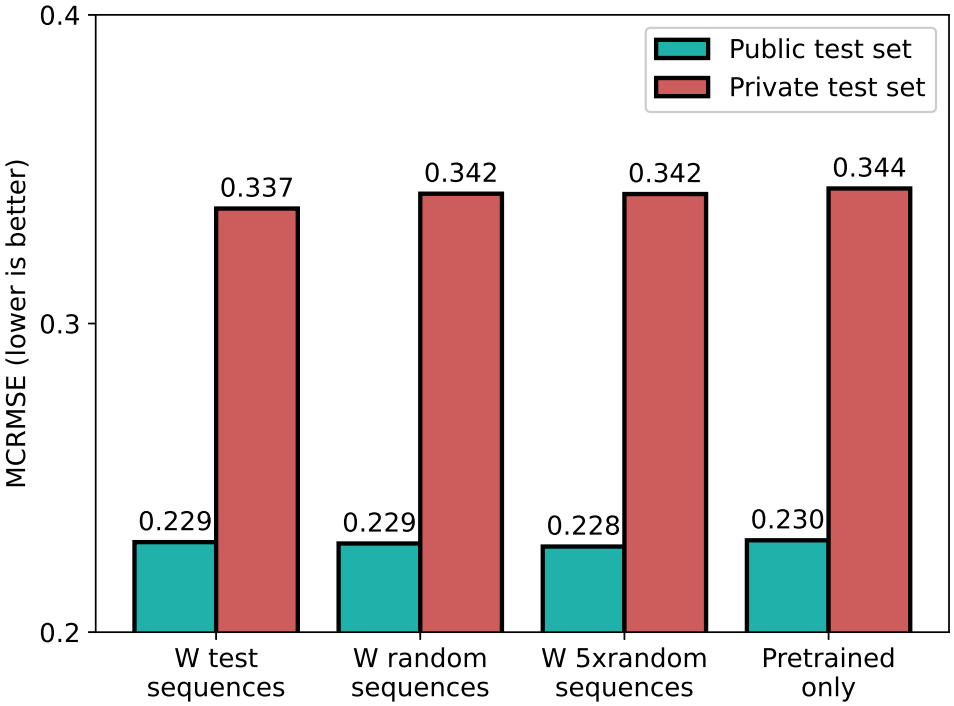

Figure 19: Semi-supervised learning with test sequences compared with random sequences.

4 Conclusion

In this work, we propose the Nucleic Transformer, an effective and yet conceptually simple architecture that is capable of high performance in classifying promoters and viral genome as well as predicting degradation properties of mRNA COVID-19 vaccine candidates per nucleotide. The Nucleic Transformer architecture outperforms other deep learning/non deep learning methods that require hand-crafted features in *E. coli* promoter classification, while also providing interpretibility and is capable of extracting promoter motifs directly from learned attention. Although always trained end to end, the Nucleic Transformer leads in terms of accuracy in classifying viral genomes compared to previous models such as Viraminer, which requires sophisticated multi-stage training and ensembling. As an additional test, we also participated in the recent OpenVaccine challenge with an adaptation of the Nucleic Transformer and placed 7th out of 1636 teams of top machine learning experts from all over the globe. We also demonstrate with semi-supervised learning, the Nucleic Transformer outperforms even the top solution in the OpenVaccine challenge by a considerable margin.

Although classification of promoters and viral genomes have been well studied in the literature, there is no precedence on predictions of mRNA degradation properties per nucleotide. After a chaotic 2020 caused by COVID-19, mRNA vaccines have emerged as a suitable solution to the COVID problem, with companies like Pfizer and Moderna rolling out mRNA vaccines at unprecedented speeds. However, storage and transport remain a challenge with fragile mRNA vaccines (Pfizer's vaccine has to be stored as -70 Celsius). One strategy to reduce mRNA hydrolysis is to redesign RNAs to code for the same proteins but form double-stranded regions, which are protected from these degradative processes. Our work can provide guidance and trained models can act as a screening tool in development of more stable mRNA vaccines in the future. It is our hope the Nucleic Transformer can aid design of more stable mRNA vaccines that can withstand harsher conditions than current ones. It is important to note, however, that there is still significant gap between errors on the 107-bp mRNA sequences and the 130-bp mRNA sequences, both due to difference in sequence length and diversity. Actual COVID-19 candidates are even longer and modeling those remains a challenge in the future.

Our results demonstrate that self-attention and convolution are a powerful combination for genomics tasks, enabling learning both global and local dependencies effectively. It has long been posited that the transformer architecture can excel in not only natural language processing but other fields as well, and our work provides evidence for that.

One key challenge, however, is the quadratic computational complexity of self-attention, which prohibits training on long sequences (greater than 512). Notably, much work has been done in very recent times on reducing the quadratic computational complexity of self-attention to linear to enable training of transformer like self-attention on much longer sequences [48, 49, 50, 51], i.e. linear transformers. Whole genomes and COVID-19 mRNA vaccines both greatly exceed the length limit of full self-attention, and COVID-19 vaccine candidates, in particular, are around 4000 bp long, so we plan to apply these linear transformers in the future.

References

[1] Berg JM, Tymoczko JL, and Stryer L. Biochemistry. *5th edition*. New York: W H Freeman; 2002. Chapter 5, DNA, RNA, and the Flow of Genetic Information, 2002.

[2] Malte Spielmann and Stefan Mundlos. Looking beyond the genes: the role of non-coding variants in human disease. *Human Molecular Genetics*, 25(R2):R157–R165, 06 2016.

[3] Frank J. Slack and Arul M. Chinnaiyan. The role of non-coding rnas in oncology. *Cell*, 179(5):1033 – 1055, 2019.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[7] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878, 2016.

[8] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, 44, June 2016.

[9] A. A. K. Nielsen and C. A. Voigt. Deep learning to predict the lab-of-origin of engineered DNA. *Nat Commun 9*, 3135, 2018.

[10] Ameni Trabelsi, Mohamed Chaabane, and Asa Ben-Hur. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. *Bioinformatics*, 35, July 2019.

[11] N. M. Angenent-Mari, A. S. Garruss, L. R. Soenksen, et al. A deep learning approach to programmable RNA switches. *Nat Commun 11*, 5057, 2020.

[12] J. H. Lam, Y. Li, L. Zhu, et al. A deep learning framework to predict binding preference of RNA constituents on protein surface. *Nat Commun 10*, 4941, 2019.

[13] N. Amin, A. McGrath, and YP. Chen. Evaluation of deep learning in non-coding RNA classification. *Nat Mach Intell*, 1, 2019.

[14] Ying He, Zhen Shen, Qinhu Zhang, Siguo Wang, and De-Shuang Huang. A survey on deep learning in DNA/RNA motif mining. *Briefings in Bioinformatics*, 10 2020. bbaa229.

[15] Y. Zhang, S. Qiao, S. Ji, et al. Deepsite: bidirectional LSTM and CNN models for predicting DNA–protein binding. *Int. J. Mach. Learn. & Cyber*, 11, 2020.

[16] Q. Liu, L. Fang, G. Yu, et al. Detection of DNA base modifications by deep recurrent neural network on Oxford Nanopore sequencing data. *Nat Commun 10*, 2449, 2019.

[17] Heng Li. Identifying centromeric satellites with dna-brnn. *Bioinformatics*, 35(21):4408–4410, 04 2019.

[18] C. Angermueller, H. J. Lee, W. Reik, et al. Deepcpg: accurate prediction of single-cell DNA methylation states using deep learning. *Genome Biol 18*, 67, 2017.

[19] Ruhul Amin, Chowdhury Rafeed Rahman, Sajid Ahmed, Md Habibur Rahman Sifat, Md Nazmul Khan Liton, Md Moshiur Rahman, Md Zahid Hossain Khan, and Swakkhar Shatabda. iPromoter-BnCNN: a novel branched CNN-based predictor for identifying and classifying sigma promoters. *Bioinformatics*, 07 2020. btaa609.

[20] Meng Zhang, Fuyi Li, Tatiana T Marquez-Lago, André Leier, Cunshuo Fan, Chee Keong Kwoh, Kuo-Chen Chou, Jiangning Song, and Cangzhi Jia. MULTiPly: a novel multi-layer predictor for discovering general and specific types of promoters. *Bioinformatics*, 35(17):2957–2965, 01 2019.

[21] Bin Liu and Kai Li. ipromoter-2l2.0: Identifying promoters and their types by combining smoothing cutting window algorithm and sequence-based features. *Molecular Therapy - Nucleic Acids*, 18:80 – 87, 2019.

[22] Ardi Tampuu, Zurab Bzhalava, Joakim Dillner, and Raul Vicente. Viraminer: Deep learning on raw dna sequences for identifying viral genomes in human samples. *PLOS One*, Sep 2019.

[23] Bin Liu and Kai Li. ipromoter-2l2.0: Identifying promoters and their types by combining smoothing cutting window algorithm and sequence-based features. *Molecular Therapy - Nucleic Acids*, 18:80–87, 2019.

[24] Bin Liu, Fan Yang, De-Shuang Huang, and Kuo-Chen Chou. iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics*, 34(1):33–40, 09 2017.

[25] Hannah K. Wayment-Steele, Do Soon Kim, Christian A. Choe, John J. Nicol, Roger Wellington-Oguri, R. Andres Parra Sperberg, Po-Ssu Huang, and Rhiju Das. Theoretical basis for stabilizing messenger rna through secondary structure design. 2020.

[26] Bin Liu, Fan Yang, De-Shuang Huang, and Kuo-Chen Chou. iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics*, 34(1):33–40, 09 2017.

[27] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[28] Socorro Gama-Castro, Heladia Salgado, Alberto Santos-Zavaleta, Daniela Ledezma-Tejeida, Luis Muñiz-Rascado, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Irma Martínez-Flores, Lucia Pannier, Jaime Abraham Castro-Mondragón, Alejandra Medina-Rivera, Hilda Solano-Lira, César Bonavides-Martínez, Ernesto Pérez-Rueda, Shirley Alquicira-Hernández, Liliana Porrón-Sotelo, Alejandra López-Fuentes, Anastasia Hernández-Koutoucheva, Víctor Del Moral-Chávez, Fabio Rinaldi, and Julio Collado-Vides. RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. *Nucleic Acids Research*, 44(D1):D133–D143, 11 2015.

[29] Hao Lin, En-Ze Deng, Hui Ding, Wei Chen, and Kuo-Chen Chou. iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition. *Nucleic Acids Research*, 42(21):12961–12972, 10 2014.

[30] H. Lin, Z. Liang, H. Tang, and W. Chen. Identifying sigma70 promoters with novel pseudo nucleotide composition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(4):1316–1321, 2019.

[31] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. *Machine Learning and Knowledge Discovery in Databases Lecture Notes in Computer Science*, page 145–158, 2011.

[32] Marek Nowicki, Davit Bzhalava, and Piotr Bała. Massively parallel implementation of sequence alignment with basic local alignment search tool using parallel computing in java library. *Journal of Computational Biology*, 25(8):871–881, 2018.

[33] Openvaccine: Covid-19 mrna vaccine degradation prediction.

[34] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[35] Martin Popel and Ondrej Bojar. Training tips for the transformer model. *CoRR*, abs/1804.00247, 2018.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[38] Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. Gradient centralization: A new optimization technique for deep neural networks, 2020.

[39] I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *CoRR*, abs/1905.00546, 2019.

[40] M. Andronescu. Rnasoft: a suite of rna secondary structure prediction and design software tools. *Nucleic Acids Research*, 31(13):3416–3422, 2003.

[41] Jessica S Reuter and David H Mathews. Rnastructure: software for rna secondary structure prediction and analysis. *BMC Bioinformatics*, 11(1), 2010.

[42] C. B. Do, D. A. Woods, and S. Batzoglou. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22(14), 2006.

[43] Hannah K. Wayment-Steele, Wipapat Kladwang, and Rhiju Das. Rna secondary structure packages ranked and improved by high-throughput experiments. *bioRxiv*, 2020.

[44] Robert M. Dirks and Niles A. Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *Journal of Computational Chemistry*, 25(10):1295–1304, 2004.

[45] Ronny Lorenz, Stephan H Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1), 2011.

[46] David Rolnick, Andreas Veit, Serge J. Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *CoRR*, abs/1705.10694, 2017.

[47] Avihu H. Yona, Eric J. Alm, and Jeff Gore. Random sequences rapidly evolve into de novo promoters. *Nature Communications*, 9(1), 2018.

[48] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[49] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[50] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

[51] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2020.