DNA aptamers modelling :
Comparison of different machine learning models in the prediction of structural properties

Hani Zaki
20176047
hani.zaki@umontreal.ca

David Arsenic
20200958
david.arsenic@umontreal.ca

Haunui louis
20189376
haunui.louis@umontreal.ca

Dean Johan Bell
20087490
dean.johan.bell@umontreal.ca

Yehao Yan
20179475
yehao.yan@umontreal.ca

Abstract

This paper investigates the predictive capabilities of various machine learning models : Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNN) and Transformers for DNA sequence analysis. Our goal for this project is to accurately predict structural properties of DNA sequences, including the minimum free energy (MFE),the number of hairpins, and the secondary structure. Our findings reveal that LSTMs excel in leveraging sequential and structural patterns of DNA, resulting in superior performance in both MFE prediction and hairpin counting. In comparison, Transformers show similar consistent results but are hindered by challenges related to their training techniques. Conversly, CNNs offer a different approach that shows potential.

1 Introduction

DNA aptamers are nucleotide sequences that have the ability to bind to specific molecules. Their high affinity and specificity for a molecule allow aptamers to have a wide variety of applications. For example, in biosensors to detect the presence of molecules (toxins, pathogens, etc.) in an environment. They are also used in nanotechnologies and more specifically targeted therapy in the treatment of diseases that use the high affinity of aptamers to bind and affect pathogenic molecules.

Aptamers are generally synthesized from random sequences by an iterative selection called SELEX. An interesting avenue to accelerate and optimize this discovery process is Machine Learning and more specifically neural architectures. For example, generative networks can be used to generate DNA sequences with desired properties.Our project focuses on the prediction of properties inherent to a DNA sequence. These properties give us an idea of their stability and affinity with specific molecules.The models explored in this project are Transformers, LSTMs, and CNNs.

2 Review of the literature

The application of convolutional networks was inspired by various scientific papers [1][2] that show good results in predicting secondary structure. It is worth noting that the models presented in these papers are used for RNA sequences, not DNA sequences. However, their methods for converting sequences into 2D representations for our CNNs were used in this project. Their architectures and hyperparameters also provided us with a starting point to explore the application of CNNs for DNA sequences.

The Transformer model is a state-of-the-art approach that has demonstrated superior performance in various domains. The competence of this model is notably illustrated by its application to the prediction of aptamer-protein interactions, as shown in two significant studies: "Aptamer-Protein Interaction Prediction using Transformer" [10] and "AptaTrans: a deep neural network for predicting aptamer-protein interaction using pre-trained encoders" [11].

These two studies exploit the Transformer architecture to improve the prediction of aptamer-protein bindings, a crucial step to accelerate the drug discovery process. The first study validates its model compared to traditional methods and shows promising improvements, while the second extends this approach by integrating pre-trained encoders to refine the predictive accuracy of the model. Inspired by these results, our project explores how transformers can advance our understanding of DNA aptamer modeling.

3 Methods

3.1 Data generation

We generated 2.5 million random sequences, each with a length between 10 and 50 characters. The generation procedure is as follows: we randomly choose a length for each sequence, between 10 and 50 characters. Then, we produce the sequence using the characters A, C, G, T, giving each the same probability of occurrence. Once the sequence is created, we use NUPACK to predict its minimum free energy and secondary structure. If the minimum free energy of the sequence is zero, we repeat the process until a valid sequence is obtained. Finally, we determine the number of hairpins by counting the groupings of parentheses in the secondary structure.

For the test set, we randomly generated 500 DNA sequences of each length, from 10 to 100, while excluding sequences with null minimum free energy. We also checked that none of the sequences were part of the previously generated dataset of 2.5 million sequences. We note that our testing is only done on the sequences from length 10 to 50.

3.2 Loss Functions and Optimizer

For all models, we use mean squared error (MSE) as the loss function for the prediction of MFE and number of hairpins. We use a Cross-Entropy loss for the prediction of the secondary structure. Specifically for the Transformer model, we use a weighted Cross-Entropy loss. We give weights of 1.0 to the unpaired nucleotide token "." and for the EOS token. We give weights of 2.0 for the paired nucleotide tokens "(" and ")". We use Adam optimizer for all models.

3.3 Models

Three models were implemented: CNN, LSTM and Transformer. Each has two variants, the first one to predict only the MFE and number of hairpins, and the second one to additionally predict the secondary structure. We note that the CNN does not predict the number of hairpins.

3.3.1 LSTM

To implement the LSTM model, we use the Keras library, which offers a standard LSTM layer implementation [3], using a forget gate bias fixed at 1, as recommended by Jozefowicz et al.[4]. Initially, each RNA sequence is encoded into a vector and then padded to achieve vectors of uniform size. The secondary structures of the sequences are then transformed using one-hot encoding.

For the model aiming to predict only the minimum free energy and the number of hairpins, each sequence is first processed through an Embedding layer with an output dimension of 128. It then passes through five bidirectional LSTM layers, each with a latent dimension of 256. Finally, two linear layers are used for prediction. The model is trained over 73 epochs. Early stopping with a patience of 3 is employed. During training, the model minimizes the total mean squared error on the free energy and the number of hairpins, with a weight of 2 assigned to the number of hairpins. In total, the model comprises 1,841,282 parameters.

For the model that predicts the minimum free energy, the number of hairpins and the secondary structure, the same architecture is used, with the addition of a TimeDistributed layer after the last LSTM layer, allowing the prediction of the secondary structure. The TimeDistributed layer outputs a 50x4 matrix representing 50 probability vectors of size 4 (the probability that element i is an opening parenthesis, closing parenthesis, a dot, or padding). This model is trained over 40 epochs with early stopping and a patience of 3. During training, the model minimizes the total mean squared error on the free energy and the number of hairpins, as well as the categorical cross-entropy for the secondary structure, with a weight of 2 given to the number of hairpins and secondary structure. In total, the model comprises 1,842,310 parameters.

Several studies also suggest the application of a sliding window with variable overlap (2 to 4) to input sequences to predict secondary structure in the context of proteins [5][6]. In the NLP domain, this method artificially generates a corpus with more words than if one were limited to the 20 amino acids. As RNA sequences are also sets of amino acids, it is natural to wonder if this approach can be adapted to our problem. Unfortunately, the use of 2 to 4 size n-grams did not improve performance and even led to decreased performance for n-grams larger than size 4.

3.3.2 CNN

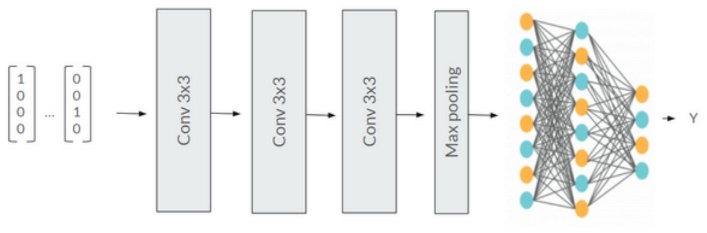


Figure 1: CNN Architecture

Our initial architectures were classic implementations of 1D-CNNs. Our first objective was to predict the free energy from the one-hot encoding of our sequences. This initial architecture had 3 convolutional layers with a kernel size of 3x3. The number of filters per convolutional layer goes from 32 to 64 to 128 before reaching the pooling layer, which reduces the dimensionality to move to the fully connected layers. For the fully connected layers, the result obtained from the convolutional layers needs to be flattened. We have 3 fully connected layers using ReLU which allow the prediction of our desired property. This model has a total of approximately 9 million parameters.

For the convolutional layers, we apply Batch Normalization to stabilize learning, and we apply dropout to the fully connected layers to avoid overfitting.

This first implementation with the one-hot encoding did not allow the network to converge and learn from the data. We therefore transformed our sequences into a 2D representation as explained in reference [1]. For our case, we use the Watson-Crick pairing rules.

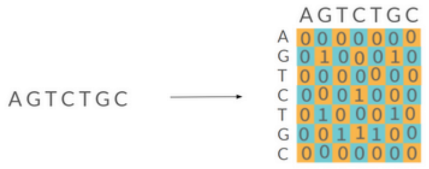


Figure 2: CNN Architecture

We thus look for each nucleotide in the sequence if it can pair with another nucleotide in the same sequence. We have 4 possible pairings: A-T, T-A, G-C, C-G, which gives us 4 channels in our 2D representation. We add to these 4 channels another channel representing unpaired nucleotides.

We thus transform a sequence of size L into an image of size LxL with 5 channels. Note that we also add padding to the images to have a uniform image size, allowing us to have sequences of variable length.

Another challenge to overcome concerns the prediction of the secondary structure. Secondary structures are represented as variable-length character strings. In order for our CNN to predict character strings, we decided to transform the secondary structures into images (similarly to the DNA sequence). We also apply padding to these images to address the variability in sequence length.

So our CNN takes an image (DNA sequence) as input and outputs another image (secondary structure). We therefore need to adjust the architecture of our model to allow the model to output an image instead of a float. To do this, we simply change the number of parameters in our last fully connected layer.

3.3.3 Transformer

We utilized two different transformer model: one for predicting both the minimum free energy (MFE) and number of hairpins, and one that additionally predicts the secondary structure. Both models are described below.

Architecture

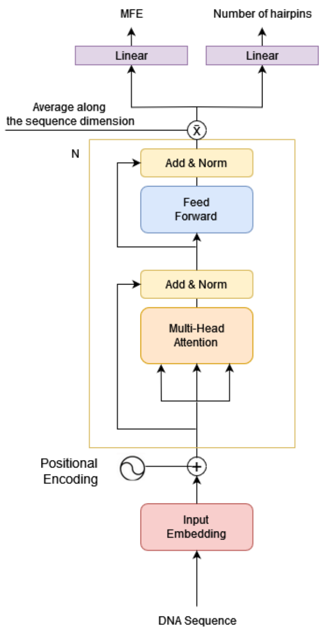


Figure 3: Transformer: Encoder only

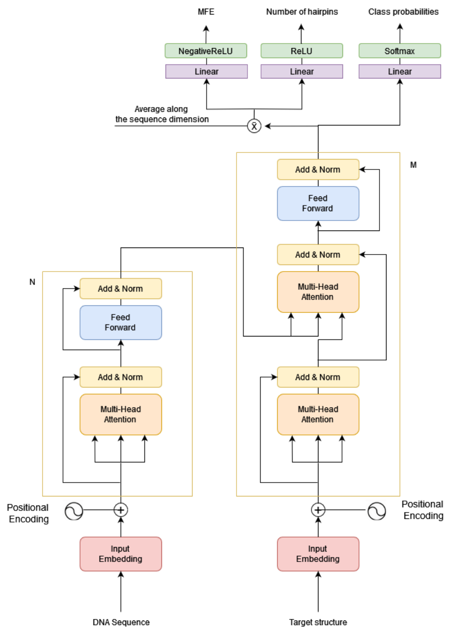


Figure 4: Transformer: Encoder & Decoder

The first Transformer (Encoder only) predicts both the MFE and the number of hairpins. After processing the input through the decoder, we compute the average along the sequence dimension at the encoder's output. This averaged tensor is then fed into two separate linear layers, one for each predicted value.
The second Transformer (Encoder & Decoder) averages the output at the decoder's end. The averaged tensor is then fed into two linear layers to predict both the MFE and the number of hairpins. This time, we use activation functions: NegativeReLU for the MFE and ReLU for the number of hairpins. For predicting the secondary structure class probabilities, we give the raw decoder output to a linear layer activated by a Softmax function.

Training

For the first Transformer model, training is done in a conventional manner. For the second transformer, we use forced teaching. This approach involves using the target structure as input to

the decoder. Additionnally, we implement a technique known as scheduled sampling[7] to help the model get used to its own predictions. This technique, consist of making two passes through the decoder. In the first pass, the target structure is used as input to obtain a preliminary prediction of the secondary structure. We then compute an embedding tensor by multiplying the decoder embedding weights with the softmax score of the predicted structure. During the second pass, we mix the embeddings of the target structure with the embedding tensor. Each epoch, the proportion of the target structure within the mixed embedding from the second pass is progressively reduced at a specified decay rate. This strategy progressively familiarizes the model with its own predictions, and therefore enhances performance during inference.
For both models, we minizime the sum of the normalized loss. To achieve this normalization, we first establish a baseline loss for each predictions. This baseline is determined by summing the individual losses across 3 epochs. We then compute the normalized losses by dividing each loss by its respective baseline. This approach ensures each tasks receives equal attention.

Configuration
Here, we present the configurations of each Transformer model. These configurations were not optimized. We opted for configurations that ensured convergence, since our primary objective is to compare the performance of different machine learning models.

Table 1: Transformer Models Configuration

Transformer 1: Encoder only

Hyperparameter	Value
Encoder Layers	6
—————	–
Model Dimension	256
Feed Forward Dimension	512
Number of Heads	8
Batch Size	128
Learning Rate	0.0002
Dropout Rate	0.1
—————	–

Transformer 2: Encoder & Decoder

Hyperparameter	Value
Encoder Layers	3
Decoder Layers	3
Model Dimension	256
Feed Forward Dimension	512
Number of Heads	8
Batch Size	128
Learning Rate	0.0002
Dropout Rate	0.1
Decay rate	0.05

4 Results

The results are divided into two parts. First, we compare the models created for predicting the MFE and the number of hairpins. Second, we compare the models' predictions when the secondary structure is included.

4.1 MFE and hairpins

In this part of the report, We present to you the results(see table 2) of the MFE and hairpins predictions. The mean squared error (MSE) was used as the evaluation metric.

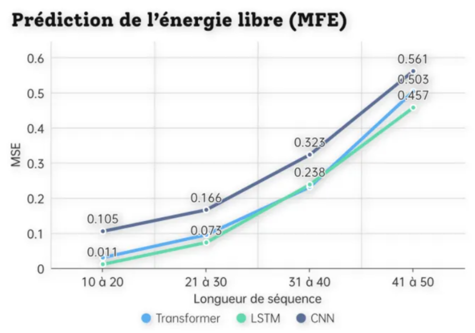
In a first time we present a table where the results are stored.

Table 2: MSE of MFE and number of hairpins prediction without Secondary Structure

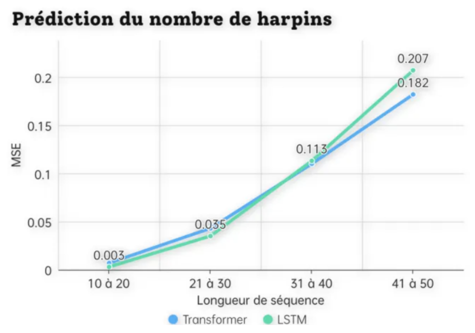
	Transformer		LSTM		CNN	
Sequence lengths	MFE	#hairpins	MFE	#hairpins	MFE	#hairpins
10 à 20	**.03**	0.007	0.011	**0.003**	0.105	**-**
21 à 30	0.095	0.043	**0.073**	**0.035**	0.166	**-**
31 à 40	**0.231**	**0.11**	0.238	0.113	0.323	**-**
41 à 50	0.503	**0.182**	**0.457**	0.207	0.561	**-**
10 à 50 (toutes)	0.211	**0.084**	**0.190**	0.087	0.284	**-**

Note: The results in bold represent the best scores.

Below we present the results in graphical form.



(a) Prediction Mfe



(b) Prediction hairpins

Figure 5: Results

The results show that the performance of the Transformer and LSTM models is quite similar, with the LSTM performing slightly better overall (see figure 5). However, both models outperform the CNN, which exhibits higher errors across all sequence length ranges. This suggests that the Transformer and LSTM architectures are more effective for this task compared to the CNN.

The performance of the models varies with the length of the input sequences. For sequences ranging from 10 to 20 nucleotides, all models perform relatively well, with MSE values ranging from 0.003 to 0.105 for MFE prediction and from 0.007 to 0.087 for hairpin prediction. However, as the sequence length increases, the performance of all models deteriorates, with higher MSE values observed for longer sequences.

Overall, the results indicate that both the Transformer and LSTM architectures are more robust and efficient for predicting MFE and the number of hairpins in DNA sequences compared to the CNN. This is consistent with expectations, as both the Transformer and LSTM architectures are designed to capture long-range dependencies in text sequences effectively.

4.2 MFE, hairpins and secondary structure

We compare secondary structure predictions across all sequence lengths.

Table 3: Secondary structure predictions across all sequence lengths

	Model	Precision	Recall	F1 Score
Prediction of paired nucleotide "("	CNN	**0.884**	0.769	0.822
	LSTM	0.844	**0.866**	**0.865**
	Transformer	0.803	0.820	0.811
	Model	Precision	Recall	F1 Score
Prediction of paired nucleotide ")"	CNN	**0.875**	0.761	0.814
	LSTM	0.843	**0.886**	**0.864**
	Transformer	0.802	0.804	0.803
	Model	Precision	Recall	F1 Score
Prediction of unpaired nucleotide "."	CNN	**0.959**	**0.982**	**0.970**
	LSTM	0.954	0.931	0.943
	Transformer	0.923	0.918	0.921

Bold number represents best score.

The CNN model demonstrates superior precision for paired and unpaired nucleotides, suggesting it has a high specificity in identifying true positives. However, its recall is lower for paired nucleotides, indicating that it misses actual pairs in the sequence more often than the other models.

The LSTM model stands out as the most consistent overall, performing better than the Transformer on all predictions. This indicates that the model not only makes accurate predictions, but also maintains a lower rate of false negatives.

The Transformer model, while not outperforming the others in any single metric, demonstrates a similar consistency as the LSTM.

We now look at the performance of each model in predicting the MFE and the number of hairpins when the secondary structure is also predicted.

Table 4: MSE of MFE and number of hairpins prediction when secondary structure is also predicted

	Transformer		LSTM		CNN			
	MSE						Average values	
Sequence lengths	MFE	#hairpins	MFE	#hairpins	MFE	#hairpins	MFE	#hairpins
10 to 20	0.132	0.009	**0.006**	**0.002**	0.105	-	1.27	1.02
21 to 30	0.205	0.084	**0.045**	**0.026**	0.166	-	2.05	1.15
31 to 40	0.563	0.212	**0.160**	**0.086**	0.323	-	2.98	1.41
41 to 50	1.220	0.373	**0.317**	**0.176**	0.561	-	4.06	1.76
10 to 50 (all)	0.521	0.166	**0.129**	**0.071**	0.284	-	5.40	2.16

Bold number represents best score

We notice that the LSTM model performs better than the other models for all sequence lengths. It even performs better than its other variant. The secondary structure provides valuable information about the pairing of nucleotides and their position. Therefore, a model that correctly predicts the secondary structure can use that information to enhance its performance for the prediction of the MFE and number of hairpins. This can explain the better performance of the LSTM model.

The Transformer model shows worst performance than its variant. The problem could be due to the use of forced teaching and greedy decode algorithm during inference. Forced teaching makes the model dependent on the real target structure, and greedy decode is known for propagating errors.

5 Conclusion

In conclusion, our project implemented and assessed three distinct models - Transformer, LSTM, and CNN - to predict various features of DNA sequences, such as the minimum free energy (MFE) and the number of hairpins across different sequence lengths. The Transformer and LSTM demonstrated robust performance, effectively capturing long-distance dependencies within the DNA sequences. However, the LSTM slightly outperformed the Transformer.

The Transformer could possibly have seen better performance if we had used a relative positional encoder[8]. Absolute positional encoding is known to not perform well in regards to handling sequences of variable lengths[9]. This limitation often affects the model's ability to generalize well across different contexts.

The CNN provided valuable insights by translating sequence data into images, thereby illustrating that all models possess unique strengths and contribute to our understanding of aptamer interactions.

Future initiatives will focus on refining these models to enhance their accuracy and applicability to more complex feature predictions, such as the structural information of aptamers. This could involve the integration of multi-modal data or the development of more sophisticated hybrid models that leverage the distinct advantages of each existing model to provide deeper insights into aptamer behavior and interaction dynamics.

From a broader perspective, the most valuable aspect of aptamer research is the prediction of aptamer-target binding. Moving forward, machine learning models need to be refined to better integrate aptamer structural data for more accurate predictions. Moreover, exploring deep learning algorithms may offer new insights into complex biological interactions, which is crucial for the discovery of innovative drugs and therapeutic applications. The integration of these computational methods promises to significantly streamline the aptamer selection process, making it less labor-intensive and more cost-effective.

6 Team contribution

David Arsenic implemented the LSTM model and wrote the LSTM section and the data generation section.

Haunui Louis implemented the CNN model and wrote the CNN section, part of the litterature review section and the introduction.

Hani Zaki implemented the Transformer model, generated the datasets, wrote the Transformer section and the secondary structure results section and formatted the report in LaTeX.

Yehao Yan wrote the conclusion and part of the litterature review section.

Dean Johan Bell implemented GRU model(see github) and section 4.1.

7 Github Codebase

The link to the github: https://github.com/hanzak/DNA-aptamers-modelling

8 References

[1] Zhang, H., Zhang, C., Li, Z., Liu, C., Xu, W., Zhang, B., & Liu, Y. (2019). A new method of RNA secondary structure prediction based on convolutional neural network and dynamic programming. Frontiers in Genetics, 10. https://doi.org/10.3389/fgene.2019.00467

[2] Booy, M. S., Ilin, A., & Orponen, P. (2022b). RNA secondary structure prediction with convolutional neural networks. BMC Bioinformatics, 23(1). https://doi.org/10.1186/s12859-021-04540-7

[3] Graves, A. (2013, August 4). Generating sequences with recurrent neural networks. arXiv.org. https://arxiv.org/abs/1308.0850

[4] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016, February 7). Exploring the limits of language modeling. arXiv.org. https://arxiv.org/abs/1602.02410

[5] Vries, J. K., Liu, X., & Bahar, I. (2007). The relationship between n-gram patterns and protein secondary structure. Proteins: Structure, Function, and Bioinformatics. https://pubmed.ncbi.nlm.nih.gov/17523186/

[6] Sharma, A. K., & Srivastava, R. (2017). Variable length character n-gram embedding of protein sequences for secondary structure prediction. Journal of Bioinformatics and Computational Biology. https://pubmed.ncbi.nlm.nih.gov/33143605/

[7] Mihaylova, T., & Martins, A. F. T. (2019, June 18). Scheduled sampling for transformers. arXiv.org. https://arxiv.org/abs/1906.07651

[8] Li, S., You, C., Guruganesh, G., Ainslie, J., Ontanon, S., Zaheer, M., Sanghai, S., Yang, Y., Kumar, S., & Bhojanapalli, S. (2023, October 6). Functional interpolation for relative positions improves long context transformers. arXiv.org. https://arxiv.org/abs/2310.04418

[9] Sinha, K., Kazemnejad, A., Reddy, S., Pineau, J., Hupkes, D., & Williams, A. (2022, October 23). The curious case of absolute position embeddings. arXiv.org. https://arxiv.org/abs/2210.12574

[10] Aptamer- Protein Interaction Prediction using Transformer. (2022b, January 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9736493

[11] Shin, I., Kang, K. H., Kim, J., Sanghun, S., Choi, J., Lee, J., Kang, H. Y., & Song, G. (2023). AptaTrans: a deep neural network for predicting aptamer-protein interaction using pretrained encoders. BMC Bioinformatics, 24(1). https://doi.org/10.1186/s12859-023-05577-6