JOIV : Int. J. Inform. Visualization, 7(2) - June 2023 588-599

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Transformer in mRNA Degradation Prediction

Tan Wen Yit [a], Rohayanti Hassan [a], Noor Hidayah Zakaria [a,*], Shahreen Kasim [b], Sim Hiew Moi [a], Alif Ridzuan Khairuddin [a], Hidra Amnur [c]

[a] *Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia*
[b] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia*
[b] *Department of Information Technology, Politeknik Negeri Padang, Sumatera Barat, Indonesia*

Corresponding author: [*]alifridzuan@utm.my

***Abstract*—The unstable properties and the advantages of the mRNA vaccine have encouraged many experts worldwide in tackling the degradation problem. Machine learning models have been highly implemented in bioinformatics and the healthcare fieldstone insights from biological data. Thus, machine learning plays an important role in predicting the degradation rate of mRNA vaccine candidates. Stanford University has held an OpenVaccine Challenge competition on Kaggle to gather top solutions in solving the mentioned problems, and a multi-column root means square error (MCRMSE) has been used as a main performance metric. The Nucleic Transformer has been proposed by different researchers as a deep learning solution that is able to utilize a self-attention mechanism and Convolutional Neural Network (CNN). Hence, this paper would like to enhance the existing Nucleic Transformer performance by utilizing the AdaBelief or RangerAdaBelief optimizer with a proposed decoder that consists of a normalization layer between two linear layers. Based on the experimental result, the performance of the enhanced Nucleic Transformer outperforms the existing solution. In this study, the AdaBelief optimizer performs better than the RangerAdaBelief optimizer, even though it possesses Ranger's advantages. The advantages of the proposed decoder can only be shown when there is limited data. When the data is sufficient, the performance might be similar but still better than the linear decoder if and only if the AdaBelief optimizer is used. As a result, the combination of the AdaBelief optimizer with the proposed decoder performs the best with 2.79% and 1.38% performance boost in public and private MCRMSE, respectively.**

***Keywords*—Transformer; optimizer; AdaBelief.**

Manuscript received 11 Mar. 2022; revised 27 Aug. 2022; accepted 12 Nov. 2022. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Vaccines have been a disease prevention trend by injecting inactivated pathogens [1] or genetic material such as DNA, mRNA, or protein. The downsides of the inactivated pathogens vaccine are the efficiency of development and deployment, and inapplicable to non-infectious diseases such as cancerous diseases where the gene mutation occurs and the cell carrying mutated gene starts to divide and grow out of control, rather than getting infected by bacteria or virus [1]. The nucleic acid therapeutics approach tackles these problems because they are safe and efficient their production is scalable [1]. For example, the mRNA vaccine is safe because it is a non-infectious molecule and undergoes degradation by normal cellular processes. Besides, mRNA can be modified to be more stable and highly translatable [1]. mRNA can be produced cheaper and more scalable through its high yield of in vitro transcription reactions [1]-[4]. Machine learning models have been highly implemented in bioinformatics and the healthcare fieldstone insights from biological data. Deep learning solutions such as Artificial Neural Networks (ANN) have been implemented as promising solutions in various fields of study. ANN as Convolutional Neural Network (CNN) [5]-[7] has been used to analyze images such as CT scans of a patient to analyze, train and predict the possible illnesses that have yet to be diagnosed. Long Short-Term Memory (LSTM) has also been utilized in analyzing and studying the nature of the nucleotide sequences such as binding sites or searching for motifs.

Transformer [8] is one of the state-of-the-art-art deep learning architectures that involve stacks of encoders to analyze and decipher the inputs, mostly a sentence composed by natural languages, and a pile of decoders to transform the encoded inputs into decoded outputs. Several models, such as recurrent neural networks (RNN) [9] and autoencoders [4], have been used in Natural Language Processing (NLP). However, the transformer is more advantageous because it can process multiple inputs in parallel, depending on the

number of encoders and decoders in the predefined architecture. Hence, every encoder has its feature and sensations when reconstructing a sentence in another language. Unfortunately, limited studies show the implementation of transformers in biological systems.

Stanford University has held a competition named OpenVaccine: COVID-19 mRNA Vaccine Degradation Prediction, aiming to gather worldwide solutions. Nucleic Transformer [10]-[12] is one of the promising solutions in the competition. There are also some solutions submitted in the competition, such as XGBoost [13], HistGradientBoostingRegressor [14], Regularized LSTM [15]-[16] and GNN + Attention + CNN ensemble [17]. Only Nucleic Transformer and the ensemble solution provide remarkable prediction loss among these solutions. This indicates that predicting degradation is a complex problem, requiring an ensemble or stacking of models.

There are several challenges to be emphasized when working with machine learning solutions. The possible difficulties are lacking data, overfitting, imbalanced data, and model interpretability [17]. Transfer learning [18]-[21] and data augmentation [22]-[24] can overcome data-hungry problems. Data augmentation is also able to minimize the overfitting of the model. Weight decay, batch normalization [25]-[26], and dropout can be utilized in the model. Penalization of over-confident output from the model is considered one of the solutions for overfitting. To balance the dataset, we can up-sampling smaller or down-sample larger categories for imbalanced data. Deep learning approaches seem like a black-box operation. It is interpretable, meaning we can know what happens when the model is training [27]-[29]. For instance, the backpropagation-based approach [30] and the perturbation-based approach [31] can interpret the model. Therefore, this paper is motivated to propose an improved optimizer to boost the performance of mRNA degradation prediction. Next, Section 2 discusses the method to be implemented, Section 3 presents the results and discussion, and Section 4 ends with a conclusion.

II. Material and Method

The enhanced Nucleic Transformer inherits from the existing Nucleic Transformer, with a proposed decoder and the utilization of AdaBelief/RangerAdaBelief optimizer instead of Ranger optimizer. The proposed decoder consists of two linear layers with a sigmoid activation function in between, acting as a normalization layer before the final output. Fig. 1 shows the design of the proposed decoder.

The degradation rate predictions require several steps, from preparing the data for the transformer model to generating prediction results. First, the exploratory data analysis (EDA) is done once. The EDA step is to study the training dataset's characteristics and explore the effect of data filtering. The filtered data is then reshaped into the specific dimension to be fed into the transformer model later. Afterward, the training and validation split is done through a stratified 10-fold cross-validation technique. The training and testing dataset is read and reshaped for the pre-training step, similar to the training step. However, there are different sequence lengths, which are 107 and 130 requiring splitting them into long and short sequences [32].

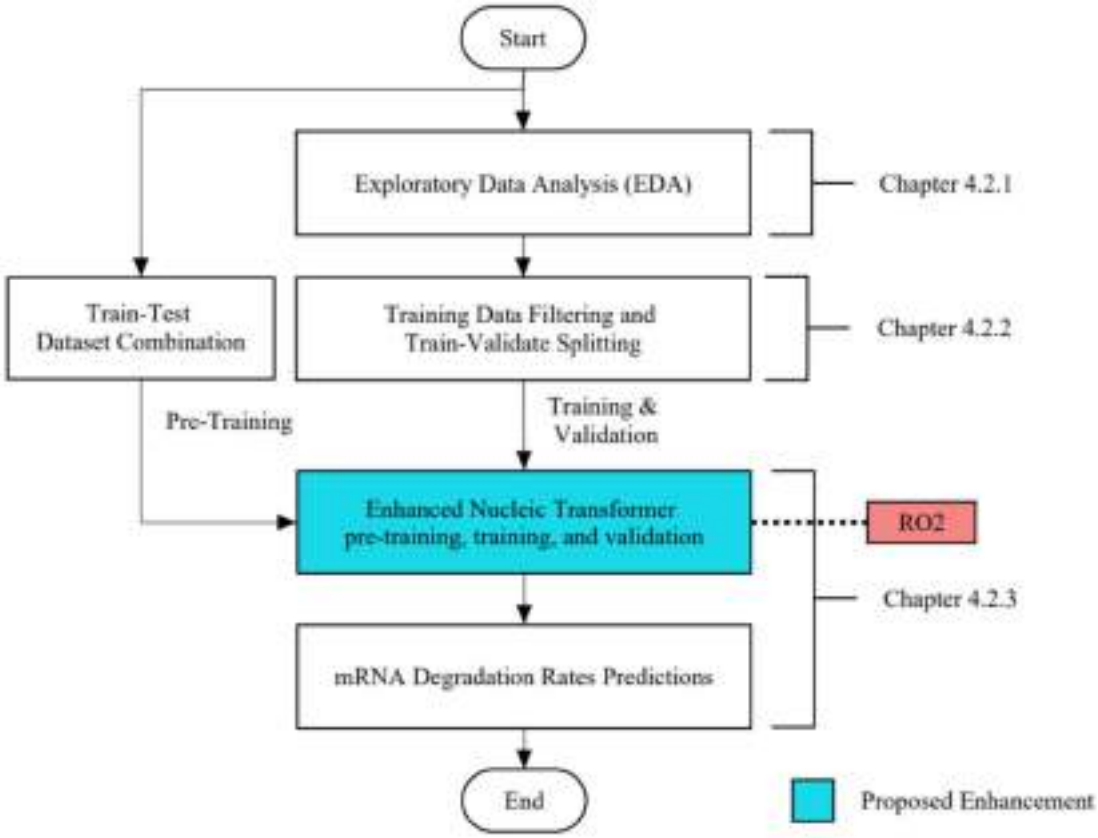

Fig. 1 Flowchart of Experimental Design

In the enhanced nucleic transformer model, five nucleic transformer encoder layers and a proposed decoder layer are used to process the data. In the pre-training step, the long sequences come first to be processed, and then to short lines. The long and short sequences are randomly mutated or masked in random positions. The objective of the pre-training step is to allow the model to predict the true line, structure, and loop type of every masked or mutated sample. The weights and state of the pre-training model are then saved and loaded during the training step. The pre-trained model tells the general rules of mRNA secondary structure. Then, the model's latest state is loaded during the training step. This time the sequences have the same length and can be processed in the transformer model. Validation will be done right after every training epoch. Again, the trained model will be saved and loaded during testing. Fig. 1 shows the steps of operating the nucleic transformer from the beginning of data analysis and preparation until the mRNA degradation rates predictions.

A. Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is a process often used by data scientists to analyze and explore datasets to identify the characteristics of the data. Usually, EDA helps data scientists discover data patterns, noises or outliers, and anomalies before making any assumptions. EDA ensures that the results are valid and applicable to our research objectives. The typical analysis done by data scientists is standard deviations, confidence intervals, data point distributions, etc. In this section, the dataset from OpenVaccine on Kaggle is explored and analyzed to study the features of mRNA vaccine candidates and the trends of their degradation properties. There are five target labels, the five degradation properties provided in the dataset. The relationships between the degradation properties and the predicted loop type of mRNA secondary structure are explored.

	index	signal_to_noise	SN_filter	seq_length	seq_scored
count	2400.000000	2400.000000	2400.000000	2400.0	2400.0
mean	1199.500000	4.530456	0.862083	107.0	68.0
std	692.964648	2.835142	0.473099	0.0	0.0
min	0.000000	-0.503000	0.000000	107.0	68.0
25%	599.750000	2.391000	0.000000	107.0	68.0
50%	1199.500000	4.442500	1.000000	107.0	68.0
75%	1799.250000	6.294250	1.000000	107.0	68.0
max	2399.000000	17.194000	1.000000	107.0	68.0

	index	seq_length	seq_scored
count	3634.000000	3634.000000	3634.000000
mean	1816.500000	126.018987	87.018987
std	1049.189787	8.702624	8.702624
min	0.000000	107.000000	68.000000
25%	908.250000	130.000000	91.000000
50%	1816.500000	130.000000	91.000000
75%	2724.750000	130.000000	91.000000
max	3633.000000	130.000000	91.000000

Fig. 2 Description of training (left) and testing (right) datasets through *pandas*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2400 entries, 0 to 2399
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   index                2400 non-null   int64
 1   id                   2400 non-null   object
 2   sequence             2400 non-null   object
 3   structure            2400 non-null   object
 4   predicted_loop_type  2400 non-null   object
 5   signal_to_noise      2400 non-null   float64
 6   SN_filter            2400 non-null   int64
 7   seq_length           2400 non-null   int64
 8   seq_scored           2400 non-null   int64
 9   reactivity_error     2400 non-null   object
 10  deg_error_Mg_pH10    2400 non-null   object
 11  deg_error_pH10       2400 non-null   object
 12  deg_error_Mg_50C     2400 non-null   object
 13  deg_error_50C        2400 non-null   object
 14  reactivity           2400 non-null   object
 15  deg_Mg_pH10          2400 non-null   object
 16  deg_pH10             2400 non-null   object
 17  deg_Mg_50C           2400 non-null   object
 18  deg_50C              2400 non-null   object
dtypes: float64(1), int64(4), object(14)
memory usage: 356.4+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3634 entries, 0 to 3633
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Index                3634 non-null   int64
 1   id                   3634 non-null   object
 2   sequence             3634 non-null   object
 3   structure            3634 non-null   object
 4   predicted_loop_type  3634 non-null   object
 5   seq_length           3634 non-null   int64
 6   seq_scored           3634 non-null   int64
dtypes: int64(3), object(4)
memory usage: 198.9+ KB
```

Fig. 3 Information of training (left) and testing (right) dataset

Fig. 2 shows 2400 and 3634 sequence samples in training and testing datasets. In the training dataset, all the sequences have a length of 107 nucleotides (nt), while there are long sequences in the testing dataset with a size of 130 nt. In the training dataset, as shown in Fig. 3, there are two features – *signal_to_noise* and *SN_filter*, both are the information regarding the signal-to-noise ratio of each sequence. Bin and Kai [16] used this feature to filter the lines used in the training and validation process. The result of the data filtering is shown in Fig. 4 below, where 2257 out of 2400 training samples had signal_to_noise values greater than 0.25.

```
data_filtered = data_train[data_train['signal_to_noise'] > 0.25]
data_filtered_count = data_filtered['id'].count()
print(data_filtered_count)
```

```
2257
```

Fig. 4 Count of the filtered training dataset

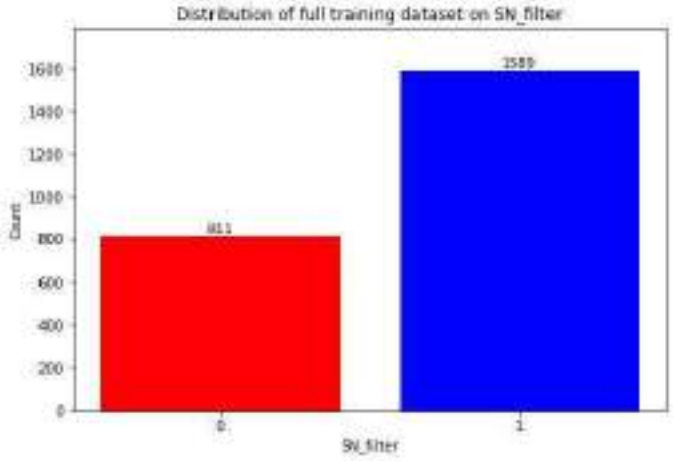

Fig. 5 Distribution of full training dataset on SN_filter

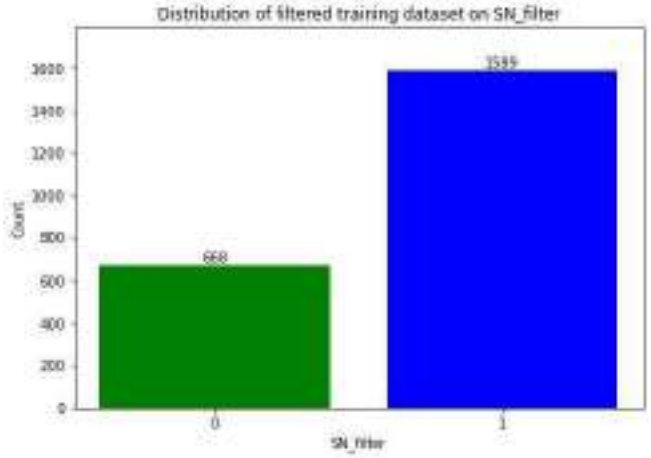

Fig. 6 Distribution of filtered dataset on SN_filter

From Fig. 5 and Fig. 6, the number of samples with a SN_filter value of 1 remains unchanged, while that with a SN_filter value of 0 reduced from 811 to 668. The training and testing datasets are diverse due to the results obtained from the laboratory through experiments, and the testing dataset does not include those five degradation properties and is required to be predicted. In the training dataset, all five degradation properties and their respective experimental error values for each position exist. The dataset has provided data grouping through the feature SN_filter, and the data frame is split by following the part mentioned. Afterward, the values of each type of predicted loop are grouped in a particular list. There are seven types of loops found in the dataset, which are bulge (B), dangling end (E), hairpin (H), internal (I), multiloop (M), stem (S), and external (X) loop. This subchapter aims to observe the distribution of values in all five degradation properties in every loop type. This can help provide information on how each kind of loop contributes to the stability of mRNA secondary structure. There are five degradation properties: reactivity, deg_Mg_pH10, deg_pH10, deg_Mg_50C, and deg_50C.

From Fig. 7 and Fig. 8, several observations can be made. The standard deviations of both full and filtered training datasets remained unchanged based on Fig. 5 and Fig. 6. In the full training dataset when SN_filter is 0, the value spans between 0 and 3. The results tell that the sequences classified as SN_filter of value 0 are statistically more diverged compared to SN_filter of value 1. However, in the filtered training dataset, when SN_filter is 0, the value span between 0 and 1 is similar to that when SN_filter is 1. Comparing their standard deviations to that when SN_filter is 1, the value of stem-loop (S) increased across all five degradation properties. According to the criteria of signal-to-noise filtering, the minimum value across all five properties must be greater than -0.5, and the average signal-to-noise must be greater than 1.0. The deg_[condition] weights depict the likelihood of decay at the base after incubating with the particular situation. The higher the value, the higher the possibility of pruning that base.

(Full Training Dataset) Standard Deviations of loop types in SN_filter = 1

	Reactivity	deg_Mg_pH10	deg_pH10	deg_Mg_50C	deg_50C
B	0.51842	0.65756	0.42666	0.51465	0.332
E	0.56375	0.76723	0.97792	0.73764	0.60316
H	0.54442	0.54292	0.37657	0.42909	0.36151
I	0.4052	0.45629	0.34077	0.40963	0.33544
M	0.39728	0.53924	0.46064	0.44893	0.33178
S	0.25074	0.33111	0.29699	0.30247	0.29309
X	0.34088	0.51693	0.49761	0.42755	0.32654

(Full Training Dataset) Standard Deviations of loop types in SN_filter = 0

	Reactivity	deg_Mg_pH10	deg_pH10	deg_Mg_50C	deg_50C
B	0.89209	1.05727	1.49282	1.16713	1.48356
E	1.13048	1.08817	2.22305	1.44071	1.8978
H	1.18398	1.01926	1.37229	1.21033	1.89413
I	0.39597	0.61398	0.71087	0.68258	0.48874
M	0.41123	0.49258	1.1673	0.59012	0.68677
S	1.15129	0.9738	2.33849	1.44149	2.05284
X	0.63233	0.6051	1.58849	0.80755	1.25187

Fig. 7 Standard deviations of full training dataset on each loop type across five degradation properties

(Filtered Training Dataset) Standard Deviations of loop types in SN_filter = 1

	Reactivity	deg_Mg_pH10	deg_pH10	deg_Mg_50C	deg_50C
B	0.51842	0.65756	0.42666	0.51465	0.332
E	0.56375	0.76723	0.97792	0.73764	0.60316
H	0.54442	0.54292	0.37657	0.42989	0.36151
I	0.4052	0.45629	0.34077	0.40963	0.33544
M	0.39728	0.53924	0.46064	0.44893	0.33178
S	0.25074	0.33111	0.29409	0.30247	0.29309
X	0.34088	0.51693	0.49761	0.42755	0.32654

(Filtered Training Dataset) Standard Deviations of loop types in SN_filter = 0

	Reactivity	deg_Mg_pH10	deg_pH10	deg_Mg_50C	deg_50C
B	0.46838	0.57489	0.49788	0.51234	0.44473
E	0.55102	0.69518	0.97275	0.7106	0.72995
H	0.62194	0.59434	0.62673	0.62931	0.58982
I	0.37871	0.47763	0.51742	0.49565	0.48185
M	0.36841	0.45119	0.51478	0.48045	0.49763
S	0.32505	0.46078	0.60868	0.53437	0.57341
X	0.33734	0.44112	0.48808	0.44553	0.41853

Fig. 8 Standard deviations of filtered training dataset on each loop type across five degradation properties

The reason for standard deviation divergence when SN_filter = 0 may be due to the predicted loop type of those sequences. Since all the dataset entries are obtained from the prediction (mRNA secondary structure) and experiments from the laboratory (decay rates across all five properties), the results must be mixed with noises. For decay rates, the errors of all five properties are provided with the length of seq_scored. For the predicted loop type, He et al. [10] have provided another six biophysical models with a temperature of 37°C and 50°C since the degradation properties to be predicted consist of different temperatures. However, these biophysical models cannot predict secondary structure in different pH values, and there are no degradation rates across all five properties for these extra models. In short, relying only on the form provided by the dataset is insufficient in predicting the degradation rates across all five conditions. In short, stem-loop (S) is the most stable loop across all five properties, and external loop (E) is the least durable loop when SN_filter is 1. The dataset filtering process improves the data quality, as shown from Fig. 7 to Fig. 8 when SN_filter is 0.

B. Data Filtering and Splitting

The training and testing dataset can be acquired in Chapter 3.3. This step is important during the training step as this procedure will affect the performance of the nucleic transformer. The filtering criteria will be based on the existing method by He et al. [10].

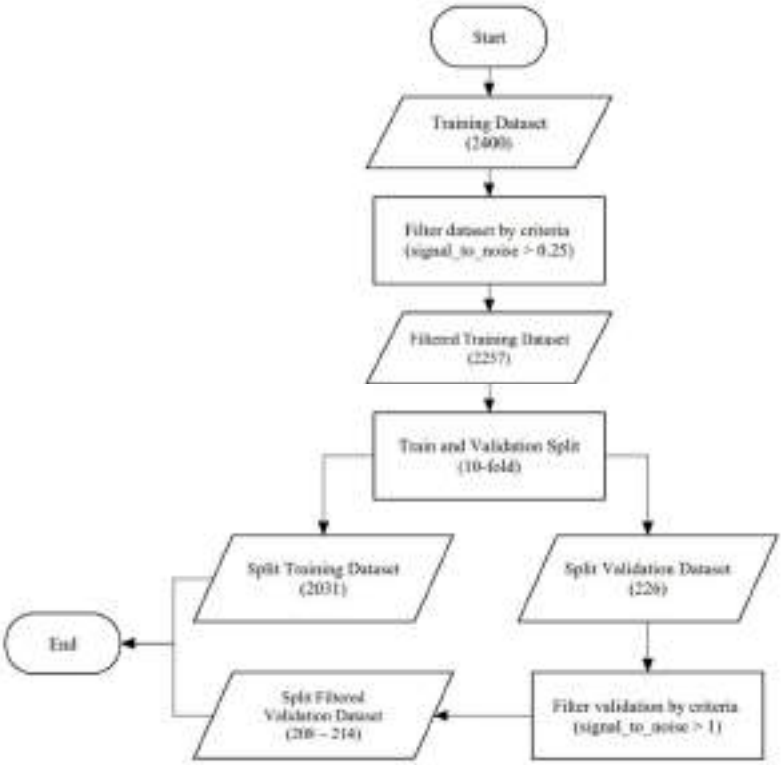

Fig. 9 Flowchart of data filtering and splitting

The original training dataset is read and filtered following the criteria in Fig. 9 above. After the filtering process, the filtered dataset has 2257 out of 2400 samples. Then, the stratified 10-fold splitting process is done, and two outcomes are generated – the indices of split training (2031) and validation (226) dataset. The validation dataset is again filtered by including those with signal_to_noise of value greater than 1. The finalized validation dataset indices are then generated. The amount may vary from 208 to 214 samples.

C. Construction of Nucleic Transformer

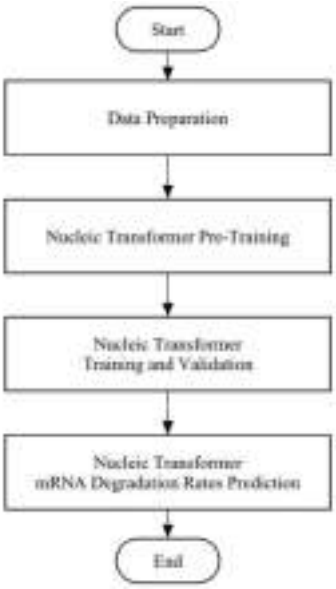

Fig. 10 Overview of enhanced Nucleic Transformer framework

From Fig. 10, the enhanced nucleic transformer construction can be divided into four main phases: data preparation, pre-training of the enhanced model, training and validating the model, and the mRNA degradation rates prediction.

1) Data Preparation: First, the filtered and split datasets are loaded into object instances, as illustrated in Fig. 11. In the figure, pre-train data requires sequences and bppm data only because it learns only the rules of mRNA secondary structure by predicting the true lines of the inputs. Regarding training and validation steps, the labels and error weights provided by the original dataset are loaded into other object instances. These object instances with full sequences data, or train and validation split data, are split into batches for pre-training and training processes. Index 0 to 11 indicates six biophysical models with two temperature results for each model. The data loader provided by PyTorch is used to shuffle and split the dataset into batches. The batches except the last set will have the same size, while the remaining will automatically become the last batch instead of discarding them. Then, all packages will be fitted and shuffled into the transformer model in every epoch.

2) Pre-training Model: The model structure is identical to the existing Nucleic Transformer workflow in the pre-trained model as illustrated in Fig. 12. To pre-train the model, the model takes mutated or masked sequences and the respective bppm data. The objective here is to allow the model to predict the true lines of the mutated or masked sequences with the help of bpm data. The embedding layer embeds the input of the series into a dimension of the model dmodel or ninp = 256. The embedded input will have 256 * 3 because the input of the sequence contains nucleotides sequence, structure, and loop type, and each has a dimension of 256. The projection is a PyTorch Linear layer to transform the embedded sequence inputs with dimension 256 * 3 into a dimension of 256.

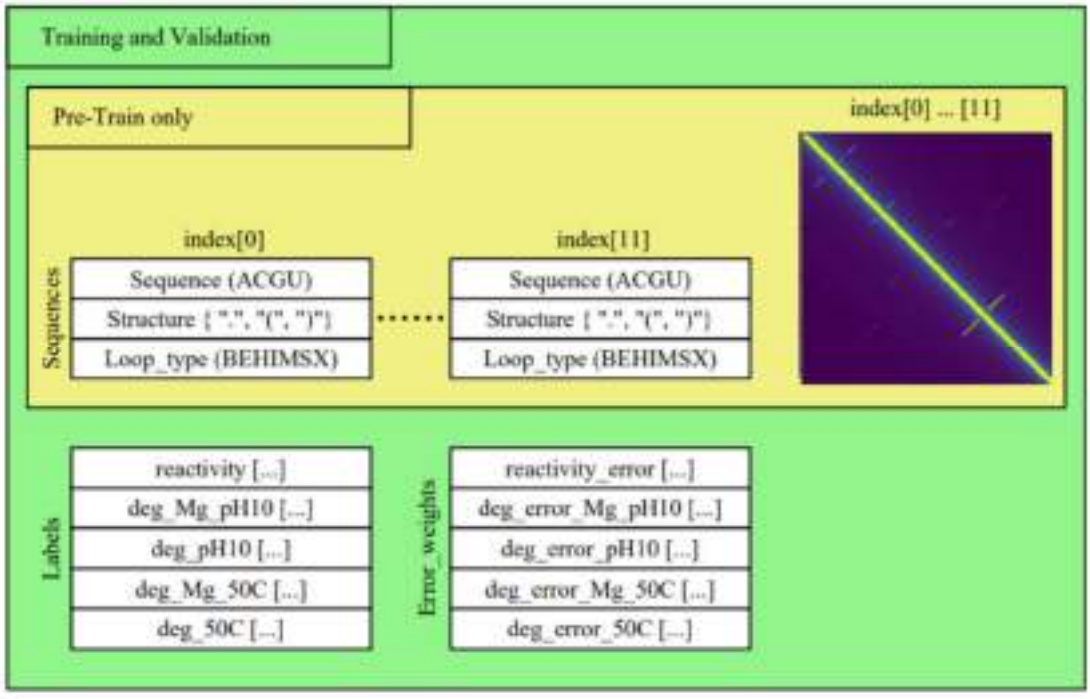

Fig. 11 Structural view of data preparation

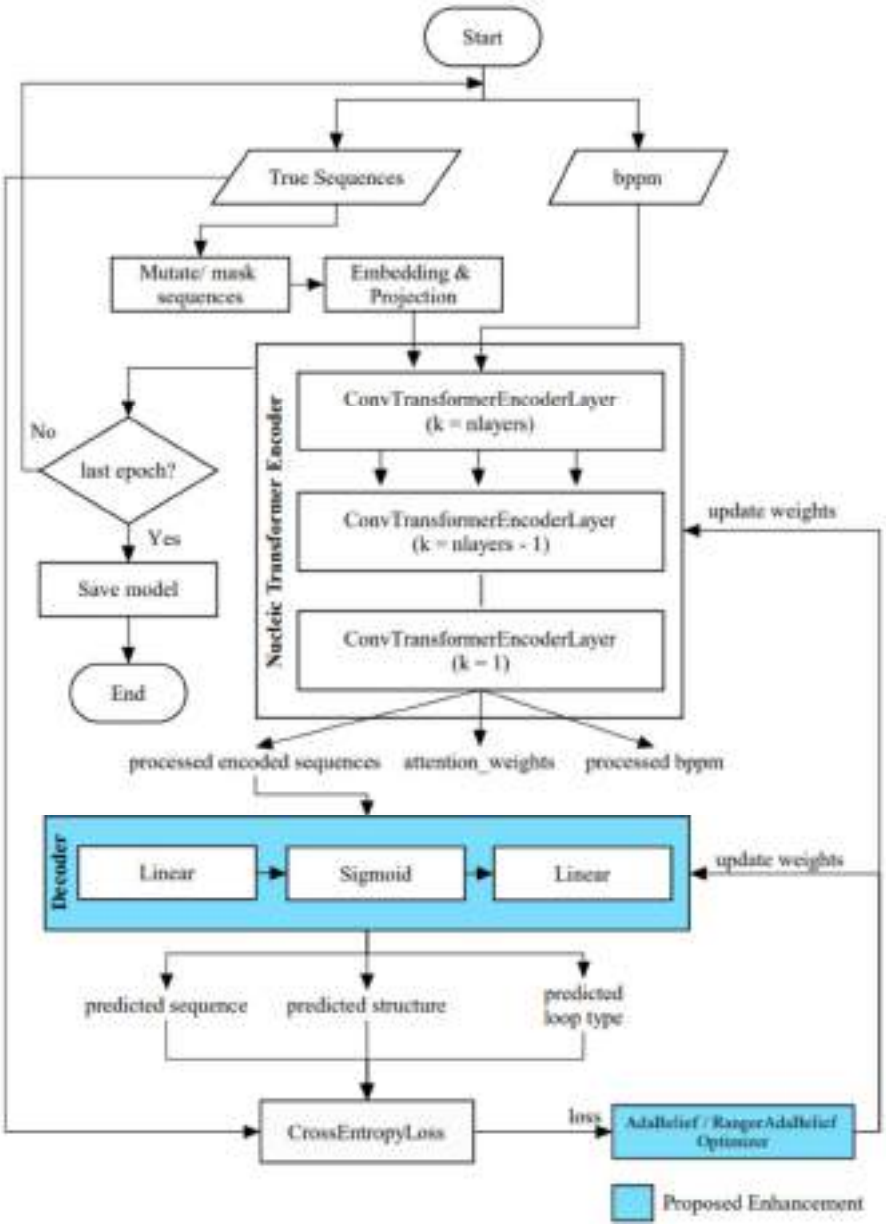

Fig. 12 Pre-Training process detailed workflow

```
3005it [02:14, 22.31it/s]
3029it [01:08, 44.45it/s]Ranger optimizer loaded.
Gradient Centralization usage = True
GC applied to both conv and fc layers
Total number of parameters: 5220351
Epoch [1/5], Step [65/64] Loss: 1.467 Lr:0.001000 Time: 1356.9
Epoch [2/5], Step [65/64] Loss: 0.390 Lr:0.001000 Time: 1355.0
Epoch [3/5], Step [65/64] Loss: 0.256 Lr:0.001000 Time: 1321.6
Epoch [4/5], Step [65/64] Loss: 0.170 Lr:0.001000 Time: 1199.0
Epoch [5/5], Step [8/64] Loss: 0.126 Lr:0.000995 Time: 149.0
```

Fig. 13 Example output of pre-training process

The embedded and projected sequence inputs and bppm data are then forwarded ithe nto Nucleic Transformer encoder stack. ConvTransformerEncoderLayer is a self-customized layer with convolutions and self-attention mechanisms, where the value k represents the kernel of the convolutions to perform kmer-to-kmer interaction mappings. Each layer will generate processed encoded sequences, attention weights, and processed bppm, and these products will be the inputs of the next encoder layer until the last layer.

Afterward, the processed encoded sequences will be decoded in the decoder. The decoder in pre-training settings has three different linear layer configurations used to determine the predicted sequence, structure, and loop type. The loss will be calculated between the predicted sequences,

structure, loop type, and the true sequences data. The loss function used is CrossEntropyLoss. AdaBelief or RangerAdaBelief optimizer will be used. An epoch ends when all the batches of sequence data are processed. In the next epoch, all sequence data is shuffled again and split into batches to ensure the model does not memorize the input sequences. After all, epochs are done, the model will be saved and loaded in the training step. Fig. 13 shows the example output of the pre-training process when the epoch is 5.

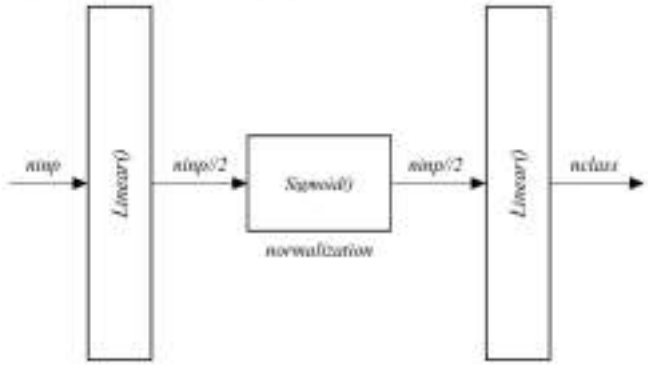

Fig. 14 Architecture of the Proposed Decoder

Besides, the proposed decoder has a normalization layer in between 2 linear layers. The sigmoid layer is used as the normalization layer to transform the output from the first linear layer to values between 0 and 1 inclusive. Compared to the linear decoder used in the existing nucleic transformer, the proposed decoder with normalization can prevent a certain degree of overfitting [2] when training the model. Fig. 14 shows the architecture of the proposed decoder. *ninp* is the input size, and *nclass* is the number of predicted classes.

3) Training and Validation of Model: In the training step, the workflow is similar to the pre-train structure until the Nucleic Transformer encoder. The decoder is modified to implement a sigmoid layer between linear layers. The decoder generates predicted degradation rates and calculates the loss from the self-customized loss function weighted_MCRMSE with true degradation rates and error_weights, all provided by the original training dataset. error_weights assist the weight updating process. The calculated loss is then backpropagated to update model weights. AdaBelief or RangerAdaBelief optimizer will be used.

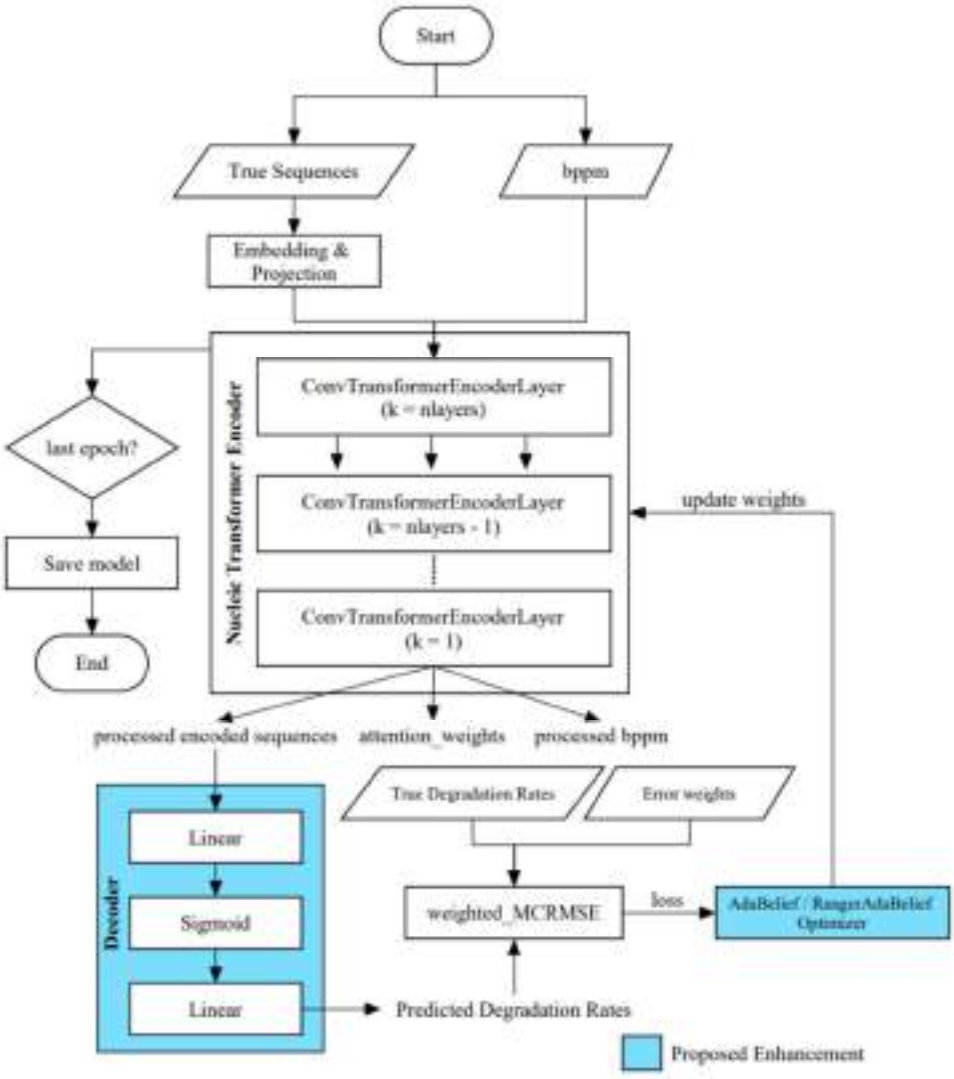

Fig. 15 Training process detailed workflow

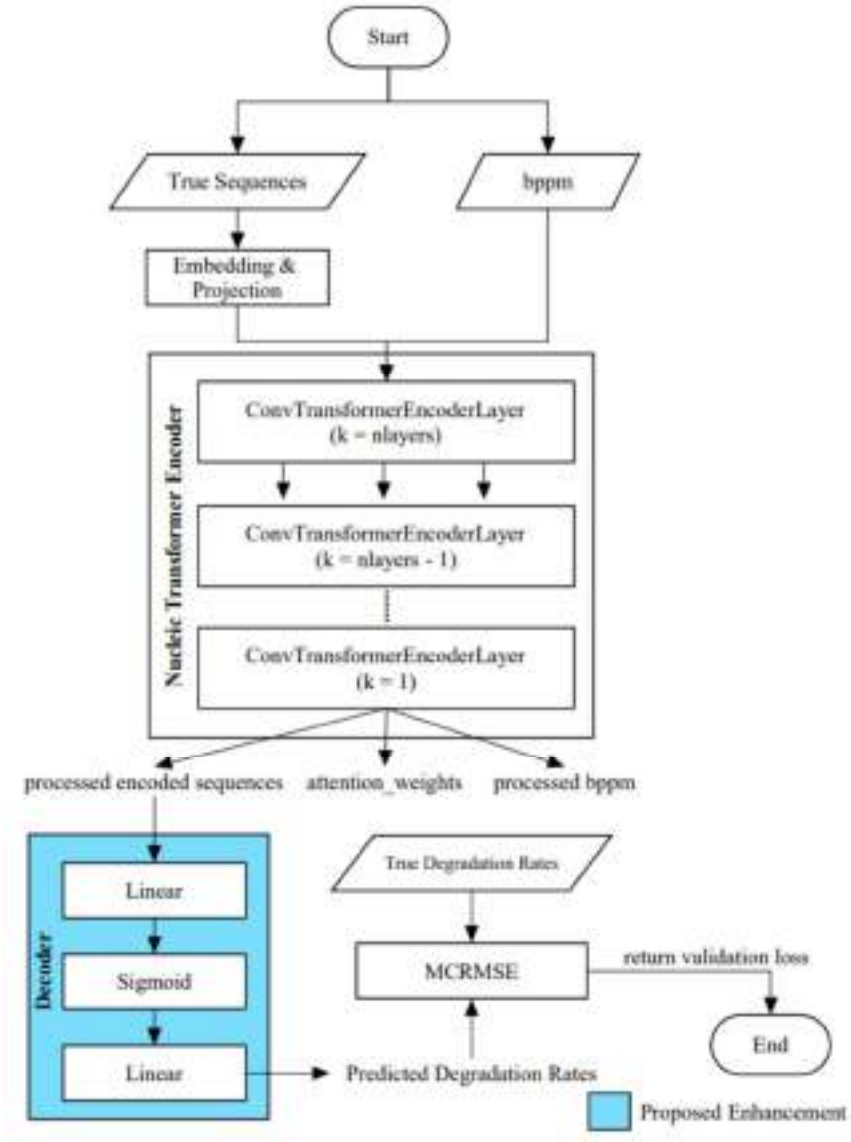

Fig. 16 Validation process detailed workflow

During the validation step, the flow is similar to the training step except for the calculation of loss and the exception of the model weights updating process. This time error_weights is unnecessary because the model weights no longer need to be updated. The validation process is triggered when all the training dataset batches are processed in every epoch; however, the validation process will be triggered after several epochs have been done. The example output of the training and validation process is shown in Fig. 17 below. The details of the proposed decoder have been mentioned in the previous chapter.

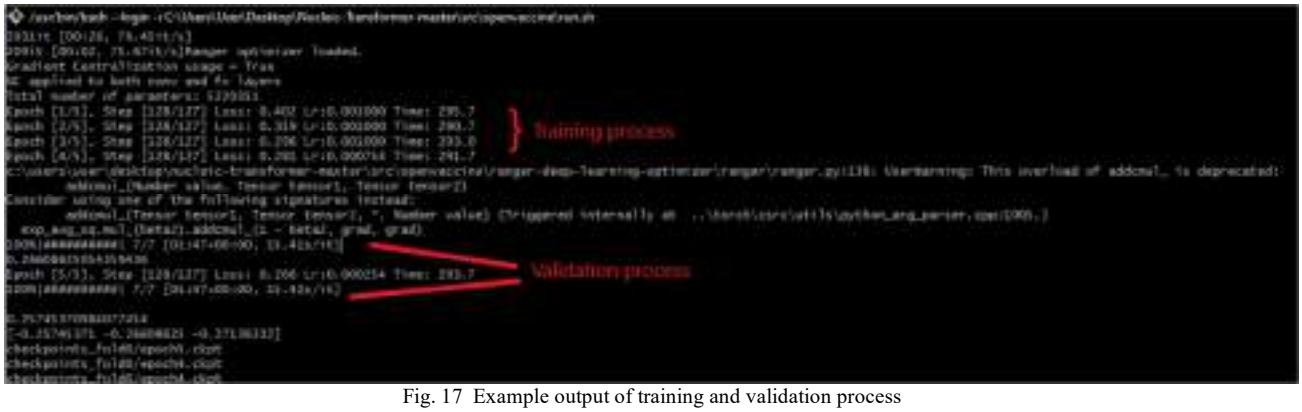

Fig. 17 Example output of training and validation process

4) Prediction Testing: The last process, mRNA degradation rates prediction on the test dataset, is done in the same manner as the validation process. The predicted result is then saved in .csv file for submission on OpenVaccine Challenge on Kaggle. The submission will return the predicted results' public and private MCRMSE scores. The score will be compared with the scores obtained from the existing nucleic transformer model. Fig. 19 illustrates the full proposed Nucleic Transformer framework.

```
/usr/bin/bash --login -i C:\Users\User\Desktop\Nucleic-Transformer-master\src\openvaccine\predict.sh
3005it [01:48, 27.71it/s]
629it [00:19, 31.68it/s]
100%|##########| 47/47 [1:10:05<00:00, 89.47s/it]
100%|##########| 10/10 [10:43<00:00, 64.31s/it]
100%|##########| 3634/3634 [00:00<00:00, 36148.36it/s]
100%|##########| 3634/3634 [00:00<00:00, 26137.58it/s]
```

Fig. 18 Example output of prediction process

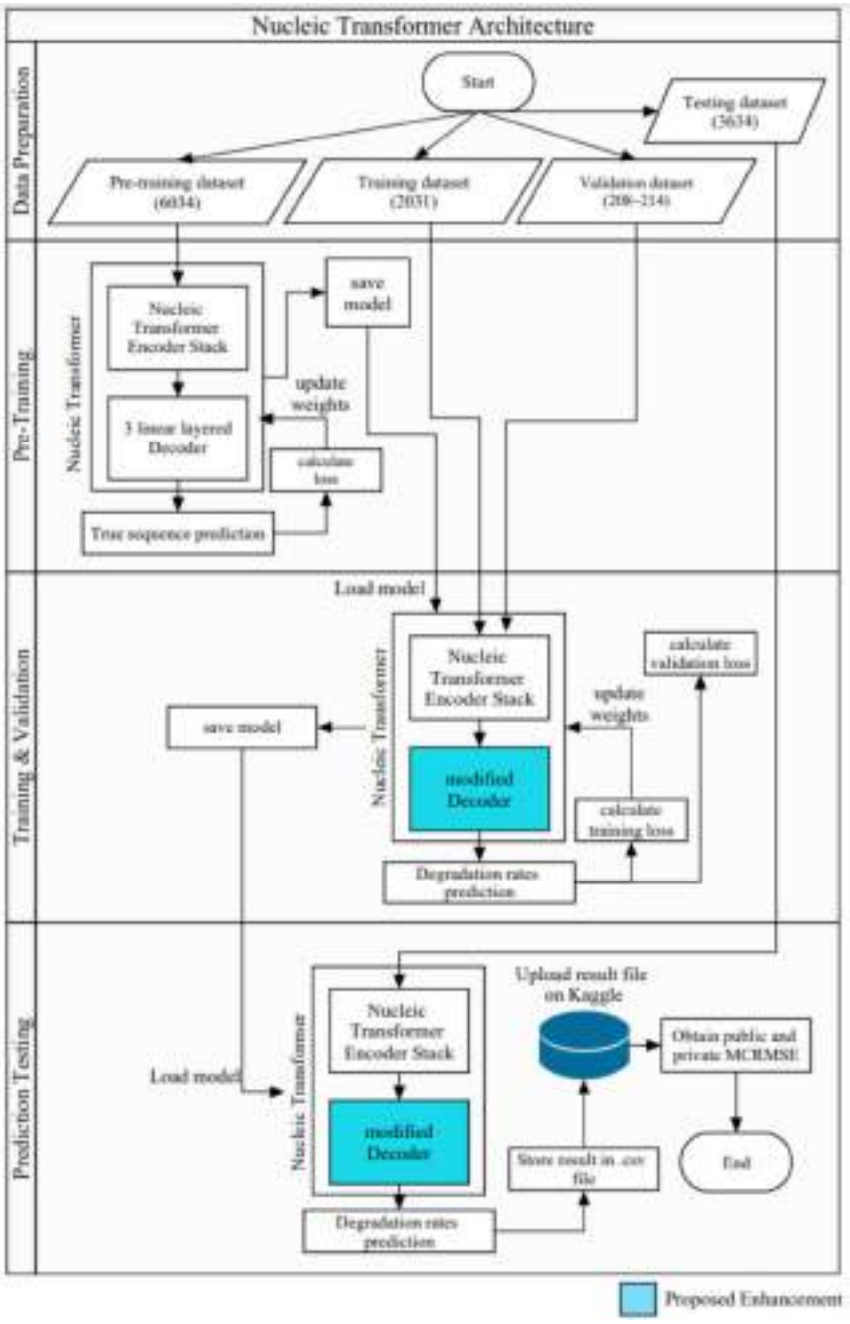

Fig. 19 Full Nucleic Transformer framework

III. Results and Discussion

The measures employed for performance analysis are public and private MCRMSE. The difference between these two metrics is that private MCRMSE is measured using 91% of test data, while the remaining 9% is for public MCRMSE.

A. Performance Metric

Fig. 20 displays the loss per epoch of every combination of optimizers and decoders. Based on the figure, the existing solution using Ranger optimizer and linear decoder (blue line) has a relatively higher loss than AdaBelief/RangerAdaBelief optimizers. Comparing RangerAdaBelief and AdaBelief optimizer, AdaBelief optimizer shows significant and better improvement on both decoders. Initially, the AdaBelief and linear optimizer combination performs the best among others. However, at the end of pre-training, the AdaBelief and proposed decoder combination slightly outperforms the former combination.

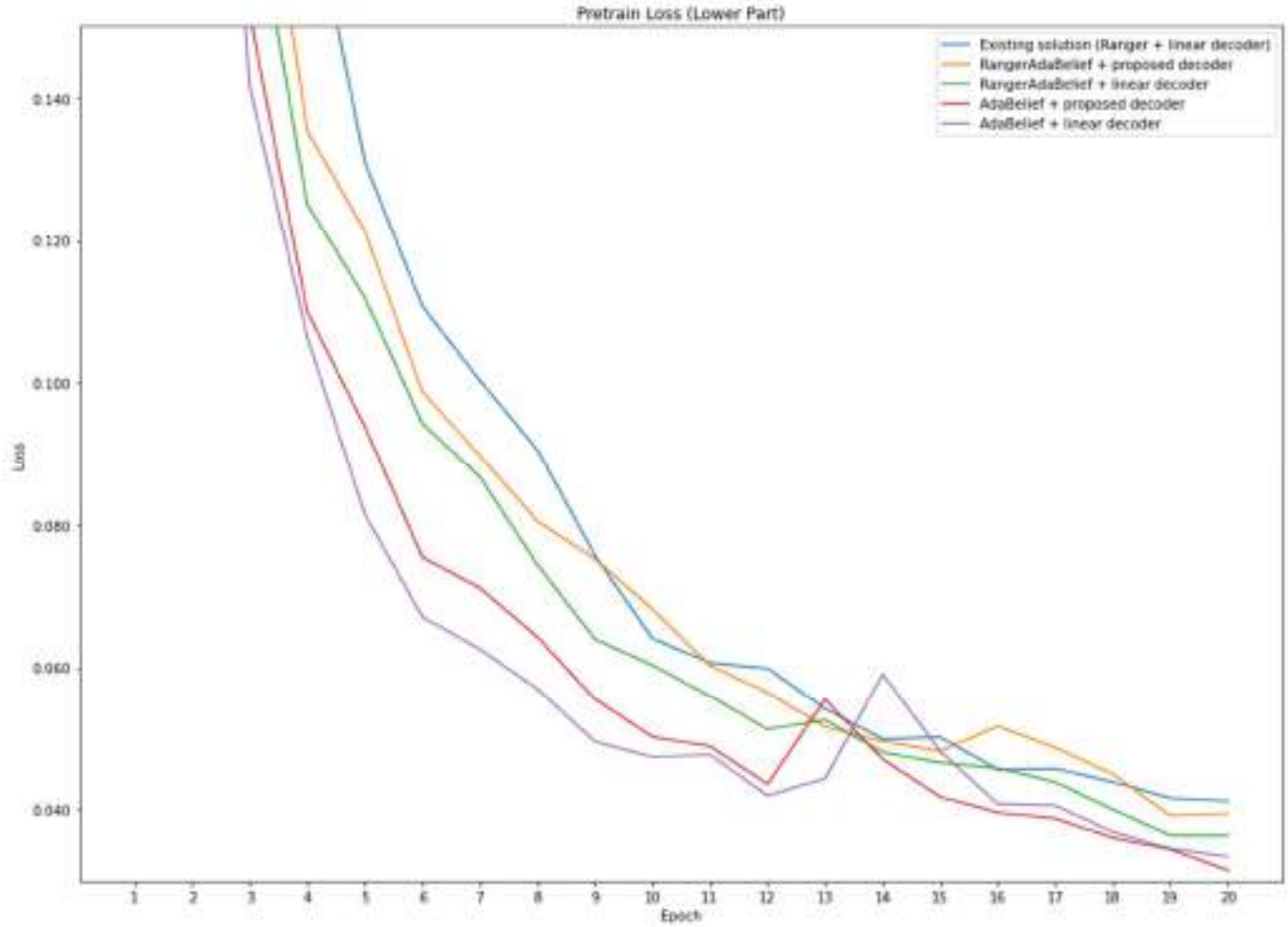

Fig. 20 Pretrain Loss per Epoch

B. Evaluation of Enhanced Nucleic Transformer

In Table 1, AdaBelief optimizers for both proposed and linear decoders were presented. AdaBelief and the proposed decoder have the best performance among the experiments on public and private MCRMSE. Comparing the proposed decoder to a linear decoder while using AdaBelief optimizer, the proposed decoder performs slightly better than the linear decoder in private MCRMSE.

TABLE I
Performance Comparison of Experiments

Optimizer	Decoder Type	Public MCRMSE	Private MCRMSE
AdaBelief	**Proposed**	**0.25034**	**0.36454**
	Linear	0.25152	0.36457

From the perspective of the pre-training performance, the AdaBelief optimizer and linear decoder outperform the rest of the combination at the beginning of the pretrain process. This further proves that AdaBelief optimizer is capable of fast convergence in the early model training process. However, when it comes to an end, the AdaBelief optimizer and the proposed decoder overtake the former combination, indicating that AdaBelief optimizer possesses strong generalization capabilities. At the same time, the model is more complicated when implementing the proposed decoder, which increases the model parameters. By normalizing the values before the final output, the performance of the Nucleic Transformer can be slightly improved. However, implementing the proposed decoder shows its advantages only in public MCRMSE, which includes only 9% of the test dataset. In other words, the advantages of the proposed

decoder can only be shown when there is limited data. When the data is sufficient, the performance might be similar but still better than the linear decoder if and only if AdaBelief optimizer is used.

IV. Conclusion

This paper addresses the enhancement that can be done on the existing Nucleic Transformer. The enhanced Nucleic Transformer consists of the AdaBelief optimizer and the proposed decoder that applies a normalization layer between two linear layers. The enhanced version of Nucleic Transformer eventually provides a 2.79% and 1.38% performance boost in public and private MCRMSE, respectively.

Acknowledgments

This work was supported/funded by Universiti Teknologi Malaysia under UTM Fundamental Research Grant (UTMFR): Q.J130000.3851.21H94.

References

[1] Pardi N, Hogan M, Porter F, Weissman D. (2018). 'mRNA vaccines — a new era in vaccinology'. Nat Rev Drug Discovery 17, 261–279. doi: 10.1038/nrd.2017.243.

[2] Mohammadi Y, Nezafat N, Negahdaripour M, Eskandari S, Zamani M. (2022). 'In silico design and evaluation of a novel mRNA vaccine against BK virus: a reverse vaccinology approach.' Immunol Res 1, 1-20. doi: 10.1007/s12026-022-09351-3.

[3] Miao L, Zhang Y, Huang L. (2021). 'mRNA vaccine for cancer immunotherapy.' Mol Cancer 20(1), 41. doi: 10.1186/s12943-021-01335-5.

[4] Fan N, Chen K, Zhu R, Zhang Z, Huang H, Qin S, Zheng Q, He Z, He X, Xiao W, Zhang Y, Gu Y, Zhao C, Liu Y, Jiang X, Li S, Wei Y, Song X. 'Manganese-coordinated mRNA vaccines with enhanced mRNA expression and immunogenicity induce robust immune responses against SARS-CoV-2 variants'. Sci Adv. 8 (51), eabq3500. doi: 10.1126/sciadv.abq3500.

[5] Li Y, Huang C, Ding L, Li Z, Pan Y, Gao X. (2019). Deep learning in bioinformatics Introduction, application, and perspective in the big data era. Methods. 166. 4-21. doi: 10.1016/j.ymeth.2019.04.008.

[6] Ramkumar T, George PD, Gnanasambandan R, Mohanraj G, Venketesh P. (2022). 'Towards computational solutions for precision medicine based big data healthcare system using deep learning models: A review'. Computers in Biology and Medicine 149, 106020. doi: 10.1016/j.compbiomed.2022.106020.

[7] Zhao L, Xiaoxuan C, Shaoqiang S, Hongwei Z, Xi W, Hong C, Weifu L, Lin L. (2022). 'DeepBSA: A deep-learning algorithm improves bulked segregant analysis for dissecting complex traits.' Molecular Plant 15(9), 1418-1427. doi: 10.1016/j.molp.2022.08.004.

[8] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones, L, Gomez AN, Kaiser L, Polosukhin I. (2017). Attention Is All You Need. Computing Research Repository (CoRR).

[9] Skansi S. (2018). Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence (Undergraduate Topics in Computer Science) (1st ed. 2018 ed.). Springer. doi: 10.1007/978-3-319-73004-2.

[10] He S, Gao B, Sabnis R, Sun Q. (2021). Nucleic Transformer: Deep Learning on Nucleic Acids with Self-Attention and Convolutions. doi: 10.1101/2021.01.28.428629.

[11] Ruhul A, Chowdhury RR, Sajid A, Md Habibur RS, Md Nazmul KL, Md Moshiur R, Md Zahid HK, Swakkhar S. (2020). 'iPromoter-BnCNN: a novel branched513CNN-based predictor for identifying and classifying sigma promoters.' Bioinformatics 7, btaa609. doi: 10.1093/bioinformatics/btaa609.

[12] Liu B, Yang F, Huang DS, Chou KC. (2018). 'iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC.' Bioinformatics 34(1), 33-40. doi: 10.1093/bioinformatics/btx579.

[13] Khare A. (2020). Covid19 feature engineering xgboost. Kaggle. https://www.kaggle.com/arnabark/covid19-feature-engineering-xgboost/notebook.

[14] Castro TF. (2020). HistGradientBoosting Baseline. Kaggle. https://www.kaggle.com/tuliofc/histgradientboosting-baseline.

[15] Imran SA, Islam MT, Shahnaz C, Islam MT, Imam O, Haque M. (2020). COVID-19 mRNA Vaccine Degradation Prediction using Regularized LSTM Model. 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), 328-331. doi: 10.1109/WIECON-ECE52138.2020.9398044.

[16] Amgad M, Suliman MF, Nur Arifin A, David A, Setyanto TW. (2022). 'iVaccine-Deep: Prediction of COVID-19 mRNA vaccine degradation using deep learning.' Journal of King Saud University - Computer and Information Sciences 34(9), 7419-7432. doi: 10.1016/j.jksuci.2021.10.001.

[17] Lin, W. (2020). [covid] AE pretrain + GNN + Attn + CNN. Kaggle. https://www.kaggle.com/c7934597/covid-ae-pretrain-gnn-attn-cnn.

[18] Yosinski, J., Clune, J., Bengio, Y., Lipson, H. (2014). How transferable are features in deep neural networks? Advances in neural information processing systems, 3320–3328.

[19] Juha K, Tapani R, Kyung HC. (2015). 'Chapter 7 - Unsupervised deep learning: A short review.' Advances in Independent Component Analysis and Learning Machines 1, 125-142. doi: 10.1016/B978-0-12-802806-3.00007-5.

[20] Alzubaidi L, Zhang J, Humaidi AJ, Ayad Al-D, Ye D, Omran Al-S, Santamaría J, Mohammed AF, Al-Amidie M, Farhan L. (2021). 'Review of deep learning: concepts, CNN architectures, challenges, applications, future directions.' J Big Data 8, 53. doi: 10.1186/s40537-021-00444-8.

[21] Horwath JP, Zakharov DN, Mégret R, Eric AS. (2020). 'Understanding important features of deep learning models for segmentation of high-resolution transmission electron microscopy images.' npj Comput Mater 6, 108. doi: 10.1038/s41524-020-00363-x.

[22] Perez, L., Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning.

[23] Manoj Krishna M, Neelima M, Mane H, Rao MatchaVG. (2018). 'Image classification using Deep learning.' International Journal of Engineering & Technology 7(2.7), 614. doi: 10.14419/ijet.v7i2.7.10892.

[24] Qing L, Suzhen Z, Yuechun W. (2022). 'Deep Learning Model of Image Classification Using Machine Learning.' Advances in Multimedia 1, 12 pages. doi: 10.1155/2022/3351256.

[25] Ioffe, S., Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift.

[26] Dean J, Corrado G, Monga R, Chen K, Devin M, Le Quoc V, Mao M, Ranzato MA, Senior A, Tucker P, Yang K, Ng AY. (2012). 'Large scale distributed deep networks.' In NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems 1, 1223-1231.

[27] Z. C. Lipton, "The Mythos of Model Interpretability," Queue, vol. 16, no. 3, pp. 31–57, Jun. 2018, doi: 10.1145/3236386.3241340.

[28] Sharma, S. (2022). 'Multi-SAP Adversarial Defense for Deep Neural Networks.' International Journal of Advanced Science Computing and Engineering 4(1), 32-47. doi: 10.30630/ijasce.4.1.76.

[29] Alani, A.A., Alani, A.A., Ani, K.A.A.A. (2021). 'COVID-CNNnet: Convolutional Neural Network for Coronavirus Detection.' International Journal of Data Science, 2(1), 9-18. doi: 10.18517/ijods.2.1.9-18.2021.

[30] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Muller, K.R. (2017). Evaluating the visualization of what a deep neural network has learned. IEEE Trans. Neural Netw. Learn. Syst., 28(11), 2660–2673.

[31] Ribeiro, M., Singh, S., Guestrin, C. (2016). Why should i trust you? Explaining the predictions of any classifier.1135–1144.

[32] Bin L, Kai L. (2019). '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. doi: 10.1016/j.omtn.2019.08.008.