DNA Sequence Classification of Six Pathogens

Moein Hasani
Computer Science
University of Saskatchewan
Saskatoon, SK, Canada
moein.hasani@usask.ca

ABSTRACT

This project aims to classify the DNA sequences of six different pathogens including COVID-19. I compare the performance of Transformer networks and a simple Convolutional Neural Network as the baseline. The dataset includes a training set and five test sets, I experiment with full lengthed and median lengthed sequences and utilized the F1 score as my metric. The baseline achieves a higher score when using full sequences, however, transformers work better when sequences are trimmed to the median length. My findings suggest that the baseline can generalize better than more complex transformers when we use the whole sequences, however when we mask some of the data the more complex models are better choices for classifying the data. You can find my implementation of the work at this link.

KEYWORDS

Transformer Networks, Convolutional Neural Networks, DNA Sequence Classification

1 Introduction

Genomic sequence classification has been a topic of interest in Bioinformatics research for many years [1]. Recognizing different sequences and classifying them is a step closer to the treatment of many genetic disorders. Traditional bioinformatics methods take a lot of time and computational resources for performing such a classification task, however, if we train an ML model for doing so the inference is done in seconds and the new input can be classified quickly. However, now that we have fast processing units and modern Machine Learning (ML) algorithms, this task is easier and more accurate than in the past. After the recent pandemic, the importance of developing our knowledge about pathogens such as SARS and MERS families has shown its importance. My project involves the classification of six different pathogens based on their DNA sequences. These pathogens are SARS-CoV-1, MERS-CoV, SARS-CoV-2 Ebola, Dengue, and Influenza. If we can categorize the newly discovered viruses into the existing groups, we can use the same treatments and methods for fighting them.

There are several research papers on the subject of pathogens and specifically, COVID-19 genome classification. The dataset that I am using in my project is provided by the paper [2]. In this paper, the authors have introduced COVID-DeepPredictor which uses the LSTM network for the classification of 6 pathogens classes. K-mer with different values of K has been applied to the sequences to create a Bag-of-UniqueDescriptors (BoUDs) as the vocabulary. The authors of [3] used ML-based alignment-free methods for the classification of COVID-19 virus genomes. For genome analyses, they combined supervised machine learning with digital signal processing (MLDSP), using a decision tree, and Spearman's rank correlation coefficient analysis for validation of results. Over 5000 viral genome sequences, including the 29 COVID- 19 virus sequences, have been analyzed by the authors. Their method has achieved a 100% accurate classification of the COVID-19 virus sequences and reveals the most relevant relationships among the 5000 viral genomes. In [4], Support vector machines, Naive Bayes, K-nearest neighbor, and random forest methods were used for binary classification of COVID-19 vs. other kinds of coronavirus. In this research, the author has managed to achieve a 93% accuracy on the 2019 Novel Coronavirus Resource Database.

2 Methods

2.1 Dataset

The dataset is gathered by the authors of the paper [2] and was available for download on their website. The dataset includes 18324 samples, the original paper has used only 1500 of this data for training, and the rest is divided into 5 test files. I will use the same split for the training and test as the original paper. The dataset is in the CSV format and each CSV file includes the class number (1-6), class name, and the DNA sequence.

I have converted the sequences to k-mers. In bioinformatics, k-mers are sub-strings of a sequence with a length of k. Kmer method has been used by other works such as [2], [5], and [6] and has been proved to be effective. Using k-mers we convert each of the k nucleotides into a word. E.g. with k = 3 the sequence 'CCAGCTG' turns into the list ['cca', 'cag', 'agc', 'gct', 'ctg']. We make a dictionary of these k-mer words, and we assign a number to each word in the dictionary so when we give the sentences to the network each word is a number. Before we feed the words (their integer equivalent) to the neural network layers, we use word embeddings to convert each token to a vector. The words are transformed into spaces of vectors with arbitrary dimensions such that the words that are closer in the vector space are expected to have the same or close meanings. I used the n = 50

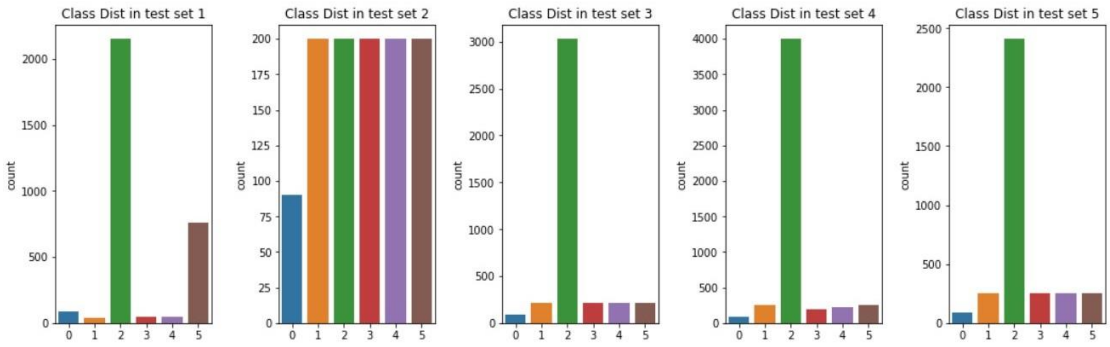

Figure 1 Distribution of classes in the 5 test files

for embedding size which generates vectors of size 50 for each kmer that are learned by the model. Kmer method with k = 3 is applied to the sequences and 259 unique kmers of size 3 are generated from the training set.

The average length of the sequences in the training set is 24861 nucleotides long, and the longest sequence in the training dataset is 30121 nucleotides. Each of the six classes has 250 samples and this makes the training set very balanced. However, the test files are imbalanced and do not include the same number of samples for each class. The distribution of classes in each of the 5 test files is depicted in figure 1. I use test set 2 for the validation purpose during the training of my models. Test set 2 includes 200 samples for classes 2 to 6 and 91 samples for class 1.

I use the T-SNE method to reduce the dimensionality of my data for visualizing the training and validation data in a 2D space. As you can observe in figure 2, the sequences related to SARS-COV-1, SARS-COV-2, and MERS seem to be more closely related to each other. This is because these three pathogens are in the same genus Betacoronavirus, and we can see that it is reflected in the data as well. I think differentiating the mentioned three pathogens from each other would be the more challenging part of the classification task. The visualization of sequences in the validation set also demonstrates the same pattern, with the mentioned three pathogens being closer to each other and the other three being less closely related.

2.2 Transformer Networks

The transformer networks were introduced in the paper "Attention Is All You Need" by Vaswani et al. The paper introduced these networks for the task of language translation, but they have shown state-of-the-art results on many different tasks as well. Transformers are parallelizable and are more efficient than the previous models such as Recurrent Neural Networks. The original paper uses the transformer in an encoder-decoder model architecture, however, I only utilize the encoder part for the classification task.

Transformers benefit from the self-attention mechanism which works by creating three vectors named query, key, and value. All three of these vectors are initially the input tokens. However, each of these vectors is dedicated to a weight matrix which will be updated during the training thus changing the initial values of the vectors. The query vectors are multiplied by the key vectors and the result is normalized and multiplied by value vectors to create the final attention results. Each transformer block includes multiple attention heads, If we have *h* number of heads, then there are *h* sets of query, key, and value vectors. The Multi-Head Attention uses the mentioned process for calculating the attention scores on multiple sets of these vectors, and the operations are performed in parallel on these sets. The *h* sets of attention results are then concatenated and fed to a feed-forward layer.

The transformer architecture benefits from the parallel matrix multiplications and this makes it much faster than RNNs. However, unlike RNNs this architecture doesn't take the position of each token in the sequence into account. In order to solve this issue, the positional encoding technique was introduced. A vector of size *d* (the same size as the embedding dimension) is added to each of the token embeddings. For each position *t* in the sequence and index *i* in the position vector, if *i* is odd the element *i* of the vector is a cosine function with *t* and *i* as inputs, and if *i* is even a sine function with t and i as input. In this way, the word embeddings that are fed to the model also include a sense position in them.

The number of attention heads is set to 2 for the transformer models in my project. Due to the long length of the input sequences, I was not able to feed them directly to the transformer model, therefore I have used a convolution block that includes a conv1d layer and a max-pooling layer to reduce the dimensions of the input. The positional encoded embeddings are fed into the conv block and the result of the convolution is then passed on to the attention block. The results of the attention block are fed to a global average pooling layer and then to a feed-forward layer of 20 nodes before the final softmax layer.

2.3 Baseline CNN

Although Convolutional Neural Networks (CNNs) are mostly used with image data, they are also applied to Natural Language Processing (NLP) tasks. Since the DNA classification is similar to text classification, these networks can be utilized for the task. I use a simple CNN as my baseline method that has 2 blocks of convolution. Each block includes a 1d conv layer with a kernel

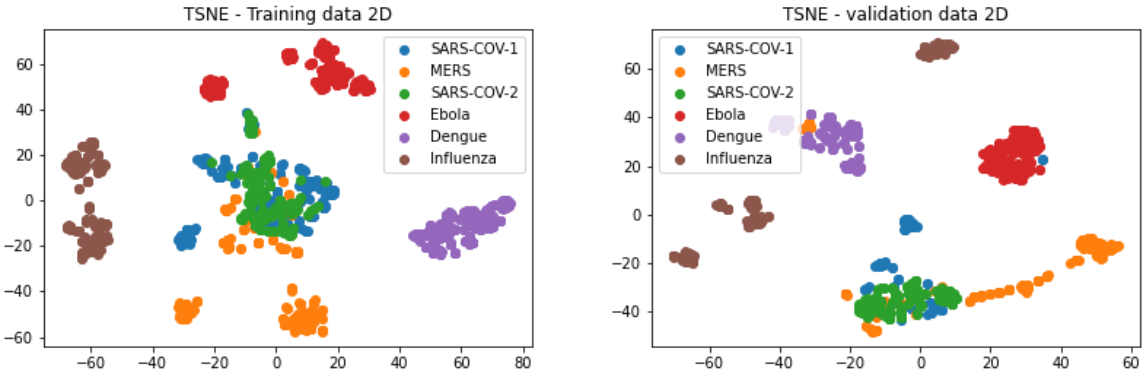

Figure 2 2D visualization of data samples from training and validation sets using the t-sne feature reduction technique

size of 3 followed by a max-pooling layer with a size and stride of 2. The first layer has 32 filters with the size of 3 and the second layer has 64 filters of the same size. I expect the Transformer models to outperform this simple CNN easily, transformer models should be more suited to the lengthy sequences of my dataset.

2.3 Metric

The authors of the paper use the plain accuracy metric which is the number of correct predictions divided by the total number of predictions. In addition to that, I use the f1 score to evaluate my models, and this is because there is a class imbalance in the test files. In these scenarios, the model can only output the class number to which the majority of the samples belong and the accuracy of the model will be high even if the predictions for other classes are wrong. The f1 score is defined as equation 1. This metric takes into account the precision and recall metric thus the model with the smallest number of false positives and false negatives will have a higher score.

$$F1 = \frac{2}{recall^{-1} + percision^{-1}} \quad (1)$$

The reported scores for each model are the result of averaging the F1 score achieved by each model on each of the five test files.

3 Results

3.1 Experimental setup

I have used the TensorFlow framework for implementing the models in this work and the models are trained on an NVIDIA GTX 1080Ti GPU. Due to the limitation of resources, the batch size has been set to 4 and all models have been trained for 20 epochs. The model's weight initialization has been done using TensorFlow's default method "glorot uniform". The global average pooling method has been used instead of flattening in all the models.

I have created three different models by combining CNN and Transformers which I will call CNN_transformers. The first CNN_Tranformer has 32 conv1d filters as its first layer, the second has 64 filters and the third has 128. I have depicted the architecture of CNN_Tranformer_32 in figure 3. I have used drop out with the rate of 10% as the regularization technique with my CNN_Transformer models inside the attention block.

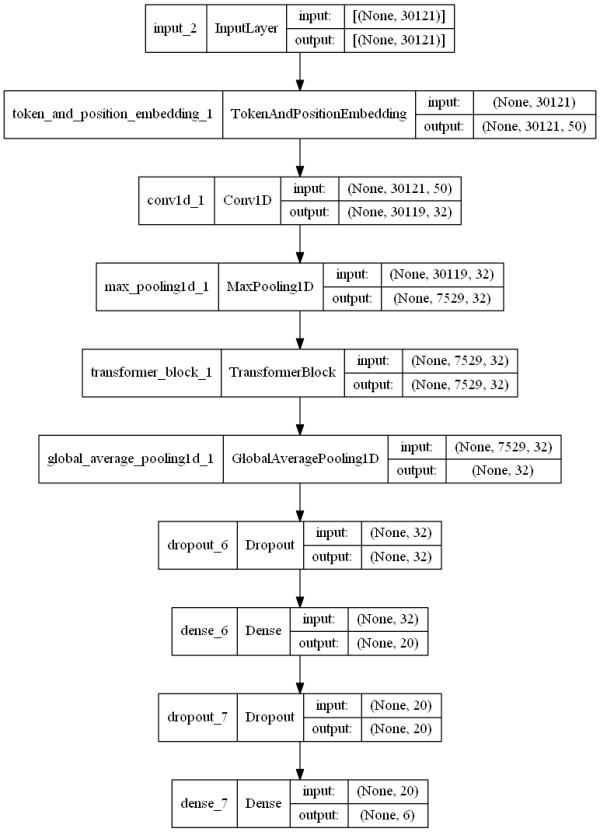

Figure 3 Architecture of the CNN_Transformer_32

3.2 Maximum lengthed sequences

The longest sequence in the training dataset was 30121 nucleotides long. Using the zero-padding method, I have padded all the sequences with shorter lengths to be the same length as the longest one. And the sequences in the test files that are longer than the mentioned number are trimmed from the right end.

The score attained by the baseline and CNN_Transformer model is very close. CNN model manages to reach 0.996, and the CNN_transformer_32 reaches 0.985 which is a higher score than the other two transformer-based models. The baseline model can be trained much faster than the other models, with only 3 seconds required to complete each epoch. On other hand, the Transformer based models are trained relatively slower with 47, 49, and 60 seconds epoch times for the model with 32, 64, and 128 filters in their input. The results of the training are displayed in table x.

3.3 Median lengthed sequences

The median length for the sequences in the training dataset is 20557. All the sequences that were longer than the mentioned sequence were resized by removing the nucleotides from the right side of the sequences. zero-padding was applied to all the shorter sequences to resize them. The baseline only attains a score of 0.906 and the transformer models with 32 and 64 kernels outperform the baseline and achieve 0.986 and 0.959 scores respectively. And The transformer with 128 filters attains the score of 0.896 which is almost the same as the baseline. Such as in the previous experiment, the CNN model is trained much faster than the transformers in this case as well with virtually the same epoch times for each model.

4 Discussion

When using the whole sequence, all models manage to achieve high accuracies and F1 scores. The CNN_transformer model with 32 filters almost attains the same average score as the baseline, however, the other two transformers fail to do so. Believe this might be due to the overfitting of the bigger models perhaps higher rates of dropout and longer training durations could alleviate this problem. And in the case of median-sized sequences, the transformer with 32 filters outperforms the other models in terms of the f1 score. The baseline that did very well with the complete sequences, has a 0.09 drop in its score. After visualizing the confusion matrix of predictions depicted in figure 4, I realized the baseline cannot differentiate between classes 1 and 2 which are Sars-Cov-1 and MERS. If we pay attention to figure 2, we can observe that the samples from these classes seem to be very close in 2D space. In all the five test sets, the average length of sequences in these two classes is bigger than 29,000 nucleotides. When we limit the length of the sequences to 20557, about 9000 nucleotides on average are lost from each sample. I believe this loss makes the detection of these classes that are already somewhat similar harder for the simple baseline model. But the transformer models which benefit from a sophisticated attention mechanism can separate these classes easier. Another interesting insight from the training with the median size sequences is even when we throw away some considerable proportions of the sequences, the models are still able to learn a great deal about the data.

Eventually, my experiments demonstrated the utility of CNNs when dealing with sequential data. They can be trained much faster than other networks as demonstrated in figure 5, and they show excellent results. The baseline achieved the highest scores among all the models when using the whole sequence. Additionally, we can always benefit from CNNs as feature extractors. The input sequences are downsized by the scale of 4 due to the conv layers and the follow-up max-pooling layers, however, the attention heads can still learn the data well and achieve high scores on test files as well.

This project demonstrates that the biggest model doesn't always yield the best result. Perhaps, it is better to always start with simpler models such as a simple CNN and then try more complex models like transformers.

5 Conclusion

In this project, I aimed to classify six classes of pathogens. Three of the classes were almost similar to each other because of the origins of the viruses. I utilized the transformer and attention model for the classification and compared the performance of the transformer with a simple CNN network. Because the sequences in my dataset were thousands of nucleotides long, I was not able to feed them directly to the transformer model so I used a convolution block to reduce the input size before feeding it to the transformers. When I used the complete sequences for the training the simple CNN showed slightly better results than the CNN_Transoformer, and it was trained much faster. However, when I resized the sequences to the median length, the transformer model displayed superior results with a 0.09 higher f1 score in comparison with the baseline.

Model	Maximum Length Sequences	Median length Sequences
Baseline	**0.996**	0.906
CNN_Transformer_32	0.985	**0.986**
CNN_Transformer_64	0.914	0.959
CNN_Transformer_128	0.950	0.896

Table 1 Average F1 scores on the five test files produced by the models

REFERENCES

[1] Mikhail S Gelfand, "Prediction of function in dna sequence analysis", *Journal of Computational Biology*, 2(1):87–115, 1995

[2] Saha Indrajit, et al. “COVID-DeepPredictor: Recurrent Neural Network to Predict SARS-CoV-2 and Other Pathogenic Viruses”, *Journal of Frontiers in genetics*, volume 12,83,2021

[3] Gurjit S Randhawa, et al. “Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens:

COVID-19 case study.” *PloS one* vol. 15,4 e0232391. 24 Apr. 2020, doi:10.1371/journal.pone.0232391

[4] Arslan, Hilal. "Machine learning methods for covid-19 prediction using human genomic data." *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 74. No. 1. 2021

[5] MASANORI Higashihara, JOVAN DAVID Rebolledo-Mendez, YOICHI Yamada, and KENJI Satou. Application of a feature selection method to nucleosome data: accuracy improvement and comparison with other methods. *WSEAS Transactions*
on Biology and Biomedicine, 5(5):95–104, 2008.

[6] Daniel Jurafsky and James H Martin. Speech and language processing (draft). *preparation [cited 2020 June 1] Available from: https://web. stanford. edu/~ jurafsky/slp3*, 2018.

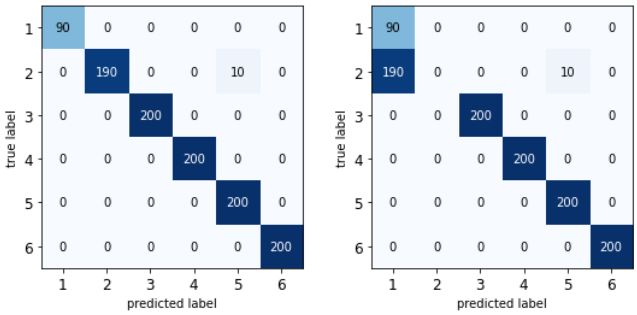

Figure 4 Confusion matrix for predictions of the baseline model. The left side shows the predictions when using the whole sequence, the right size shows the predictions with median lengthed sequences

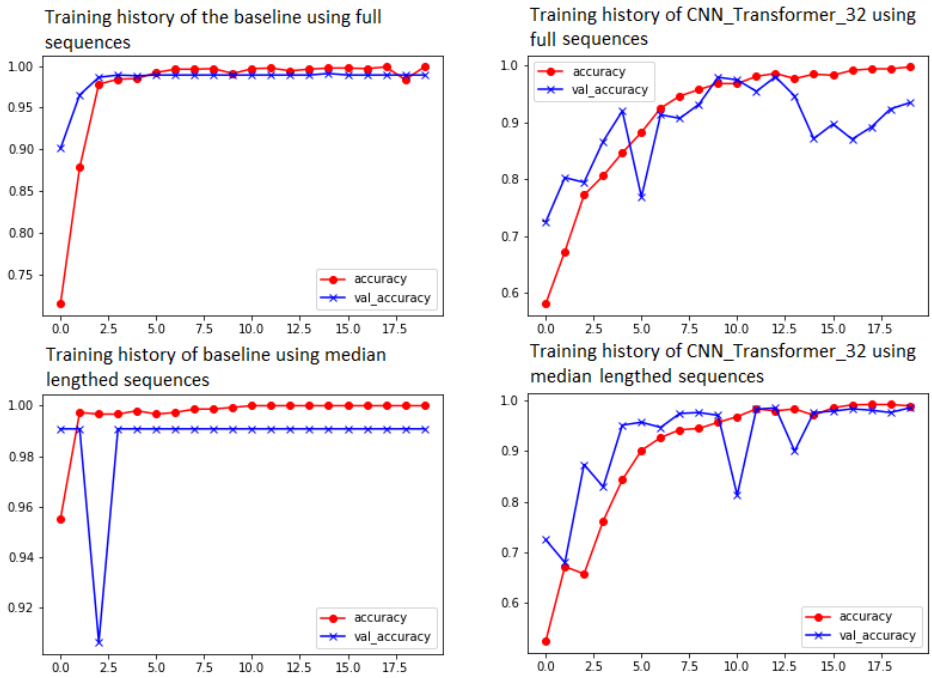

Figure 5 Training history of baseline model and the CNN_Transformer_32. The upper graph shows the training with full-length sequences and the lower ones are with median lengthed sequences. Unlike the transformer model, the baseline in both scenarios reaches a high accuracy in the first few epochs.