Final Writeup

Gabriel Gallardo, Liam O'Connor,
Samuel Murk Caya, Jacques von Steuben

May 12, 2022

1 Introduction

We sought to improve upon the methodology of a paper by Gunasekaran et al., wherein the authors used convolutional neural networks and other hybrid models for the classification of DNA sequences, with a potentially novel application to MERS-CoV, SARS-CoV-1, SARS-CoV-2, dengue, hepatitis, and influenza. Likewise, we developed a convolutional neural network in order to classify the same six viruses, but we approached the preprocessing and architecture somewhat differently. Moreover, we also compared our results and that of the paper to baseline models that make classifications based upon the length of a DNA sequence and upon the guanine-cytosine (GC) content of a sequence.

The architectures in the paper utilize convolutional neural networks (CNNs), embedding layers, LSTMs, and more, so the paper integrates many of the concepts we have discussed throughout the semester! We chose the paper because of its relevance and impact to the real world, especially with an ongoing pandemic. Rapid identification of pathogens is of vital importance to the health and safety of our communities.

2 Methodology

2.1 Data Collection

Like the original paper, we collected our DNA sequences from the National Center for Biotechnology Information (NCBI). More specifically, we utilized the NCBI Virus community portal's public nucleotide database. We selected samples that are whole or nearly whole DNA sequences for each of the six viruses. The respective counts are 692 for MERS-CoV, 351 for SARS-CoV-1, 1,072 for SARS-CoV-2, 10,113 for dengue, 1,732 for hepatitis, and 29,404 for influenza. Because the NCBI Virus portal is a live and constantly updated database, we forwent the impossible task of attempting to determine which DNA sequences the paper used and instead selected samples that were sensible for the changes we made to the original methodology. One can view the criteria

we used for MERS-CoV, SARS-CoV-1, SARS-CoV-2, dengue, hepatitis, and influenza at the embedded links.

2.2 Preprocessing

We preprocessed the DNA sequences for our CNN model. We randomly sampled up to 1000 sequences of each virus and then divided each sequence into subsequences of 300 nucleotides. Because we collected whole or nearly whole DNA sequences, each sample for a virus is likely to be quite similar to other samples for that virus. Therefore, when creating the subsequences, we randomly picked one of the first 300 nucleotides in a sequence from which to start a "reading frame". This ensured that the model trained and tested on most possible 300-mers of each virus; otherwise, nearly identical sequences would heavily populate both the train and test sets. This design choice was informed by a paper by Dasari and Bhukya. We also converted the DNA nucleotides into one-hot encodings. Here we diverged from the original paper in several respects. First, the original paper used the synthetic minority oversampling technique (SMOTE) algorithm in order to compensate for viruses that did not have enough samples. However, for each virus, we collected much more comparable numbers of sequences. Second, the original paper fed entire DNA sequences into an embedding layer, but we divided the DNA sequences into subsequences instead.

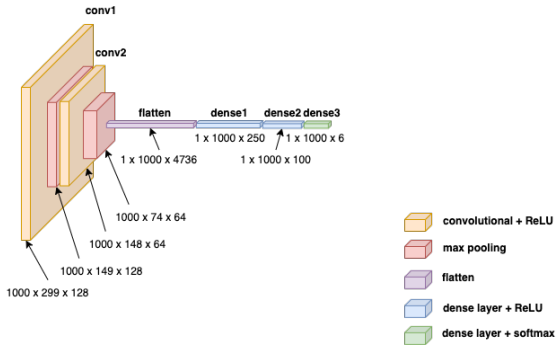
One of our biggest concerns regarding the original paper was the vast irregularities present and left unaddressed regarding data collection and prepossessing. From our exploration of the NCBI's viral genome database, we noted that while the vast majority of data were whole genome sequences, some were very small—reads as small as 8 base pairs (bp) were mentioned in the paper. This resulted in some sequences needing extensive padding for stable input dimensions. Furthermore, many of these viruses have clearly distinct genome lengths ($\approx$ 30 kb for SARS-CoV-2, $\approx$ 13 kb for influenza, etc.). Thus, we concluded that the paper's model was likely not really learning much, and generally categorized based almost only on length—full genome sequences of distantly related viruses were easily distinguished, but nearly all smaller reads and more closely related viruses would be indistinguishable. Moreover, the original paper utilized sequential encoding for the nucleotides, which erroneously introduces a notion of order to the nucleotides that we did not want. That is why we instead used one-hot encoding.

2.3 Architecture

2.3.1 CNN Model

For our full CNN model, we fed the batched subsequences through two one-dimensional convolution layers, each of which uses ReLU for an activation function and is followed by a max pooling layer. We then flattened the output of the second max pooling layer and fed the result through three dense layers. Each of the first two dense layers uses ReLU for an activation function, and the

third dense layer uses softmax in order to generate probabilities for each of the six viruses. Unlike the model in the paper, we did not include an embedding layer in our implementation, because we use subsequences with a standardized length.



2.3.2 Length Model

For the length baseline model, we fed the lengths of the entire DNA sequences through three dense layers. The first two dense layers have an output size of 1000, and the third has output size 6 (our number of virus classes). None of the layers have activation functions. We used softmax cross-entropy with logits in our loss function, which is called within the GradientTape in our train function, to generate probabilities for each of the six viruses.

2.3.3 GC-Content Model

GC-content, or guanine-cytosine content, measures the proportion of bases in a DNA sequence that are either guanine or cytosine. A form of measuring nucleotide composition, this is one of the simplest ways of characterizing genomes due to the fact that different viruses show different properties at their base composition. Our data for this model then consisted of a GC percentage per sequence

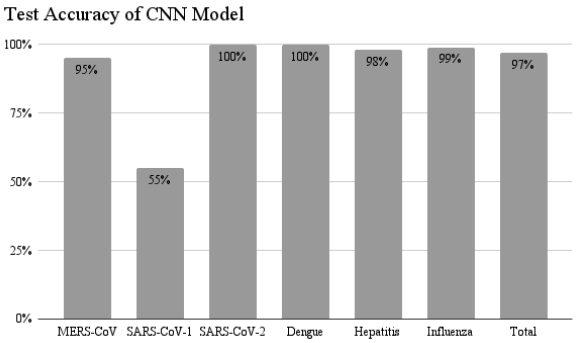
We made use of `Biopython`'s `SeqUtils` library. More specifically, we used `SeqUtils.GC()`, which takes in a DNA sequence string and returns a GC-content percentage according to the following formula:

$$\text{GC-content} = \frac{G + C}{A + T + G + C} \times 100$$

For the GC-content baseline model, we fed the GC-content percentages of the entire DNA sequences through three dense layers. The first two dense layers have an output size of 1000, and the third has output size 6 (our number of virus classes). None of the layers have activation functions. We used softmax cross-entropy with logits in our loss function, which is called within the GradientTape in our train function, to generate probabilities for each of the six viruses.

3 Results

3.1 CNN Model



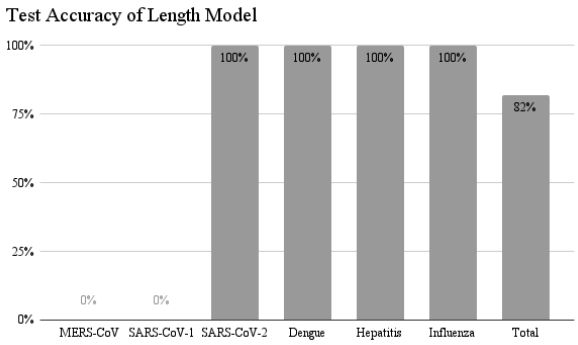
For the full CNN Model, we received a total accuracy of 97% for the six viruses. When we broke down the accuracy into individual accuracies for each of the six viruses, we found that the model correctly classified MERS-CoV with 97% accuracy, SARS-CoV-1 with 55% accuracy, SARS-CoV-2 with 100% accuracy, dengue with 100% accuracy, hepatitis with 98% accuracy, and influenza with 99% accuracy.

Interestingly, the paper's CNN model with label encoding made a similar mistake when it came to classifying SARS-CoV-1. For the test set, the paper's model incorrectly classified SARS-CoV-1 as SARS-CoV-2 62% of the time and only correctly classified SARS-CoV-1 38% of the time. Meanwhile, our model has an accuracy of 55% for SARS-CoV-1! It is intriguing that our model does not confuse MERS-CoV with either of the other coronaviruses in the data set. One would imagine there exists something identifiable in the MERS-CoV subsequences. Nonetheless, without being able to learn the lengths of the whole DNA sequences, our CNN model still performs exceedingly well.

3.2 Length Model

Again, we feel strongly that the paper's model was implemented in such a way that its predictions were overly dependent upon each DNA sequence's length rather than its content. Consequently, we decided to implement a simple neural network, as described before, training it solely on the lengths of the DNA sequences. This simple model yielded around 82% accuracy, correctly predicting virtually all of the SARS-CoV-2, dengue, influenza, and hepatitis samples.

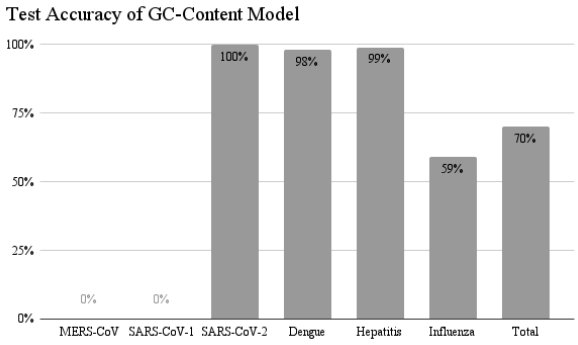
However, the model incorrectly predicted every single MERS-CoV and SARS-CoV-1 test sample. Upon closer inspection, the highest probability for each of these incorrect classifications was always SARS-CoV-2 (followed closely in probability by the correct classification). Ultimately, this consistently incorrect classification occurred because the three coronaviruses have extremely similar



sequence lengths. Therefore, because there were more SARS-CoV-2 samples in our data set than MERS-CoV or SARS-CoV-1, the model learned to classify any test sample of a coronavirus as SARS-CoV-2, because classifying them all as SARS-CoV-2 was correct most often.

The paper's CNN model with label encoding made a similar mistake when it came to the coronaviruses. For the test set, the model incorrectly classified SARS-CoV-1 as SARS-CoV-2 62% of the time. While this error rate is not as egregious as our length model, 62% still seems very high for a multi-layer CNN model. Moreover, if the model in the original paper is indeed incorporating the length of the DNA sequences into its classifications, our simple length model shows the predictive power of such an approach, and how a model could easily exploit the lengths of the DNA sequences in its learning.

3.3 GC-Content Model



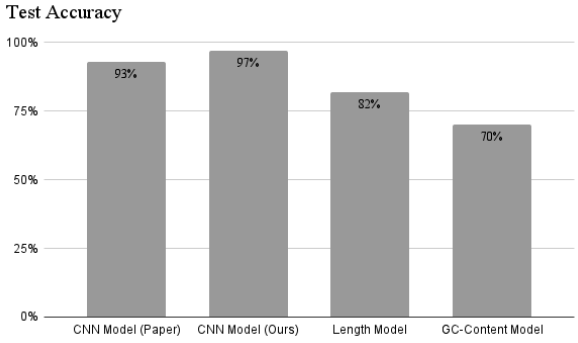
The GC-content model yielded a lower accuracy than the length model, totalling about 70%. However, just like with the length model, each and every MERS-CoV and SARS-CoV-1 sequence was incorrectly classified as SARS-CoV-2. In the GC-content model, this occurs not because the coronaviruses have

similar sequence lengths—as was the case with the length model—but perhaps because the coronaviruses are so similar genetically that their GC-content is virtually identical. Also just like with the length model, because there are more SARS-CoV-2 samples, the MERS-CoV and SARS-CoV-1 are incorrectly classified as SARS-CoV-2, because classifying them all as SARS-CoV-2 was correct most often.

Furthermore, the GC-content model classified dengue and hepatitis almost as accurately as the length model, at 98.4% compared to 100% for dengue and 99.5% compared to 100% for hepatitis. Interestingly, the GC-content model struggled with influenza. The length model correctly classified influenza 100% of the time, but the GC-content model only classified influenza correctly 59% of the time. Upon further inspection, the GC-content incorrectly classifies influenza as SARS-CoV-2 41% of the time—again, likely because there is some overlap in the range of the GC-contents of influenza and SARS-CoV-2

While the GC-content model yields a lower accuracy than the length model, it seems significantly more sensible in its approach. The length model relies entirely on the sequence lengths, which works reasonably well when we feed the length model whole or nearly whole genomes of the viruses. However, if we were to feed it just the first 1000 base pairs for each sequence (or 1000 random base pairs), it would amount to guesswork. The GC-model, on the other hand, would still be able to classify these sequences with reasonable accuracy because it evaluates (to an extent) what each sequence actually contains rather than simply how long it is.

3.4 Conclusions



We see that our CNN model vastly outperformed the length and GC-content models, in large part because neither baseline model managed to correctly classify MERS-CoV and SARS-CoV. We did not use the exact same data set as the original paper, so it is somewhat speculative to compare our accuracy to that of the paper. Still, we see that our model performs extremely well on our data set, and seems to possess a slight edge on the paper's CNN model.

4 Challenges

Early on, we hit a roadblock in replicating the data collection of the original paper. We had assumed the authors utilized an existing data set and therefore that this step would be as simple as locating and downloading the same data set. However, we soon realized the authors had compiled their data from the NCBI Virus community portal's public nucleotide database, which is constantly being updated with new samples. Therefore, it was impossible to exactly match our data set of DNA sequences to that in the paper. This complicated the comparison of the accuracy of our CNN model to that of the author's CNN model. Still, we collected DNA sequences of the same six viruses, and the authors compared their models in their paper to that of other papers, so we still felt comfortable drawing conclusions from comparing our CNN model to theirs.

Moreover, as we delved more deeply into the paper, we unearthed other problems, such as the sequential encoding, the possibility that the model was using sequence lengths in its predictions, and discrepancies between the number of model parameters in the paper's model and our re-implementation of said model. These findings temporarily halted our work on the project. We discussed amongst ourselves and then with our mentor TA. Eventually, we decided to shift our goals for the project to focus on scrutinizing the results and methodology of the paper. This course correction represented another challenge, but one that we strongly feel resulted in a better project.

Specifically, devising a better way of preprocessing the DNA sequences became an unexpectedly difficult part of the project. We toyed with numerous different approaches and deliberated over each one's merits. Ultimately, we decided that collected whole or nearly whole genomes and subdividing them was the best approach given our circumstances. Even within the baseline models, preprocessing was somewhat problematic - despite the fact that returning a set of one-hot encoded labels as well as string DNA sequences should have been relatively simple. Here, problems arose as a result of the SeqUtils Bio library not functioning on tensors or even numpy arrays. Thus we had to make some smart design decisions in order to support labels as tensors and corresponding data as python lists without breaking the rest of the model (and without converting huge DNA sequence tensors to python lists, which would have been incredibly time inefficient).

In conclusion, the lack of guidance we experienced due to the poor execution of our reference paper definitely meant we had to critically think about our design decisions and their sensibilities every step of the way - certainly more than if we had just reimplemented a well-executed paper. However, this also meant that we learned a lot more along the way, and had the chance to draw upon all of the experience and knowledge we had collectively gained throughout the course.

5 Reflection

Our initial plan was simply to re-implement the paper we had selected and to see if we could produce comparable results. However, as we began to work on our implementation of the author's model, we discovered some worrying design choices in the original paper's methodology, particularly in preprocessing. We decided thereafter to deviate from our plan. We would still implement a CNN model, albeit with some modifications, and we would also create a couple baseline models for comparison. Our original base/target/stretch goals became somewhat irrelevant once we changed direction, but our progress in this new direction feels comparable to achieving the target goal. We produced a CNN model using one-hot label encoding with good accuracy across the six viruses, and we also created two other models, although not the CNN-LSTM and CNN-bidirectional LSTM models we had initially discussed. Not only did we feel that the baseline models would be a better use of our time, but the paper showed that of the three models the authors tried for label encoding, the simple CNN model performed the best.

When we first pivoted onto our new path, we assumed our model would perform worse than the models in the paper, because we had eliminated the possibility that our model could learn and use the sequence lengths. Still, we felt strongly that our approach meant our model would learn more meaningful aspects of the DNA sequences of the viruses rather than exploiting sequence length, and we believed this remained truer to the ethos of deep learning. However, our model outperformed any of models in the paper, including the author's CNN model that used label encoding. This realization was a pleasant surprise.

If we could do the project over again, we would have investigated the authors and publications of the papers we were considering more closely and perhaps selected a different paper. The approach of the paper by Gunasekaran et al. seemed sound when we initially read through it, and we incorrectly thought that the paper's being on a US government website was an indication of its importance and quality. This, along with its clean formatting and well-designed graphics lended it credibility at first glance. However, that was not necessarily the case. Had we the time, it would have been interesting to work on the interpretability of our CNN model, perhaps visualizing the filter weights of the convolution layers to see which patterns in the subsequences are important for classifying each virus. Moreover, we could have attempted to solve the inability of our CNN to consistently and correctly classify SARS-CoV-1.

A big takeaway from this project is that simpler is sometimes better. As we have discussed in class, there is a tendency in deep learning to make increasingly complex models that train on increasingly massive data sets in order to achieve marginally better performance. On a smaller scale, we see that phenomenon here. Our simpler baseline models performed quite well, even if not as well as our CNN model. Furthermore, if we had just been considering a subset of the six viruses, perhaps just SARS-CoV-2, dengue, and hepatitis, and if the current per virus accuracies are any indication, it is likely that our baseline models would have performed as well as our more complex CNN model.

Moreover, we saw the importance of preprocessing as another important takeaway. Preprocessing occupied a large portion of our time, much more than we had originally thought, but as a result, we are much more confident that our high accuracy is indicative of the power and generalizability of our model. One is reminded of what many say about data science: the majority of a data scientist's time is spent cleaning data. The same seems to have been true for us while working on this project.