Predicting The Viral Human-Host Probability Using Deep Learning Network

Hina Dixit
Software Engineering
Apple Inc.
hdixit@stanford.edu

***Abstract*—We want to create a model that will be able to predict which viruses have higher human adaption and have an ability into becoming a pandemic. We will be focusing on predicting the viral host adaption based on the viral genome sequence. Such predictions will help the people to focus on specific viruses to find more relevant solutions in advance. This endeavor will be of great value in containing future outbreaks and save millions of lives. We will evaluate our results by testing the currently existing data on Covid-19 and train on viruses like Rabies, etc. We can have different viruses (4-5 different datasets) and the subsequent genome sequences to test against human dna sequencing.**

Keywords—Covid-19, Corona, Deep Learning, RNN, CNN, Tensor-flow, LSTM, RNA, DNA Sequencing.

I. Introduction (*Heading 1*)

In the current era, where Covid-19 has hit the world as a pandemic, most scholars are attempting to create different models to help out the humanity fight with virus and enable our researchers and medical practitioners to come up with better solutions and medication for the people who are affected by the virus. The current stats are devastating and affecting the economy heavily. If only, there was a way to predict which viruses have better human adaption and are highly contagious, we should be able to advance in finding a vaccination for those deadly viruses in a timely manner. This small project is just a try to help with the existing situation and create a small tool that can be useful in avoiding such Pandemics in the future. We are going to take RNA type viruses and predict the human adaptability of those viruses based on a percentage. We will consider different viruses like Rabies, Sars, Covid-19 etc to create a dataset with the respective nucleotide sequences labeled as per their relative scientific names. Our goal is to create a model to predict the probability of human adaptation given a subsequence of the virus. document and are identified in italic type, within parentheses, following the example. We are going to use deep learning models for this, which include recurrent neural network, long short term memory and tensor flow for this project. The end result will be a graph for different viruses (Names on the X-axis) and the probability of their adaptation by a human on the Y-axis.

II. Proposed Idea

A. Method

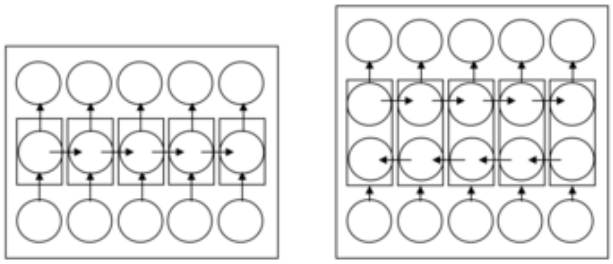
We can use deep neural network architecture for solving this problem. We will be using Python, TensorFlow and Keras as our technical stack. We can use RNN type (Recurrent Neural Network) based algorithm ex. Long short-term memory & CNN (Convolutional neural network) for the same. We will also need to compress the viral dna

sequencing information in a compressed format but ensure to minimize the loss of data. We will often hit the results which are far from idol, we will need to re-align these results to ensure the model training is done correctly.

B. Implementation

i) Previous Method

We collected nucleotide sequencing data from the European Nucleotide Archive and labeled our set correctly as advised by NCBI with respective scientific names. Now for any dataset we are going to divide it into three types: 1. Training 2. Testing 3. Validation. Once, we have prepared our input correctly, our neural network should be enabled to predict the probability of the host's ability to adapt to the viruses. One virus sequence is generated per host. We will create 1 training set per epoch from the existing pool of virus sequences for the given host. This way we are able to randomize our training data and also use all the sequences. Long Short Term Memory networks (LSTMs) are kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in the following work. They work tremendously well on a large variety of problems, and are now widely used. We will create a bidirectional LSTM architecture to understand the context in the dataset consisting of sequences of nucleotide data labeled in both the directions to be able to recognize different patterns within our data that may be useful for us and also enabling randomization of our data. Bidirectional LSTMs are supported in Keras via the Bidirectional layer wrapper which takes a recurrent layer as an argument. Such networks are useful in sequence classification problems as well. For forward propagation , we have forward pass, for the backward propagation we have backward pass and lastly the output neurons are passed. After the forward and backward passes are done, we will be evaluating the weights.



Structure overview
(a) unidirectional RNN
(b) bidirectional RNN

In general mathematical formulation for LSTM is:

$i(t) = \sigma(W(i)x(t) + U(i)h(t-1))$ (Input Gate)

$f(t) = \sigma(W(f)x(t) + U(f)h(t-1))$ (Forget Gate)

$o(t) = \sigma(W(o)x(t) + U(o)h(t-1))$

(Output Exposure Gate)

$\tilde{c}(t) = \tanh(W(c)x(t) + U(c)h(t-1))$

(New Memory Cell)

$c(t) = f(t) \circ \tilde{c}(t-1) + i(t) \circ \tilde{c}(t)$

(Final Memory Cell)

$h(t) = o(t) \circ \tanh(c(t))$

ii) Current Approach

We add CNN network along with the bi-directional Long Short Term Memory to our architecture with 150 nodes excluding the layer pertaining to the output. We also add Keras' Conv1D model which creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. We perform max pooling operation for temporal data and add the leaky version of a Rectified Linear Unit. We train each neural network with an epoch of 500 with equal number of subsequences. For the given epoch, we validate the given model using the validation set comparing to pre-existing known virus-host. When one viral sub-sequence is entered we get viral-host activation

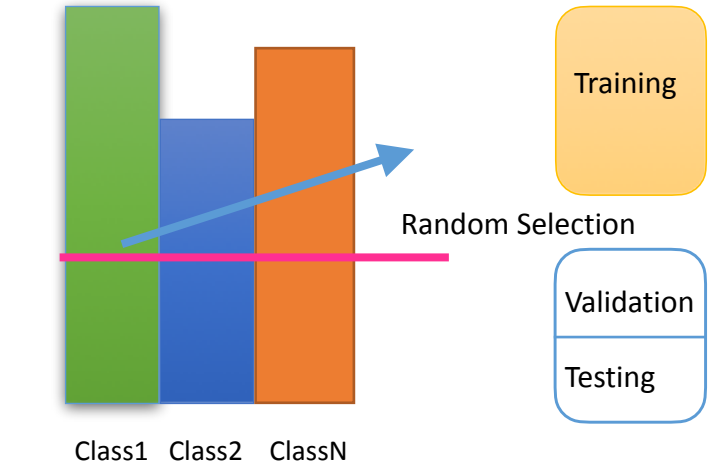
score for the given host output node and which adds upto 1, thus we call it the prediction score. So, each score predicts the probability of viral host prediction in this case. Now we can divide these predictions to give final predictions for the given sub-sequence by either using mean in which we show the mean score per host out of all the subsequences and predicts the host with the highest score thus, having the highest probability of becoming a host to the virus. The other method is to calculate the standard deviation for each score calculated and them use them as weights. We then combine the different predictions in the last stage.

III. Experimental Results

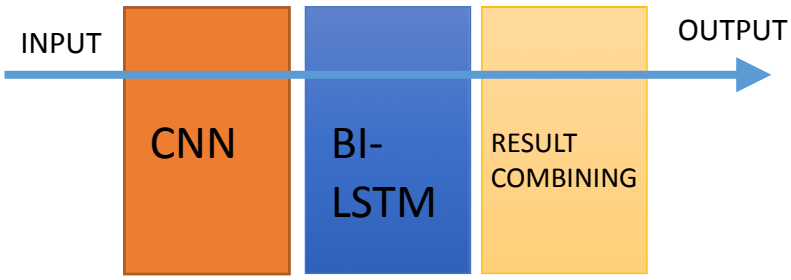
A. Dataset

We are using virus genome structures for RNA type viruses (Covid-19, Rabies, etc) available on Kaggle. The viral genome sequencing is also available on the National Center for Biotechnology Information (NCBI)/European Nucleotide Archive (ENA) database website for free. We can create one entry of genome sequencing per virus in contact with a human with a minimum of 50-100 sequences. For the Dataset, we labeled our hosts and viruses using scientific names. We calculated the 0.95 quantile of all sequence length in order to create the partial sequence, by either truncating the sequence or by padding and converted it into numerical data to do processing using NumPy. We append the subsequence in a repetitive manner in order to get same length subsequence for the inputs in a circular list fashion. For the datasets, we distributed it in following way: Training Set: 60%, Validation Set: 20%, Testing Set: 20%. We consider the sequences as data points and host as a class. We assume that we are creating unbiased training model as per subsequences per host. We also assume that we are limited by the hosts which smallest subsequence of DNA, which means that for viruses for subsequences larger than the host's subsequence, we will ignore the rest of the viral subsequence to simplify the problem which will lead to unbalanced data set causing bias. Instead, we feed randomized combination of subsequences for each host of same number per epoch which helps us reduce the bias for our training sets and we use all available sub-sequence as well.

For converting the data into numerical format, we use one hot encoding, where A-> [1, 0, 0, 0], T-> [0, 1, 0, 0], G-> [0, 0, 1, 0], C-> [0, 0, 0, 1].



Dataset Preparation Figure



Overall Architecture Figure

B. Training Platform

To train the model, we have used the Amazon EC2 P2.XLarge instance, providing 16 NVIDIA K80 GPUs, 64 vCPUs and 732 GB of host memory. We are also using Anaconda to do this locally and to ship it for easier usage. We use Tensorflow as our backend.

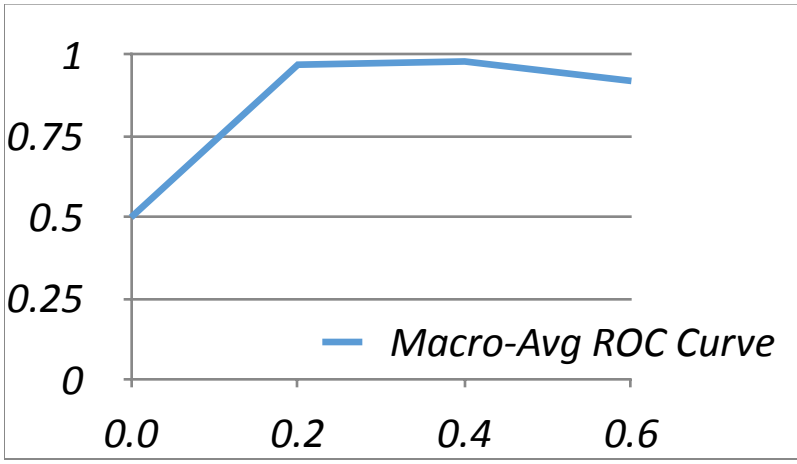
C. Results

We are going to calculate our results in terms of the following three: 1. Standard 2. Mean 3. Standard Deviation. Standard results will display original accuracy for sub-sequence. Mean will imply the mean activation score per class over all the subsequences and predicts the class with the highest mean activation.

We create three different datasets, each containing multiple virus genome sequences along with the host genome sequencing. The Rotavirus A has around fifty-thousand subsequences corresponding to six viral hosts. Our accuracy is 81.3%. With Rabies, we got thirteen thousand viral sequences corresponding to nineteen host sequences. The accuracy for the CNN+LSTM model is 71.98%. For Influenza-A virus host we consider 49 classes with a possibility of three lac and ten thousand viral sequences with an accuracy of 43.54%. We validate these results based on different research papers quoted in the references section. Here is a sample output for Covid-19 genome sequence when fed to the model in fasta format displaying the mean exact for each given class of host for the given genome subsequence.

Using TensorFlow backend.
None
all hosts
Gallus gallus: 0.24587133526802063
Struthio camelus: 0.18236668407917023
Meleagris gallopavo: 0.13173215091228485
Anas platyrhynchos: 0.11124585568904877
Sus scrofa: 0.0951138511300087
Cairina moschata: 0.08452598750591278
Homo sapiens: 0.07660625129938126
Equus caballus: 0.031874097883701324
Sibirionetta formosa: 0.01936531811952591
Cygnus columbianus: 0.0097846994176507
Canis lupus: 0.003913487307727337
Chroicocephalus ridibundus: 0.0026402573566883802
Anas clypeata: 0.0009738129447214305
Anser fabalis: 0.0009337968658655882
Cygnus cygnus: 0.0006534393178299069
Anser indicus: 0.000434194429544732
Arenaria interpres: 0.000378929398721084
Larus argentatus: 0.0003566846717149019
Anas acuta: 0.00025644132983870804
Anas crecca: 0.0002325345849385485
Anas discors: 0.00020833799499087036
Anas carolinensis: 0.0002021956315729767
Anser albifrons: 0.000160943265655078
Larus glaucescens: 7.02400939189829e-05
Tadorna ferruginea: 3.0653558496851474e-05
Calidris canutus: 2.2140733562991954e-05
Calidris ruficollis: 1.4937109881429933e-05
Cygnus olor: 8.345333299075719e-06
Anas rubripes: 7.90096146374708e-06
Leucophaeus atricilla: 5.888669875275809e-06
Branta canadensis: 3.5206726352043916e-06
Uria aalge: 1.6310566479660338e-06
Mareca strepera: 1.616739950804913e-06
Calidris alba: 9.57596398620808e-07
Mareca penelope: 8.725043016966083e-07
Mareca americana: 3.235087220332389e-08

The AUC (Area under the curve for graph True Positive Rate vs False Positive Rate) reached for rotavirus A and rabies is 0.98, please see the figure below.



True Positive Rate vs False Positive Rate

Methods	Virus Type	Mean	Std. Deviation	Standard
LSTM	Rotavirus A	86.7	87.5	85.83
CNN+ LSTM	Rotavirus A	85	85.83	81.3
LSTM	Rabies	74.02	79.21	80
CNN+ LSTM	Rabies	71.98	77.63	77.63

Results for two architectures for two viruses.

D. Next Steps

Our Next steps are to improve any anomaly that we detect on the results and continue training our models. We may need to adapt a variant of Long Short Term Memory architecture to see if we are getting better results. We also want to try different other models to improve the accuracy from an average of 65.6% to at the least 80% for all virus

types. We also want to be able to generate graphs for the output representation for various virus types. I think to convert this output in some kind of web-app is one of the necessary next step. I also want to try other IndRNN model type instead of LSTM+CNN as the new approach to improve the model accuracy.

References

1. Jing Li, Sen Zhang, Bo Li, Yi Hu, Xiao-Ping Kang, Xiao-Yan Wu, Meng-Ting Huang, Yu-Chang Li, Zhong-Peng Zhao, Cheng-Feng Qin, Tao Jiang, Machine Learning Methods for Predicting Human-Adaptive Influenza A Viruses Based on Viral Nucleotide Compositions, *Molecular Biology and Evolution*, Volume 37, Issue 4, April 2020, Pages 1224–1236

2. https://bedford.io/papers/castro-flu-prediction-bounds/ Castro LA, Bedford T, Meyers LA. 2019. PLoS Comput Biol 16: e1007683.

3. https://www.kaggle.com/paultimothymooney/coronavirus-genome-sequence

4. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

5. https://2-bitbio.com/2018/06/one-hot-encode-dna-sequence-using.html

6. https://www.statsdirect.com/help/nonparametric_methods/quantiles.htm

7. V Martella, Krisztián Bányai, Jelle Matthijnssens, Canio Buonavoglia, and Max Ciarlet. Zoonotic aspects of rotaviruses. Veterinary microbiology, 140(3-4):246–255, 2010

8. Christine L P Eng, Joo Chuan Tong, and Tin Wee Tan. Predicting host tropism of influenza a virus proteins using random forest. BMC Med Genomics, 7 Suppl 3:S1, 2014.

9.