COVID-19 Case Prediction CNN/LSTM Sequentially Stacked Neural Network Comparison

Robert Martinez
Department of Mechanical Engineering
Michigan State University
East Lansing, Michigan
mart2414@egr.msu.edu

Aarush Mathur
Department of Electrical Engineering
Michigan State University
East Lansing, Michigan
mathura5@msu.edu

***Abstract*—COVID-19 has infected many and unfortunately taken many lives. Using time series confirmed case and fatality data a CNN/LSTM Neural Network can be trained to deliver predictions on expected confirmed cases [1]. The model tested employs normalization techniques on the input confirmed cases and fatalities and their logarithms. In addition, a one hot geoencoding method has been used to reduce the input information to the network further .**

***Index Terms*—Convolutional Neural Networks (CNN), Long-Short Term Memory (LSTM), Stacked LSTM, Stacked CNN**

I. Introduction

This paper, model and surrounding data analysis is performed in response to The White House Office of Science and Technology Policy (OSTP) pulling together a coalition of research groups and companies (including Kaggle) to prepare the COVID-19 Open Research Dataset (CORD-19) to attempt to address key open scientific questions on COVID-19 [2]. Those questions are drawn from National Academies of Sciences, Engineering, and Medicine's (NASEM) and the World Health Organization (WHO). This challenge involves forecasting confirmed cases and fatalities between April 15 2020 and May 14 2020 by region. The provided dataset reports the geographical location (Province/State and Country), number of cases and number of fatalities as well as the date of the record entry.

Similar to authors Livieris et al. the proposed model exploits the ability of convolutional layers for extracting useful knowledge and learning the internal representation of time-series data as well as the effectiveness of long short-term memory (LSTM) layers for identifying short-term and long-term dependencies [3]. Their preliminary experimental analysis illustrated that the utilization of LSTM layers along with additional convolutional layers could provide a significant boost in increasing the forecasting performance [3]. We seek to develop a similar model architecture that we can test for the time series forecasting of COVID-19 cases.

II. Background

A. Convolutional Neural Networks (CNN)

The convolution layer is called the main building block of CNNs because it calculates the convolution of a set of learnable filters and the input data. Each filter is small in width and height but extends through the depth of the input. During convolution, the filter slides over the width and height of the input data and calculates the dot product at each position. This results in a 2D activation map that creates the response of the filter at every position [4].

B. Long-Short Term Memory (LSTM)

Due to the difficulties of Recurrent Neural Networks (RNNs) in learning long-term dependencies we took The LSTM-based models, which are an extension of RNNs. They will address the vanishing gradient problem in a smooth way. The LSTM models will enable the RNNs to keep and learn long-term dependencies of inputs. This has memory to remember information over a longer period of time. Thus it enables reading, writing, and deleting information from their memories. The LSTM memory is called a "gated" cell, where the word gate is inspired by the ability to make the decision of preserving or ignoring the memory information. An LSTM model captures important features from inputs and preserves this information over a long period of time. The decision of deleting or preserving the information is made based on the weight values assigned to the information during the training process. Hence, an LSTM model learns what information is worth preserving or removing [5].

C. Convolutional Neural Networks and Gated Recurrent Neural Networks

The typical order is a convolutional layer or layer followed by a Gated RNN layer or layers. The convolutional layers achieve data transformation to new representations that are best extractors of useful data information/features, while the recurrent neural networks are used to process the time-series nature of data such as speech, videos, or in the case of our dataset, time-sequences. Literature on COVID 19 predictions reveal similar convolutional and then RNN configurations with GRU RNN, but here in this paper we elected to leverage a stacked LSTM configuration.

III. Problem Statement

We are to predict the spread of COVID-19 using a CNN/LSTM Neural Network. The network will deliver a prediction using the given time series dataset. The results of

various CNN/LSTM configurations will be tested and their performance compared by runtime and RMSLE.

IV. Training Dataset

In the training dataset provided by Johns Hopkins' Whiting School Engineering Center for Systems Science and Engineering, date of record entry, countries with province/state if applicable for that entry, confirmed cases and fatalities were included [1]. In addition log of confirmed cases and fatalities was computed and trained on as well. Number of cases and fatalities were also normalized by their largest values so that their resulting entries can be given on a scale from 0 to 1. In order to simplify the given regions in the dataset combinations of state/province and country were turned into a single entity known as a geocode. To further speed up the training process the geocodes were then encoded using one hot encoding, where the geocodes were turned into a vector consisting of all zeros and a one, where each vector is unique to each unique Geo code.

V. Model Architecture

The following model and results were generated on Google Colaboratory using the TensorFlow Keras framework. The benchmark model comparison is based on a model created by Yohan Cheong in response to the same challenge. He provided a general framework for a CNN/LSTM network. [6].

A. CNN/Stacked LSTM - Benchmark Model Comparison

Our benchmark model consisted of time series confirmed case and fatality data being sorted by geocode location, normalized and transformed logarithmically. The resulting geocodes were then one hot encoded to speed up the training process. The resulting training data were fed through two embedding layers before being passed into a 1D convolutional layer following a relu activation layer and pooling. The relu activation function and corresponding nonlinear layer are helpful for the neural network to learn nonlinear dependencies.

This result of the convolutional network was pooled, flattened, repeated and then passed into our stacked LSTM decoder, was normalized and then the output was time distributed. The described model architecture is depicted in Fig. 1. The resulting model allowed us to predict confirmed cases and compare and validate with historical confirmed case data. The root mean square log error (RMSLE) and run time (seconds) has been reported for all cases presented in this study. Parameters tested include 2000 epochs, 100 batches and 100 nodes. Other parameter changes are noted in the following case study comparison.

B. Stacked CNN/Stacked LSTM - Modified Network 1 Model Comparison

This variation of the benchmark sought to compare the performance of adding a 1D convolutional layer with another relu activation layer in a stacked configuration prior to feeding in the embedded geocoded case and fatality data. This model

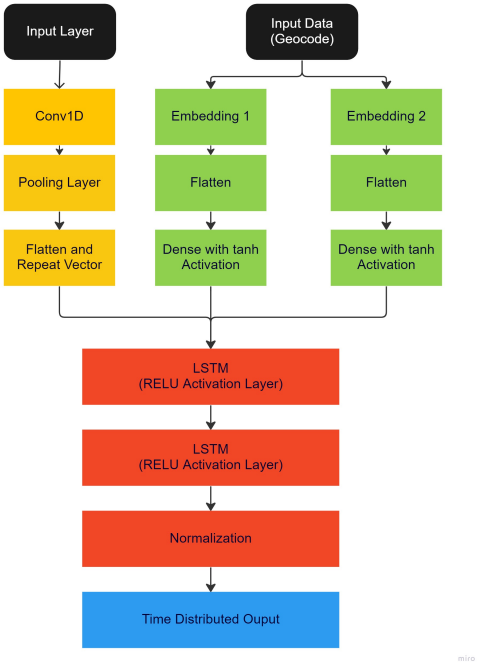

Fig. 1: Model Architecture for Benchmark Model - CNN/Stacked LSTM

architecture is depicted in Fig. 4. All parameters unless otherwise noted have been kept consistent with the CNN/Stacked LSTM - Benchmark Model Comparison.

C. Stacked CNN (increased pooling/Stacked LSTM - Modified Network II Model Comparison

In this variation the pooling was increased after the convolutional neural network layers. All other layers and hyperparameters were kept consistent with the model architecture above for Stacked CNN/Stacked LSTM.

VI. Results - Model Performance

Comparing the best performing model to the worst performing model in terms of RMSLE reveals an interesting trend where in the worst performing many predictions seem to overestimate the confirmed case prediction, while South Korea converges well with the historical trendline. This observation is seen inverted in the case of the best performing model, where most predicted trendlines converge nicely with the historical trendline, while South Korea's trendline is underestimated. This result hints to a need to have different model parameters that could identify and capture social characteristics and customs of a given region that can better train the model to provide an accurate prediction over a greater range of hyperparameter cases.

A. CNN/Stacked LSTM - Benchmark Case Result

The model architecture flow diagram for the CNN/Stacked LSTM Neural Network is shown in Fig. 1. The benchmark model case comparison used parameters 2000 epochs, 100 batches and 100 nodes. The learning rate was varied from .1 to .001 and filter size was either 64 or 128. The results are

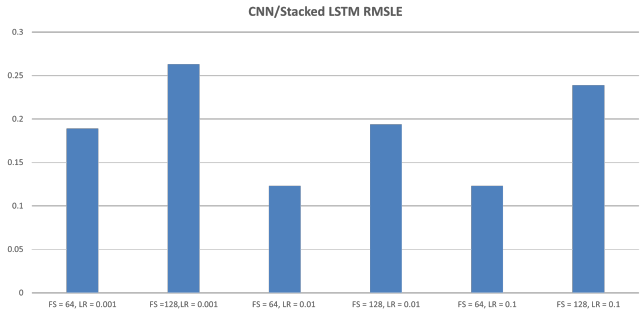

Fig. 2: Benchmark RMSLE Performance Comparison - CNN/Stacked LSTM

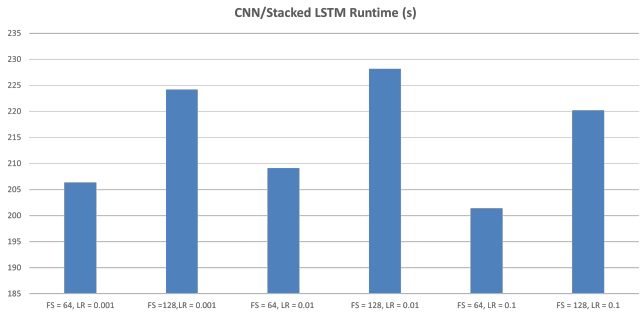

Fig. 3: Benchmark Runtime Performance Comparison - CNN/Stacked LSTM

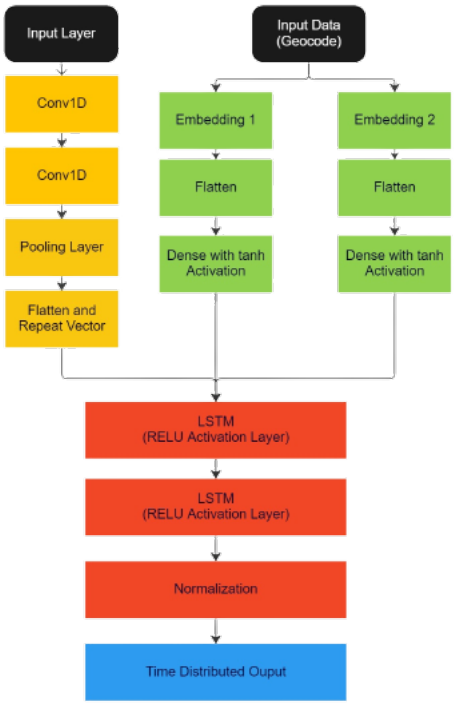

Fig. 4: Model Architecture for Modified Network I and II - Stacked CNN/Stacked LSTM

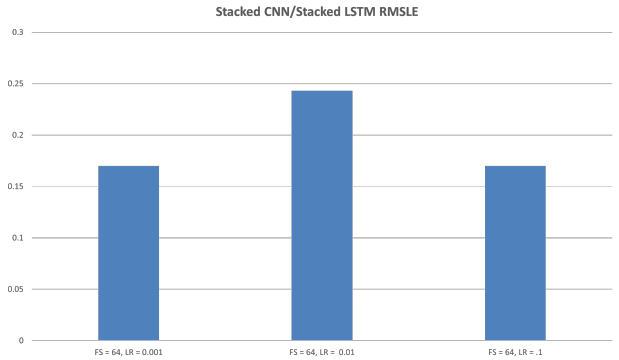

Fig. 5: RMSLE Performance Comparison - Stacked CNN/Stacked LSTM

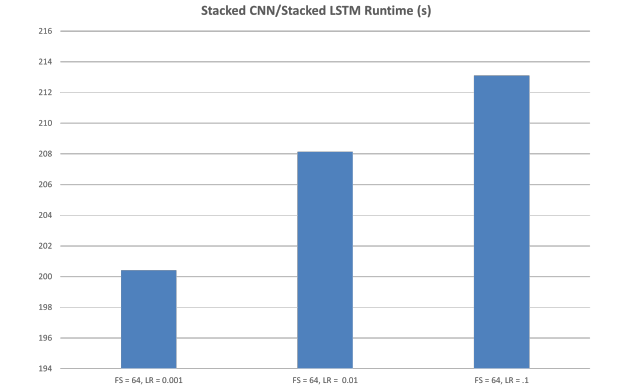

Fig. 6: Runtime Performance Comparison - Stacked CNN/Stacked LSTM

reported in Table I. The case predictions for selected regions using the benchmark model are depicted in Fig. 7. Plots for RMSLE and runtime are given Fig. 2 and 3.

B. Stacked CNN/Stacked LSTM - Performance Result

In an attempt to compare model performance an additional convolutional layer with nonlinear activation function relu (rectified linear unit) was added to the benchmark model in a stacked fashion. The filter size of 64 was kept constant. Similar to the benchmark case: 2000 epochs, 100 batches and 100 nodes. The learning rate was varied from .1 to .001. The

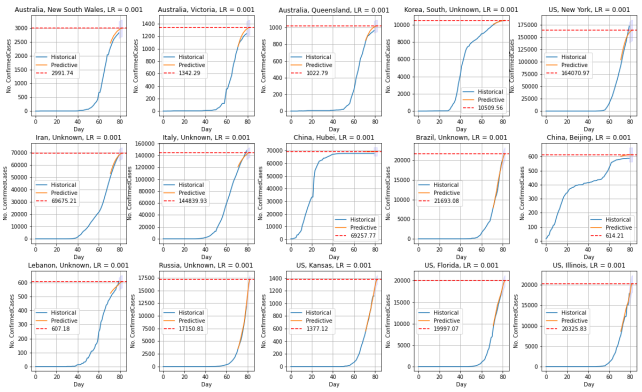

Fig. 7: Confirmed Case Prediction for select geocodes - CNN/Stacked LSTM, LR = .001 and FS = 128 - Worst Performing Model

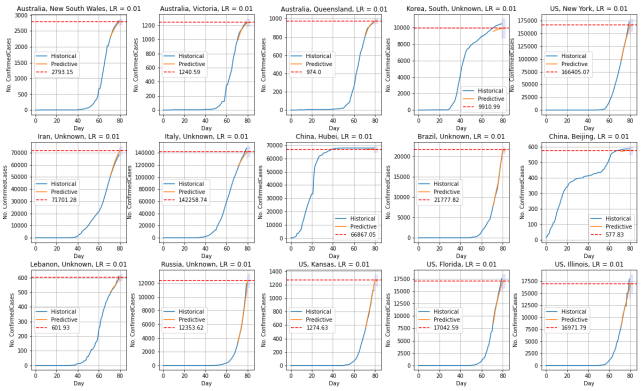

Fig. 8: Confirmed Case Prediction for select geocodes - Stacked CNN (increased pooling)/Stacked LSTM, LR = .01 and FS = 64. - Overall Best Performing Model

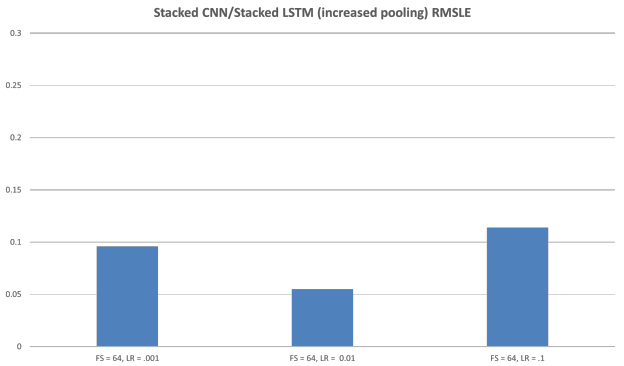

Fig. 9: RMSLE Performance Comparison - Stacked CNN (increased pooling)/Stacked LSTM

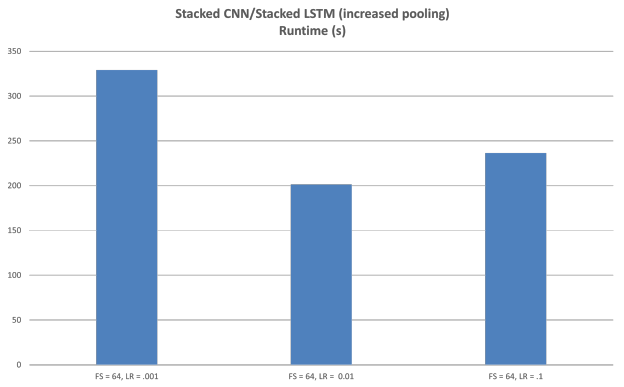

Fig. 10: Runtime Performance Comparison - Stacked CNN (increased pooling)/Stacked LSTM

TABLE I: CNN/Stacked LSTM Model Performance Comparison[a]

Learning Rate	Filter Size	RMSLE	Runtime (s)
.001	64	.188	206.35
.001	128	.262	224.2
.01	64	.123	209
.01	128	.194	228
.1	64	.123	201.38
.1	128	.238	220.19

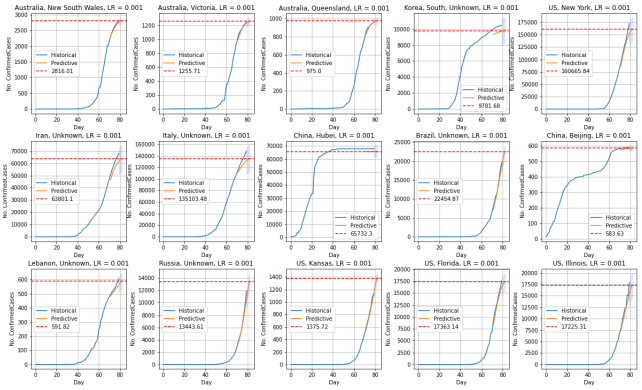

Fig. 11: Confirmed Case Prediction for select geocodes - Stacked CNN /Stacked LSTM, LR = .001, FS = 64

flow diagram for this model architecture is depicted in Fig. 4 and will also be referred to as Modified Network I. Refer to Fig. 5 and 6 and Table II. To see confirmed case predictions for selected regions refer to Fig. 11.

TABLE II: Stacked CNN/Stacked LSTM Model Performance Comparison[a]

Learning Rate	Filter Size	RMSLE	Runtime (s)
.001	64	.170	200.419
.01	64	.243	208.138
.1	64	.170	213.110

C. Stacked CNN (increased pooling)/Stacked LSTM - Performance Result

Our final variation which increased the pooling size from 2 to 3 yielded the best performing model throughout this case study comparison with filter size = 64 and learning rate = .01. In regards to RMSLE this model architecture gave us the best performance of .055. In addition to having an accurate model performance this case gave the second fastest runtime overall. Refer to Fig. 9 and 10 and Table III. To see confirmed case predictions for selected regions refer to Fig. 8.

TABLE III: Stacked CNN (increased pooling)/Stacked LSTM Model Performance Comparison[a]

Learning Rate	Filter Size	RMSLE	Runtime (s)
.001	64	.096	329.275
.01	64	.055	201.389
.1	64	.114	236.599

VII. Conclusions

Our case study results yielded greatly improved performance in the case of a learning rate of .01, filter size = 64, Stacked CNN (increased pooling)/Stacked LSTM. When considering the combination that yields the lowest RMSLE and runtime this model outperforms the rest.

Increasing pooling in the case of Modified Network II (Stacked CNN/Stacked LSTM) led to a substantial increase in runtime for the LR = .001 and .1 cases, but for the LR =

.01 case was a close contender with the the best performing model from Modified Network I. Modified Network I's LR = .001 case revealed fast run times, but also over three times the RMSE as Modified Network II's best performer.

ACKNOWLEDGMENTS

We would like to express our thanks to Professor Fathi Salem for introducing us to CNN and LSTM Networks and providing the resources to conduct this final project.

REFERENCES

[1] "Cssegisanddata/covid-19," https://github.com/CSSEGISandData/COVID-19/blob /master/README.md.

[2] "Covid19 global forecasting (week 4)," https://www.kaggle.com/competitions/covid19-global-forecasting-week-4 .

[3] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural computing and applications*, vol. 32, no. 23, pp. 17 351–17 360, 2020.

[4] A. Ameri, M. A. Akhaee, E. Scheme, and K. Englehart, "Regression convolutional neural network for improved simultaneous emg control," *Journal of neural engineering*, vol. 16, no. 3, p. 036015, 2019.

[5] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 3285–3292.

[6] Yohancheong. (2020) Forecasting covid-19 infections (cnn lstm). [Online]. Available: https://www.kaggle.com/code/yohancheong/forecasting-covid-19-infections-cnn-lstm.