COVID-19 Deaths: Comparing Time Series Neural Network Models

Indrajit Nilkanth Swami

***Abstract*- It has been year since COVID-19 pandemic hit the world. People lost their loved ones to this disease. Forecasting the deaths due to COVID-19 is very challenging problem as there is so much less data to work on. There are many conventional as well as hybrid neural network models used in Time Series Forecasting. This project compares neural network models for predicting in sample data and future 14 days based on last 21 days data. All models are univariate time series models training and forecasting deaths caused by COVID-19. The results are recorded for Multilayer Perceptron, Convolutional Neural Network, Auto Encoder GRU, Auto Encoder LSTM and CNN-LSTM. The primary metrics used for performance measurement is sMAPE and Auto Encoder GRU gives the best results.**

Introduction

The novel corona virus emerged from china in December 2019. COVID-19 has high mutating capacity, every country is facing different type of variants at some point. The infection spreads through air, to control the number of increasing infections options like social distancing, testing large number of people, and containment of infected people are considered. The infections increase at high speed in short time, overwhelming the medical facilities across the globe. Hospitals are overcrowded finding difficulties to allocate beds, ventilators, and provide oxygen to patients. The virus attacks the respiratory system specially lungs, patients with mild symptoms are lost within some days because of damaged lungs, even some parts of recovered patient's lungs are damaged permanently.

The virus is so deadly that it has caused about 3.1 Million deaths worldwide bye the end of April 2021. Some studies used statistical models, and artificial intelligence to highlights impacts in incoming days. A study shows data used from different countries to predict number cases growing daily using deep learning models like GRU, Bi-LSTM, VAE. The major challenge with neural network models arises when we are looking at scenarios where less training dataset is available such that the networks are not fully able to learn limited number of samples [1]. This project focuses on different deep learning techniques, while comparing them to get best results possible. The models are compared based both out-of-sample and in-sample predictions. The result by Auto-Encoder GRU model is better than other. To achieve the great results only building a model is not enough, factors like key hyper-parameters, number of layers, optimizer to be used are very important. All models are also tuned using Keras tuner for deep learning models and several experiments were conducted to find the optimal model that can predict future deaths with minimum error.

In order to overcome the barriers of statistical models, development of deep learning-based models is required to predict deaths. The predictions are helpful for governments to keep track of their patients, reasons behind the deaths, how they can reduce the future occurrences, and what preventive measures they can take.

Related Work

In [1] the statistical models are performing comparable with the deep learning models like Artificial Neural Networks and ConvLSTM model. They [1] are using the same data after augmentation for predictions and training of models. In [2] they are using day level information of COVID-19 spread for cumulative cases from across the 10 mostly affected countries: US, Spain, Italy, France, Russia, Iran, UK, Turkey, and India. They perform predictions using statistical model ARIMA. In [3] using Google Trends of specific related search terms related to COVID-19 pandemic with European Centre for Disease prevention and Control (ECDC) data on COVID-19 spread, to forecast the future trends of daily new cases, cumulative cases and deaths for India, USA and UK. They developed hybrid GWO LSTM where network parameters of LSTM are

optimized using Grey Wolf Optimizer. [4] is also a comparative study of five deep learning models to forecast the number of new cases and recovered cases. Models used were Recurrent Neural Networks, Long Short Term Memory, Bidirectional LSTM, Gated Recurrent Units and Variational Autoencoders. In [5] Long Short Term Memory (LSTM) used to predict the possible ending point of this outbreak to be June 2020. They also presented the 2, 4, 6, 8, 10, 12, and 14^{th} day predictions.

COVID-19 Dataset

There are numerous data sources available those are tracking COVID-19 infections, deaths from different countries. We are focused on tracking COVID-19 in United States. The source of data for this project is from The COVID Tracking Project. They collect, cross check, and publish COVID-19 data from 56 US states and territories in three main areas: Testing, Hospitalization, and Patient outcomes. The dataset in csv format with 391 rows and 18 columns. The project focuses only on deaths as all the models are univariate.

The start date for data collection was 13 June 2020 and end date were 6 February 2021. The data shows no deaths in the start as the infections were increasing and first death was recorded on 26 February 2020. The data used for this project is considered from 26 February 2020 when the first death due to Coivd-19 in United States was recorded. Due to change in start date first 44 rows were excluded and last 347 days data was used in this project. The used dataset can be found here https://github.com/scalation/data/tree/master/COVID .

Neural Networks

Deep learning models learns automatically and extract data from raw data. This feature of neural networks is useful for time series forecasting models. Deep learning models learn random complex mappings from inputs to outputs. These features offer a lot for time series forecasting. Time series forecasting are harder because of complexity of order or temporal dependence between samples. Time series data requires preparation before it can be used train a supervised learning model.

Sliding Window- In univariate time series we have a vector of observations. A supervised learning algorithm requires data in the form of collection of samples with each sample having an input component (X) and an output component (y). To transform time series into samples with input and output component split_sequence function is used. The observations at prior time steps are called lags. These lags are used as input for prediction of current or future time steps. The split_sequence function splits the given univariate sequence into multi timestep input and single or multi timestep outputs. For all models except for Multilayer Perceptron Model input data shape is 3 dimensional and for Multilayer Perceptron Model data is 2 dimensional. The input lags for all models for in-sample data is 5 and for out of sample data it is 21. The predictions on in-sample data gives 1 horizon while predictions on out of sample data gives 14 horizons ahead.

Multilayer Perceptron:

Multilayer perceptron is simple feedforward network model that can be used for time series forecasting using lag observations as input features to predict one or more timesteps. The model has an input layer with some lags. The model has 3 hidden Dense layers and an output layer. The Rectified linear function is used for action of hidden layers. For compilation of the model an Adam optimizer is used.

Convolutional Neural Network:

Convolutional Neural Network are developed for two-dimensional image data and can also be used on one dimensional data such as sequences of text. As our models are univariate time series, we have one dimensional data, samples are provided as input in the form of multiple lags. The model has one dimensional convolution layer followed by Maxpooling1D layer and Flatten layer. Convolution layer is configured using filters and kernel. The Max Pooling layer distil the weighted input feature into the most salient one and Flatten layer flattens all the outputs from Max Pooling layer into a single vector. This vector is input to dense layers and lastly an output layer.

Auto Encoder GRU:

Gated Recurrent Unit (GRU) is a type of recurrent neural network developed for sequence prediction

problem. The GRU is preferred over LSTM when the dataset is small. GRU cells are less complex as compared to LSTM as there are only 2 gates reset gate and output gate. The architecture used in this project contains 3 hidden layers and a Dense output layer. To make normal stacked GRU auto encoder GRU we changed some parameters like return_sequence is set false which is by default is false. The output of GRU layer when return_sequence set to false is 1D vector of values and to give this vector as input to next layer we add RepeatVector layer which acts as bridge between 2 GRU layers. By setting value in RepeatVector it will replicate the 1D vector times the set value making it 2D vector.

Auto Encoder LSTM and CNN-LSTM:

Long Short Term Memory (LSTM) is a type of recurrent neural network developed for sequence prediction problem. As the name suggest LSTM remembers values over random intervals. LSTM cell consists of 3 gates Input Gate, Output Gate and Forget Gate. Forget gate does the work of remembering the values. The architecture used in this project contains 3 hidden layers and a Dense output layer. To make normal stacked LSTM auto encoder LSTM we changed some parameters like return_sequence is set false which is by default is false. The output of LSTM layer when return_sequence set to false is 1D vector of values and to give this vector as input to next layer we add RepeatVector layer which acts as bridge between 2 LSTM layers. By setting value in RepeatVector it will replicate the 1D vector times the set value making it 2D vector. If we add an additional Conv1D layer on top followed by Maxpooling and Flatten we get the CNN-LSTM architecture. In CNN-LSTM there are only 2 hidden LSTM layers.

Keras Tuner:

Keras Tuner is library that helps to select optimal set of hyperparameter for neural network layers. Hyperparameters are variables that govern the training process. During this project we searched for number and width of hidden layers. The tuner used for hyperparameter tuning was *RandomSearch*. Every model ran for 200 combinations of trials and 3 executions each trial. While searching for best model every model was set for 200 epochs and batch size was set to 32 with split validation of 10%. Tuner provides best 10 possible models out of all the trials it ran. [8], [9], [10] give more information about keras tuner.

Rolling Validation for Multiple Horizons:

This is the same technique used in [1] for validation of multiple horizons. A simple form of rolling validation divides a dataset into an initial training set and test set. For example, in this work, the first 60% of the samples is taken as the training set and rest as the test set. For horizon h= 1 forecasting, the first value in the test set is forecasted based on the model produced by training on the training set. The error is the difference between the actual value in the test set and the forecasted value which is used to calculate the symmetric mean absolute percentage error (sMAPE).

Results:

We got differentiable results for all models before and after tuning using Keras Tuner. We predicted the 14 horizons (2 weeks) ahead. These horizons are recorded from 1 to 14. The performance metric used for all results is sMAPE which is performance metric [1], [6], [7]. The results are shown in following **Table 1** and **Table 2. Table 1** contains results for all models before using Keras Tuner while **Table 2** contains results after tuning models with Keras Tuner.

Horizon	MLP	CNN	AE-GRU	AE-LSTM	CNN-LSTM
in-sample	**12.81**	13.59	12.88	14.53	14.85
h=1	21.00	**17.19**	18.00	50.86	34.39
h=2	18.24	**17.49**	18.82	52.57	35.35
h=3	21.04	**19.68**	23.12	62.85	42.88
h=4	20.22	**19.75**	19.66	64.32	39.34
h=5	19.73	**18.43**	19.18	66.23	26.46
h=6	**19.58**	20.21	23.03	64.00	25.88
h=7	24.99	**20.82**	22.08	57.17	35.01
h=8	23.10	**24.17**	25.51	69.20	39.86
h=9	25.17	**21.97**	28.85	68.55	39.85
h=10	27.74	**22.74**	29.44	73.04	48.08
h=11	29.33	**24.28**	26.79	77.64	44.91
h=12	30.94	**24.99**	29.32	84.71	35.18
h=13	27.85	**27.33**	30.23	78.20	36.84
h=14	**26.10**	27.86	34.65	80.92	43.73

Table 1: sMAPE values before Tuning of Models.

The Convolutional Neural Network performs better than other models when they are not tuned. The keras tuner gives better results for Gated Recurrent Units. The results by Multilayer Perceptron model and Gated Recurrent Unit model are very similar, but they are much after tuning using Keras Tuner. The results for 14 horizons of **MLP** are improved by **8%** while **Auto Encoder GRU** are improved by **13%**. The Convolutional Neural Network does not show much of improvement even after tuning. Auto Encoder LSTM and CNN LSTM does not show any good results even after using the tuner. This shows LSTM is not suited for less amount of data, the results are not improved even after tuning the model.

Horizon	MLP	CNN	AE-GRU	AE-LSTM	CNN-LSTM
in-sample	**12.24**	13.51	13.90	13.60	14.43
h=1	15.87	17.18	**14.06**	51.60	33.81
h=2	18.01	17.78	**17.77**	49.37	35.16
h=3	18.14	19.45	**17.65**	40.34	26.16
h=4	18.90	20.72	**18.53**	40.72	29.20
h=5	19.70	20.18	**18.47**	36.04	31.41
h=6	**18.72**	22.26	20.53	32.50	26.75
h=7	22.28	**20.70**	21.85	41.91	21.96
h=8	**23.98**	24.26	24.52	53.56	36.32
h=9	**22.93**	23.24	24.32	58.54	40.76
h=10	23.99	25.29	**23.93**	50.57	37.92
h=11	25.45	24.94	**24.53**	49.02	38.72
h=12	26.01	26.90	**23.16**	44.71	39.60
h=13	26.69	26.82	**25.63**	41.67	36.70
h=14	27.79	**25.87**	27.37	56.66	31.23

Table 2: sMAPE values after Tuning of Models.

The reason behind better performance of GRU as compared to LSTM is number of gates and complexity of computation. The GRU has 2 gates reset and update which are less complex as compared to LSTM's 3 gates input, output and forget gate. Therefore, GRU's perform better than LSTM on small datasets.

In [1] they used ConvLSTM and Neural Network as deep learning models. As we are using the same techniques and dataset as in [1] we are able to compare our results with them. They are using augmented data and they used AutoKeras which is AutoML framework for network architecture search. The **Table 3** represents the comparison between results of [1] and this project.

Horizon	MLP	CNN	AE-GRU	AUG-NN	AUG-ConvLSTM
in-sample	12.24	13.51	13.90	11.04	**10.49**
h=1	15.87	17.18	14.06	**13.06**	14.07
h=2	18.01	17.78	17.77	**14.59**	15.96
h=3	18.14	19.45	17.65	**15.00**	16.99
h=4	18.90	20.72	18.53	**14.22**	18.08
h=5	19.70	20.18	18.47	**15.96**	18.79
h=6	18.72	22.26	20.53	**15.70**	18.01
h=7	22.28	20.70	21.85	**16.64**	20.49
h=8	23.98	24.26	24.52	**18.16**	21.72
h=9	22.93	23.24	24.32	**18.86**	22.72
h=10	23.99	25.29	23.93	**19.81**	23.56
h=11	25.45	24.94	24.53	**18.92**	26.24
h=12	26.01	26.90	23.16	**19.76**	26.99
h=13	26.69	26.82	25.63	**20.34**	25.58
h=14	27.79	25.87	27.37	**20.87**	28.78

Table 3: Comparison with results of [1] (AUG- Augmented, NN- Neural Network)

The **Table 3** shows that augmented data will improve the results. The Neural Network with Augmented data gives best results. The Autoencoder GRU and AUG-ConvLSTM perform at same level, if we use augmented data for GRU it will definitely improve the performance. Even without augmented data AutoEncoder GRU was able to give comparable performance with respect to ConvLSTM with augmented data.

Conclusion and Future work:

For time series forecasting there is need of larger datasets for deep learning models to get better results. When we have small dataset Multilayer Perceptron and Convolutional Neural Network also give competitive results for time series forecasting. GRU still performs better for time series forecasting. LSTM and CNN-LSTM are not good enough for small datasets because of their complex structure. The keras tuner used in this project is effective to get better results. We can further improve results if we do network architecture search on these models as in this project it was done on trial basis. Even the data augmentation is very good option [1].

References:

[1] Indrajeet Y. Javeri, Mohammadhossein Toutiaee, Ismailcem B. Arpinar, and John A. Miller "Improving Neural Networks for Time-Series Forecasting using Data Augmentation and AutoML".

[2] Naresh kumar, Seba Susan "COVID-19 Pandemic Prediction using Time SeriesForecasting Models".

[3] Sikakollu Prasanth, Uttam Singh, Arun Kumar, Vinay Anand Tikkiwal, and Peter H.J. Chong "Forecasting spread of COVID-19 using google trends: A hybrid GWO-deep learning approach".

[4] Abdelhafid Zeroual, Fouzi Harrou, Abdel kader Dairi, Ying Sun "Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study".

[5] Vinay Kumar Reddy Chimmula, Lei Zhang "Time series forecasting of COVID-19 transmission in Canada using LSTM networks".

[6] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 com-petition: Results, findings, conclusion and way forward" International Journal of Forecasting, 2018.

[7] S. B. Taieb, R. J. Hyndmanet al., Recursive and direct multi-step forecasting: the best of both worlds. Citeseer, 2012.

[8]https://www.tensorflow.org/tutorials/keras/keras_tuner

[9]https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html

[10]https://keras-team.github.io/keras-tuner/