Projection of COVID-19 Pandemic - A Deep Learning Approach

Marimuthu S
Christian Medical College and Hospital: Christian Medical College Vellore
Thenmozhi Mani
Christian Medical College and Hospital Vellore: Christian Medical College Vellore
Melvin Joy
Newcastle University Faculty of Medical Sciences
Jeyaseelan Lakshmanan (prof.ljey@gmail.com)
MBRU: Mohammed Bin Rashid University of Medicine and Health Sciences https://orcid.org/0000-0002-9090-3005

Research Article

Keywords: COVID-19 Projection, Convolutional Neural Network, Deep Learning, Forecasting, Long-Short Memory model, Recurrent Neural Network

Posted Date: May 24th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1655899/v1

Abstract

Background: Several mathematical models have been developed to project the COVID-19 cases since the beginning of the pandemic, which are helps for administration to plan effectively and strengthening public health services and facilities. Many researchers incorporated long short term memory model (LSTM) and reported a better performance as compared to classical models. However, they have not clearly explained the workflow and steps involved in the LSTM models in a simple manner. Moreover, these models are not popularized among many biomedical researchers due to lack of expertise. Therefore, our aim of the study is to present such models as a tutorial, in a simple way with Python codes using real time data with good amount of instructions so that many researchers could use these methods to project pandemics like COVID-19.

Data and Methods: A daily case in India from 1st December 2021 to 10th February 2022, and in UK from 1st May 2021 to 10th February 2022 was used to train the models. We used Convolutional Long Short Term Memory (CNN-LSTM) model and simple LSTM models to forecast COVID-19 cases. Both models were validated using data from 11th - 25th February 2022.

Results: CNN-LSTM and simple LSTM models fit very well with R^2 0.95 and 0.97 for India. Validation data, RMSE and MAE of CNN-LSTM model were 9972.81 and 8993.33. And these were 19285.57 and 18870.87 for simple LSTM model. The R^2 value of CNN-LSTM and simple LSTM models for UK data were 0.77 and 0.84 respectively. And validation data, RMSE was 12111.95 for CNNLSTM and 8935.75 for simple LSTM. The MAE of CNN-LSTM and simple LSTM model were 10060.9 and 7824.821 respectively.

Conclusion: Simple LSTM model works better than CNN-LSTM model while training. The performance of CNN is better in the validation for India. Therefore, our suggestion is, train various models instead of sticking into one model to project the future cases and report the range of values. Revise the models weekly once or once in a couple of week as the behaviour of an epidemic may change over a time.

Introduction

The COVID19 pandemic has had a substantial impact on global economic and health-care systems in the recent years. Several countries are experiencing multiple waves due to the mutation of virus. Variants such as Alpha and Delta caused more severe illness; whereas Omicron caused mild disease but rapid spread in the community. The continuous evolution of a virus may lead the healthcare crisis all over the world. In order to control the situation, it is necessary to forecast the epidemic for planning and improving health care facilities.

In order to forecast the future pandemic, frequently updated reliable data are required. From the beginning of the pandemic, several models such as compartmental model, logistic growth model and time series model were widely used for projection. However, the accuracy of prediction and its uncertainties are depends on the assumptions, availability and quality of the data(1). The results may vary significantly when the assumptions are violated, or the difference in the given parameters. During a pandemic like COVID-19, the availability and quality of data keep improving as the epidemic progress, which makes predictions uncertain in the early stages and expected to improve as the epidemic progress. Moreover, an epidemic may not always behave in the same manner as pathogens are likely to behave differently over a time(2).

The susceptible-exposed-infectious-recovered (SEIR), susceptible-infectious-recovered (SIR) models, Agent-based models, Curve-fitting, and Logistic growth model due to the exponential nature of growth of the epidemic are commonly adopted in different biological and social processes(3–8). Especially, the logistic growth model could be relevant for short term projection while SEIR and SIR models could be useful to estimate the maximum number of active cases and the peak time of attaining it. Malavika et al. (2020) used SIR and logistic growth model to predict the COVID-19 pandemic in India(9). They validated the SIR with Italy and South Korea data and incorporated the correction factor for India data. We have also fitted the logistic growth model and found that it under predicts the cases once the pandemic crosses the point of maximum growth. Few researchers have used the exponential growth model to predict the number of cumulative cases. But, this model do not have upper bound and not get stabilised, rather likely to go on increasing(10). In this scenario, the logistic growth model is better preferred option.

Kirbas et al. (2020) compared and evaluated Auto-Regressive Integrated Moving Average (ARIMA), Non-Linear Auto Regression Neural Network (NARNN) and Long-Short Term Memory (LSTM) models and found that LSTM was the most appropriate model(11). Rguibi et al. (2020), and Namini et al. (2018) compared LSTM and ARIMA model and found that deep learning-based algorithms such as LSTM outperform as compared to traditional ARIMA model (12,13).

Whenever, a new phase of the epidemic begins due to new variants of the virus or mutations in the DNA structure, the researchers should validate the models with other countries where the data is available as it is too early to project the infected cases. In such situation it would be appropriate to work with a model that depends on few numbers of time points and parameters if the forecasting is the main intention. The limitations of the above models made us to explore more on deep learning models such as Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM).

RNN is designed to recognize the sequential characteristics in data and use patterns to predict the next likely scenario. Unlike other neural networks such as Artificial Neural Network (ANN), RNN has an internal memory that enables it to remember historical input; this allows it to make decisions by considering current input alongside learning from previous input. It can also effectively handle non-linear time series data by using non-linear activation functions.

However, RNN suffers due to vanishing gradients and exploding gradients problem (14). Long training time, poor performance, and bad accuracy are the major issues in gradient problems. Therefore, Long Short Term Memory (LSTM) models are suggested to overcome the issues of vanishing gradient and exploding gradient, which plagues RNN (15). LSTMs are special kind of recurrent neural networks (RNN), which had capable of learning long term dependencies by remembering information for long period.

Furthermore, our preliminary works show that LSTM models are better as compared to previous models. Moreover, the Deep learning models would be appropriate as the users have the flexibility to decide the hidden layers which would improve the performance. However, they are not popularized among many medical investigators due to lack of expertise. Therefore, our aim is to present such model in a simple way with Python codes using real time data with good amount of instructions so that more researchers could use this method to project the pandemics like COVID-19.

Methods

Recurrent Neural Network:

Rumelhart (1986) introduced the Recurrent Neural Network (16). The unfolded architecture of RNN is given in Fig. 1.The initial hidden state (internal memory) is denoted as h_0. The hidden state at time step 't' is denoted as h_t which is updated by previous time step and has the shape 'number of neurons by 1'. There are two weight matrices and a column vector of bias associated with each RNN layer. A weight matrix and a bias associated with the output layer (Fig. 1).

Work flow of RNN

Kavita et al. (2017) investigated the evolution of RNN model in the last three decades (17). The RNN is an updated version of feed forward neural networks, improved by the addition of recurrent borders which cover adjoining temporal phases, establishing a concept of time to the design. The hidden state (h_t) is computed from the actual sample (x_t) and the previous hidden state (h_{t-1}) in the network i.e $h_t = \tanh(U * x_t + W * h_{t-1} + b_h)$, where U is the weight matrix between input and hidden layer, W is the weight matrix between hidden layer to hidden layer and b_h is the bias matrix. Then result y_t is computed by performing the activation function to state h_t and weight V, that $f\left(V * h_t + b_o\right)$, V is the weight matrix between 'hidden and output layer' and b_o is the bias matrix in the output layer. Hence, at time t – 1, feed x_{t-1} may affect the result y_t at time through such recurrent links.

After computing the outputs (forward pass), calculate the loss /error. Using this loss, calculate the gradient of the loss function for back-propagation. With the gradient that obtained, update the weights (U, W, and V) and bias (b_h, and b_0) in the model accordingly so that future computations with the input data will produce more accurate results. This entire process of calculating the gradients and updating the weights is called back-propagation. Combined with the forward pass, back-propagation is looped over and again for a given number of epochs, and the optimal values of weight matrix and bias are obtained. Unlike the ANN, the weight matrix is same throughout the process.

LSTM models:

LSTM is a neural network model proposed by Schmidhuber et al. (1997)(15). The architecture of LSTM module is given in Fig. 2 which contains forgot gate, input gate, internal state and output gate (18). (Fig. 2)

Work flow of LSTM module:

The work flow of LSTM model was described by Wang et al.(19). The LSTM first determines what information will be excluded from the state of the unit. This operation is implemented by a 'sigmoid' layer called a forget gate. The forget gate read the last hidden state $h\left(t_{i-1}\right)$ and the current input$x\left(t_i\right)$, and gives output a value $f\left(t_i\right)$ between 0 and 1$i.\ e.\ f\left(t_i\right) = \sigma\left(w_f x\left(t_i\right) + w_{hf} h\left(t_{i-1}\right) + b_f\right)$.

Then, it determines what information is reserved in the current unit. This operation includes two parts. The first part uses the sigmoid layer to decide the updating target by using last hidden state $h\left(t_{i-1}\right)$and the current input $x\left(t_i\right)$ that is, $a\left(t_i\right) = \sigma\left(w_a x\left(t_i\right) + w_{ha} h\left(t_{i-1}\right) + b_a\right)$.

The second part uses the 'tanh' layer to construct the vector $c^{'}\left(t_i\right) = \tanh\left(w_c x\left(t_i\right) + w_{hc} h\left(t_{i-1}\right) + b_c\right)$ and $c^{'}\left(t_i\right)$ will play a role in the subsequent unit status updates. Then the last cell state $c\left(t_{i-1}\right)$ updated in the forget gate will take part in the following calculation to attain the current cell state$c\left(t_i\right)$:

$$c\left(t_i\right) = f_t \times c\left(t_{i-1}\right) + a_t \times c^{'}\left(t_i\right)$$

Lastly, the LSTM determines the output for the next moment. The $c\left(t_i\right)$ calculated by the previous step will have two output directions, one of which is directly connected to the unit at the next moment as its input data, and the other requires a filtering process.

The process first passes the current data $x\left(t_i\right)$ and the hidden state from last state $h\left(t_{i-1}\right)$ into the 'sigmoid' layer as the input, and we have the output$o\left(t_i\right) = \sigma\left(w_o x\left(t_i\right) + w_{ho} h\left(t_{i-1}\right) + b_o\right)$. Then, $\tanh\left(c\left(t_i\right)\right)$ is multiplied with $o\left(t_i\right)$, and we gives the current hidden state *i. e.* $h\left(t_i\right) = o\left(t_i\right) \times \tanh\left(c\left(t_i\right)\right)$. Finally, $h\left(t_i\right)$ is passed in the unit of the next moment.

The following two LSTM models have been demonstrated

1. Simple LSTM (ii) Convolutional Neural Network (CNN) LSTM

Steps to construct the LSTM models are given below:

1. Plot ACF/PACF and choose time step / lag
2. Data pre-processing
3. Reorganize the data as in the format of input to the model
4. Model building
5. Fit the model and evaluate the performance
6. Predict for future

Step 1: Plot ACF/PACF and choose time step / lag

Using the time series data, draw auto correlation function (ACF) / partial auto correlation function (PACF) plots then choose the parameter (p or q), which defines the lag (number of time steps that the observations are correlated). The lag size has significant impact on the performance of time series forecasting and it has to be identified using ACF/PACF plot.

Step 2: Data pre-processing

Unlike the classical time series model, LSTM model requires input (X) and the outcome (Y) to train the model which is same as supervised learning such as regression or classification. Therefore, we have to split the data into input (X) and outcome (Y) by using the lag or time step.

For instance, {Xi, i = 1, 2, 3, …, n} be the daily incident cases. Suppose, lag is three, then the fourth day counts depends on previous 3 days (third, second and first) and fifth day counts depends on fourth, third, and second day counts, and so on. Therefore, the data should be organized as follows:

$$\begin{matrix} X & & Y \end{matrix}$$

$$\begin{bmatrix} X_1 X_2 X_3 \\ X_2 X_3 X_4 \\ X_3 X_4 X_5 \\ \ldots \end{bmatrix} = \begin{bmatrix} X_4 \\ X_5 \\ X_6 \\ \ldots \end{bmatrix}$$

Step 3: **Reorganize the data as in the format of input to the model**

The input data X has the shape [samples, timesteps] and it reshaped before feed into the model. The input shape is differing from each model. The input shape of the simple LSTM model is [samples, timesteps, features], where feature is always one for univariate time series analysis.

But CNN-LSTM model has one more argument called 'subsequence'. Split the lag into number of sequence and sub-steps. The sub-steps are multiple of the kernel size in the convolutional layer. For example, lag is six days then split the six into three subsequences and two sub-steps (3*2 = 6). The input shape of the CNN LSTM model is [samples, subsequences, sub-steps, features].

Step 4: Model building

Simple LSTM:

Schmidhuber et al. (1997) proposed the LSTM model which is the simple LSTM configuration model compared to other models with one hidden LSTM layer and output layer(15). But, in our model we used one LSTM layer followed by four to eight dense layers which were determined based on the model performance. The structure of the LSTM model is given in Fig. 3.

CNN LSTM model

LeCun et al. (1998) developed CNN model which is a type of neural network developed to work with two-dimensional image data (20). The CNN can be very effective on automatically extracting and learning features from one-dimensional sequence data such as univariate time series data. A CNN model can be used in a hybrid model with an LSTM backend where the CNN is used to interpret subsequences of input that together are provided as a sequence to an LSTM model to interpret(21). This hybrid model is called a CNN-LSTM.

The Time distributed wrapper function is used to read the subsequence of data separately. The CNN model first has a convolutional layer for reading across the subsequence that requires a number of filters and a kernel size

to be specified. The number of filters is the number of reads or interpretations of the input sequence. The kernel size is the number of time steps included in each 'read' operation of the input sequence. The convolution layer is followed by a max pooling layer that distils the filter maps down to 1/2 of their size that includes the most salient features. These structures are then flattened down to a single one-dimensional vector to be used as a single input time step to the LSTM layer.

Our CNN-LSTM model has one dimensional CNN layer followed by max-pooling layer and flatten layer. Then it has two LSTM layers with 500 neurons followed by four to eight dense layers.

Step 5: Fitting and evaluating the model performance

Fit the model using the train data and evaluate the performance. The Mean Squared Error (MSE) / Mean Absolute Error (MAE) can be used as the metrics to evaluate model performance of both train and test data.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \text{ and } MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

Step 6: Future prediction

In order to predict the future value, we have to feed past data as the input. In the absence of past value we can use our predicted value of the corresponding day as the input. The Python code for all the above steps are given in the link for the future work.

Data:

Daily COVID-19 cases in India were taken from https://covid19tracker.in/. Data from 1st December 2021 to 10th February 2022 were used to train the models. Daily COVID-19 cases in UK were extracted from https://covid19.who.int/info and data from 1st May 2021 to 10th February 2022 were used to train the models. The model was validated with the data from 11th February to 25th February 2022. The analysis was done using python in Google colab platform. The codes and data are available in https://drive.google.com/drive/folders/16G92yYDemYELBF7fNVvUjhRdxZKGz5y8?usp=sharing (Figure 3)

Results

India:

The modelling of India Covid19 considered the cases reported by the Government from 1st December 2021 to 10th February 2022. Figure 4 represents the observed and projected COVID-19 cases in India by using simple LSTM and CNN-LSTM models. In the beginning of December the cases were started to increase and reached the peak of 3.4 lakh cases on 20th January. Then it was decreased steadily and reached about 58,000 cases on 10th February 2022. The CNN-LSTM model shows that the cases will be decreasing steadily and reach about 17,000 cases on 25th February 2022. The reported case in same day was about 11,000. In the same manner, simple LSTM model shows the decreasing trend but the figure is slightly higher than CNN LSTM model which is about 30000 cases on 25th February 2022. The fits statistics were presented in Table 1. And table 2 presents the

observed and projected cases of validation data for both India and UK. The R square values were 0.95 and 0.97 for CNN and simple LSTM model respectively. The RMSE and MAE of CNN LSTM model were 24440.04and 15027.58 respectively. And it was 19252.12and 13237.22 for simple LSTM model respectively. On the other hand for the test data, MAE was 8993.33 and 18870.87 for CNN and simple LSTM model and MSE was 9972.81 and 19285.57 respectively. The simple LSTM model fits very well as compare to CNN LSTM model. However, the performance of CNN LSTM model was quite well in the test data.

The United Kingdom (UK):

Figure 5 represents the observed and projected COVID-19 cases in UK. The pattern of UK is differed from India. In the beginning of May 2021, the cases were increasing slowly and remaining almost same from August 2021 till December 2021. Then it was increased rapidly and reached the peak of 2.7 lakh cases on 06th January 2022. And then the cases were decreased rapidly. In the beginning of February 2022, UK experienced about an average of 77,000 cases every day. Simple LSTM model shows that the cases will be decreasing slowly and reach about 40,000 cases on 25thFebruary 2022. CNN-LSTM model shows that the cases will be decreasing little faster than Simple LSTM model and reach about 36,000 cases on 25th February 2022 which is close to the reported cases in the same day. The R square value of CNN and simple LSTM models was 0.77 and 0.84 respectively. The RMSE was 21287.39for CNN LSTM and it was17830.48for simple LSTM. The MAE for CNN LSTM was 12163.53 and it was 9722.01 for simple LSTM model. For the test data, MAE for CNN and simple LSTM model was 10060.92 and 7824.82 respectively. And the MSE value was 12111.95 and 8935.75 for CNN and simple LSTM model respectively.

Discussion

Short- and long-term model forecasts of the number of cases in any epidemic or pandemic can be a useful tool for several government agencies and policymakers in different countries to prepare for the progression and spread of a disease. Given the rapid growth of COVID-19 cases worldwide, it is imperative that these tools be widely available to relevant agencies and personnel to estimate number of hospital beds and ventilators that are needed. Also, to estimate the requirement and availability of healthcare workers, and the demand for personal protective equipment (PPE). These would in turn be directly reflected in better outcomes for patients as well as the number of lives saved due to a deadly or fatal disease (22).

Many researchers incorporated LSTM models in their work and compared it with other models. Barman (2020) forecasted cumulative COVID-19 cases using LSTM model and compared it with ARIMA (22). They found that errors in prediction for LSTM models were slightly less than ARIMA models. Moreover, Shyam et al. (2020) analyzed the performance of the LSTM network and compare it with two parameter reduced variants of LSTM model (23). Chimmula et al.(2020) evaluated the key features to predict the trends and possible stopping time of the current COVID-19 outbreak in Canada and around the world in 2020 by using long short-term memory (LSTM) networks and forecast the COVID-19 cases (24). Hawas et al. (2020) predicted the daily infection cases in Brazil using LSTM and GRU (Gated Recurrent Unit) model (25). Kafieh et al. (2021) applied Multilayer perceptron, random forest, and various LSTM models for nine countries and forecast the epidemic (26). However, all the articles were published in the technical journals not in medical or epidemiology journals. Moreover, none of them has explained the steps involved in the LSTM models in a simple way. Therefore, we

did not compare LSTM with any other models as many studies have indicated that it is better than other classical time series models. Besides that, we aimed to train young researchers who are interested in this field but have a little capacity in using this method. Therefore, this manuscript provides the details of using LSTM model using Python codes. The codes are made available for the readers in such a way that they can understand the hyper parameters and the steps easily.

One of the important steps in LSTM models is to fix dense layers and hyper parameters such as number of lag and epochs. There is no clear guidance to choose lag. Some researchers training the models with various time step values and choose the optimal one which gives the better performance. However, we do not know how many time points that the future data depends on exactly. Therefore, based on our experience, we choose the time step (lag) by plotting ACF / PACF plot. However, the pattern of the epidemic changes very frequently, we recommend to revise the model frequently may be weekly once or once in a couple of weeks. Invariably, we used four to eight dense layers in our model. The number of dense layers will be adjusted based on model performance. Except the output layer, we used the “he_uniform” distribution to initialize the weights of the layers and also adopt L2 kernel regularization to avoid over fitting. Rectified Linear Unit (ReLU) activation function was used in both LSTM and all the dense layers. Moreover, Adam optimizer technique was used to update the weights. We have also used call backs to decrease the learning rate when there is no improvement in the past epochs.

We have used India and UK data for the modelling. However, UK data shows that oscillating pattern and after reaching the peak, a sharp decrease in the counts. In both of the cases LSTM models work well with R^2 value was about 0.96 for India and about 0.80 for UK. This statistic suggested that the model performs well at various trend of the epidemic curves.

One of the concern is that, the strategies and polices of the government may change based on the virus effectiveness. The sudden changes in the testing policy may mislead the prediction. Therefore the model may over or under estimate the cases.

Limitations

Out of several possible LSTM models, only two were evaluated in this study due to space (word count) constraint. The time series analysis and forecasting presented in this study are restricted to short term forecasts only. The users of this algorithm should have minimal knowledge of time series analyses and should be able to estimate the Auto Regression parameter (p) and Moving Average parameter (q) using autocorrelation function and partial auto-correlation function. The model needs enough data to learn the pattern of the epidemic. We have done all the projections using python code and, therefore the users of this method need to know the basics of python codes.

Conclusion

Simple LSTM model works better than CNN-LSTM model while training. Nonetheless, the performance of CNN is better in the validation. Therefore our suggestion for the researcher is, train various LSTM models instead of sticking only one and forecast future, report the range of values. Moreover, we suggested to revise the model

frequently may by weekly once or once in couple of weeks. Because, the behaviour of an epidemic may change a over time.

Abbreviations

COVID-19: Corona virus Disease 2019; ARIMA– Auto Regressive Integrated Moving Average; ANN: Artificial Neural Networks; RNN: Recurrent Neural Network; LSTM – Long Short Term Memory; CNN – Convolutional Neural Network; SARSCOV-2: Severe Acute Respiratory Syndrome Corona virus 2; SEIR: susceptible-exposed-infectious-recovered; SIR: susceptible-infectious-recovered; WHO: World Health Organization; RMSE: Root Mean Square Error; MAE: Mean Absolute Error;

Declarations

Acknowledgements

No acknowledgement

Authors' contributions

SM – Marimuthu S

TM – Thenmozhi Mani

MJ – Melvin Joy

LJ – Jeyaseelan L

Funding

No grant has been received for this project.

Availability of data and materials

Data analysed during the current study are available in https://drive.google.com/drive/folders/16G92yYDemYELBF7fNVvUjhRdxZKGz5y8?usp=sharing

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The Authors declare that there is no conflict of interest.

References

1. Moran KR, Fairchild G, Generous N, Hickmann K, Osthus D, Priedhorsky R, et al. Epidemic Forecasting is Messier Than Weather Forecasting: The Role of Human Behavior and Internet Data Streams in Epidemic Forecast. The Journal of Infectious Diseases. 2016 Dec 1;214(suppl_4):S404–8.

2. World Health Organization. Managing epidemics: key facts about major deadly diseases [Internet]. Geneva: World Health Organization; 2018 [cited 2022 Jan 31]. 257 p. Available from: https://apps.who.int/iris/handle/10665/272442

3. Roda WC, Varughese MB, Han D, Li MY. Why is it difficult to accurately predict the COVID-19 epidemic? Infect Dis Model. 2020;5:271–81.

4. Arti MK. Modeling and Predictions for COVID 19 Spread in India. 2020 [cited 2022 Jan 31]; Available from: http://rgdoi.net/10.13140/RG.2.2.11427.81444

5. Singh R, Adhikari R. Age-structured impact of social distancing on the COVID-19 epidemic in India. arXiv:200312055 [cond-mat, q-bio] [Internet]. 2020 Mar 26 [cited 2022 Jan 31]; Available from: http://arxiv.org/abs/2003.12055

6. Gupta R, Pandey G, Chaudhary P, Pal SK. SEIR and Regression Model based COVID-19 outbreak predictions in India. medRxiv. 2020;

7. Ivorra B, Ruiz Ferrández M, Vela M, Ramos AM. Mathematical modeling of the spread of the coronavirus disease 2019 (COVID-19) taking into account the undetected infections. The case of China. Communications in Nonlinear Science and Numerical Simulation Accepted (In Press). 2020;

8. Mandal S, Bhatnagar T, Arinaminpathy N, Agarwal A, Chowdhury A, Murhekar M, et al. Prudent public health intervention strategies to control the coronavirus disease 2019 transmission in India: A mathematical model-based approach. The Indian Journal of Medical Research. 2020 Mar;151(2–3):190.

9. Malavika B, Marimuthu S, Joy M, Nadaraj A, Asirvatham ES, Jeyaseelan L. Forecasting COVID-19 epidemic in India and high incidence states using SIR and logistic growth models. Clinical Epidemiology and Global Health. 2021 Jan 1;9:26–33.

10. Rai B, Shukla A, Dwivedi L. COVID-19 in India: Predictions, Reproduction Number and Public Health Preparedness. medRxiv. 2020;

11. Kırbaş İ, Sözen A, Tuncer AD, Kazancıoğlu FŞ. Comparative analysis and forecasting of COVID-19 cases in various European countries with ARIMA, NARNN and LSTM approaches. Chaos, Solitons, and Fractals. 2020 Sep;138:110015.

12. Rguibi MA, Moussa N, Madani A, Aaroud A, Zine-Dine K. Forecasting Covid-19 Transmission with ARIMA and LSTM Techniques in Morocco. SN Comput Sci. 2022;3(2):133.

13. Siami-Namini S, Tavakoli N, Siami Namin A. A Comparison of ARIMA and LSTM in Forecasting Time Series. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). 2018. p.

1394–401.

14. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks. 1994 Mar;5(2):157–66.

15. Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997 Nov 15;9(8):1735–80.

16. Rumelhart DE, Hinton GE, Williams RJ. Learning Internal Representations by Error Propagation. :23.

17. kavita D, Saxena A, Joshi J. Using of Recurrent Neural Networks (RNN) Process. 2017;4(2).

18. Abbasimehr H, Shabani M, Yousefi M. An optimized model using LSTM network for demand forecasting. Computers & Industrial Engineering. 2020 May 1;143:106435.

19. Wang H, Song Y, Tang S. LSTM-based Flow Prediction. arXiv: Learning [Internet]. 2019 Aug 9 [cited 2022 Jan 31]; Available from: https://www.scinapse.io

20. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE [Internet]. 1998 Jan 1 [cited 2022 Jan 31]; Available from: https://www.scinapse.io

21. Lu W, Li J, Li Y, Sun A, Wang J. A CNN-LSTM-Based Model to Forecast Stock Prices. Complexity. 2020 Nov 24;2020:e6622927.

22. Barman A. Time Series Analysis and Forecasting of COVID-19 Cases Using LSTM and ARIMA Models. arXiv:200613852 [cs, stat] [Internet]. 2020 Jun 5 [cited 2022 Jan 31]; Available from: http://arxiv.org/abs/2006.13852

23. Shyam Sunder Reddy K, Padmanabha Reddy YCA, Mallikarjuna Rao C. Recurrent neural network based prediction of number of COVID-19 cases in India. Mater Today Proc. 2020 Nov 17;

24. Chimmula VKR, Zhang L. Time series forecasting of COVID-19 transmission in Canada using LSTM networks. Chaos, Solitons & Fractals. 2020 Jun 1;135:109864.

25. Hawas M. Generated time-series prediction data of COVID-19's daily infections in Brazil by using recurrent neural networks. Data in Brief. 2020 Oct 1;32:106175.

26. Kafieh R, Arian R, Saeedizadeh N, Amini Z, Serej ND, Minaee S, et al. COVID-19 in Iran: Forecasting Pandemic Using Deep Learning. Computational and Mathematical Methods in Medicine. 2021 Feb 26;2021:e6927985.

Tables

Table1: Fits Statistics for India and UK

	India				The UK			
	Train data		Validation data		Train data		Validation data	
	CNN LSTM	Simple LSTM	CNN LSTM	Simple LSTM	CNN LSTM	Simple LSTM	CNN LSTM	Simple LSTM
R^2	0.95	0.97	-	-	0.77	0.84	-	-
MAE	15027.58	13237.22	8993.33	18870.87	12163.53	9722.01	10060.9	7824.821
RMSE	24440.04	19252.12	9972.81	19285.57	21287.39	17830.48	12111.95	8935.75

Note: CNN- Convolutional Neural Network; LSTM – Long Short Term Memory; RMSE - Root Mean Squared Error; MAE - Mean Absolute Error

Table2: COVID-19 projected cases for India and UK

	India			The United Kingdom		
Date	Validation data	CNN LSTM	Simple LSTM	Validation data	CNN LSTM	Simple LSTM
11-02-2022	50399	62620	60087	66638	80751	71995
12-02-2022	45129	50337	57063	58316	51152	72275
13-02-2022	34493	52038	54483	45500	44257	53853
14-02-2022	26052	43681	51869	40844	57774	48056
15-02-2022	30987	42871	49357	41170	53938	29048
16-02-2022	30907	34424	46965	54665	65601	50484
17-02-2022	26274	34286	44696	52453	76969	37778
18-02-2022	22042	30296	42544	51520	48643	55745
19-02-2022	19753	30031	40492	44670	40035	46578
20-02-2022	16678	23535	38539	33217	39333	41775
21-02-2022	12207	23244	36679	34642	47027	45896
22-02-2022	15479	19639	34910	37667	43537	41407
23-02-2022	14484	22669	33226	45089	24988	30627
24-02-2022	13083	17453	31623	39115	50025	42111
25-02-2022	11601	17344	30098	36442	36793	40812

Note: CNN- Convolutional Neural Network; LSTM – Long Short Term Memory;

Figures

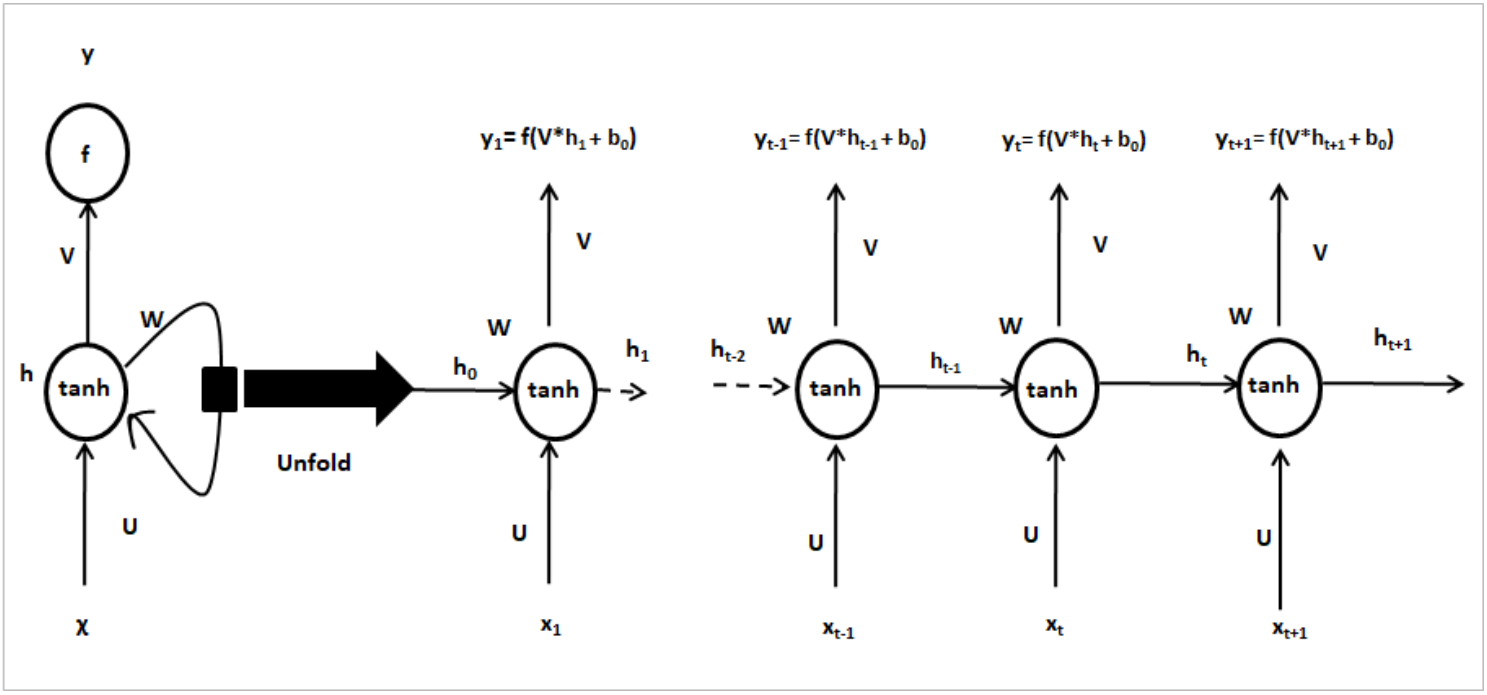
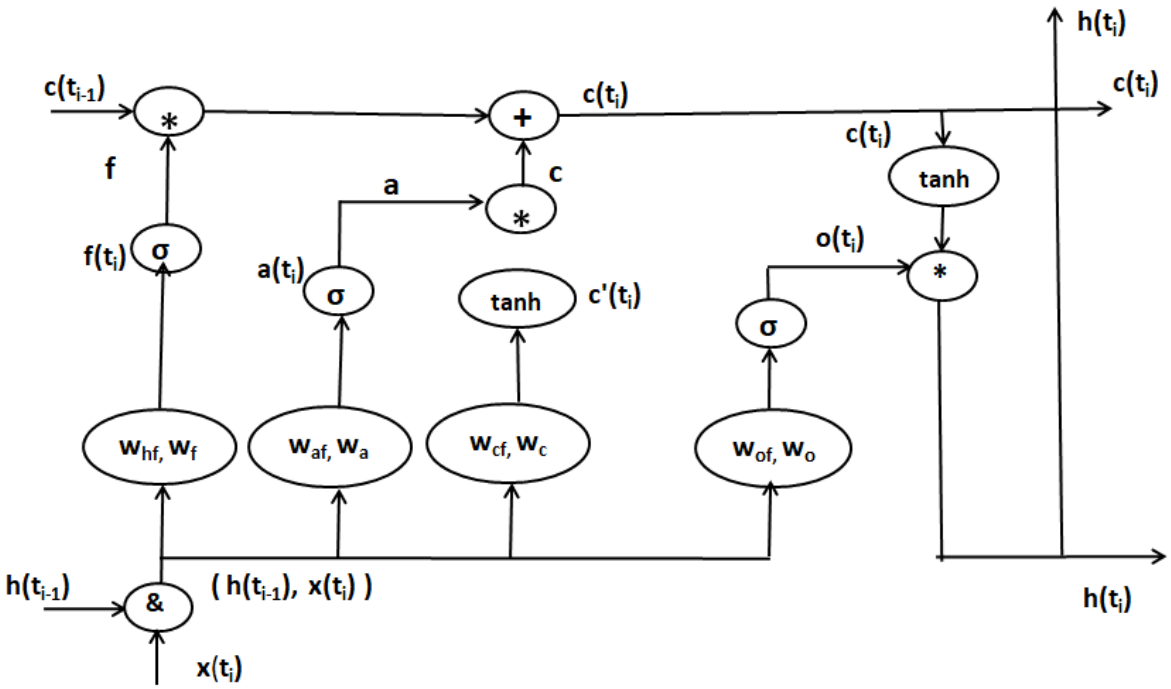


Figure 1

Simple RNN Architecture



Notations:

$\boldsymbol{x(t_i)}$**:** The input value; $\boldsymbol{h(t_i)}$: The current state output value at time point t_i; $\boldsymbol{c(t_i)}$ **:** Cell state value at time points t_i;

$\boldsymbol{b} = \{\boldsymbol{b_a}, \boldsymbol{b_f}, \boldsymbol{b_c}, \boldsymbol{b_0}\}$ are biases of input gate, forget gate, internal state and output gate.

$\vec{W}_1 = \{\boldsymbol{w_a}, \boldsymbol{w_f}, \boldsymbol{w_c}, \boldsymbol{w_o}\}$ are weight matrices of input gate, forget gate, internal state and output gate and output.

$\vec{W}_2 = \{\boldsymbol{w_{ha}}, \boldsymbol{w_{hf}}, \boldsymbol{w_{hc}}, \boldsymbol{w_{ho}}\}$ are the recurrent weights ;

$\vec{\boldsymbol{a}} = \{\boldsymbol{a(t_i)}, \boldsymbol{f(t_i)}, \boldsymbol{c(t_i)}, \boldsymbol{o(t_i)}\}$ are the results for input gate, forget gate, internal state, and output gate.

σ and **tanh** are activation functions. **&,** +, and * represent the Concatenation, Addition, and point-wise multiplication respectively.

Figure 2

The architecture of LSTM memory cell (18)

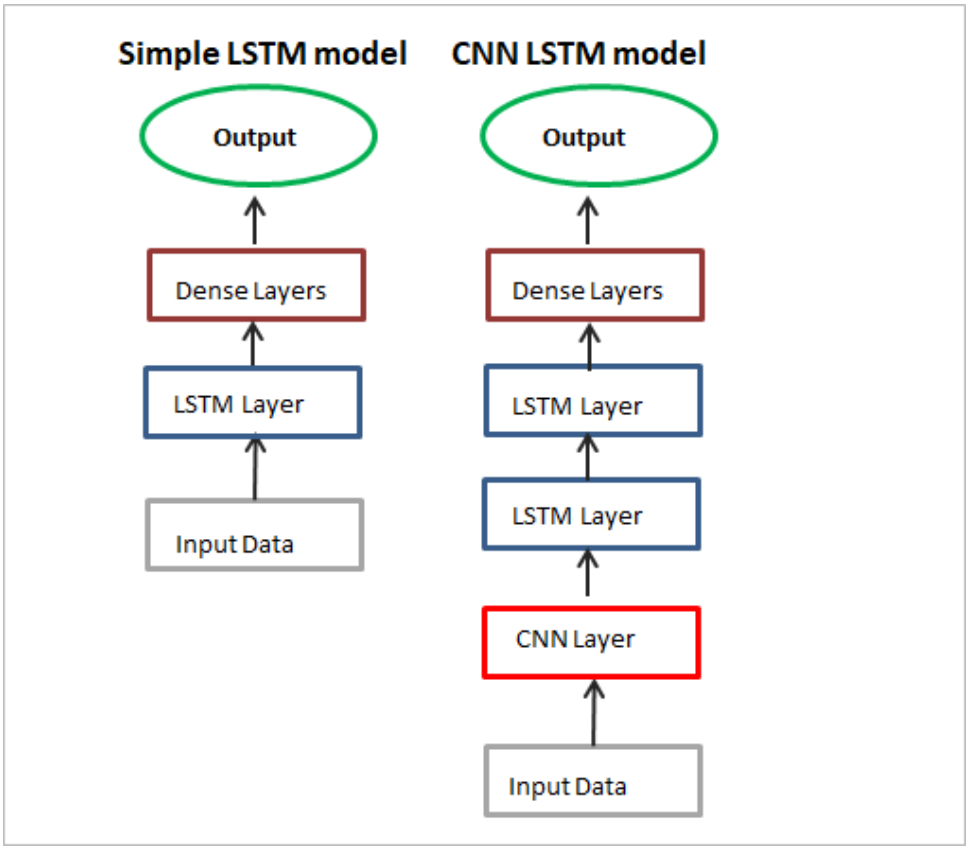


Figure 3

Structure of LSTM Models

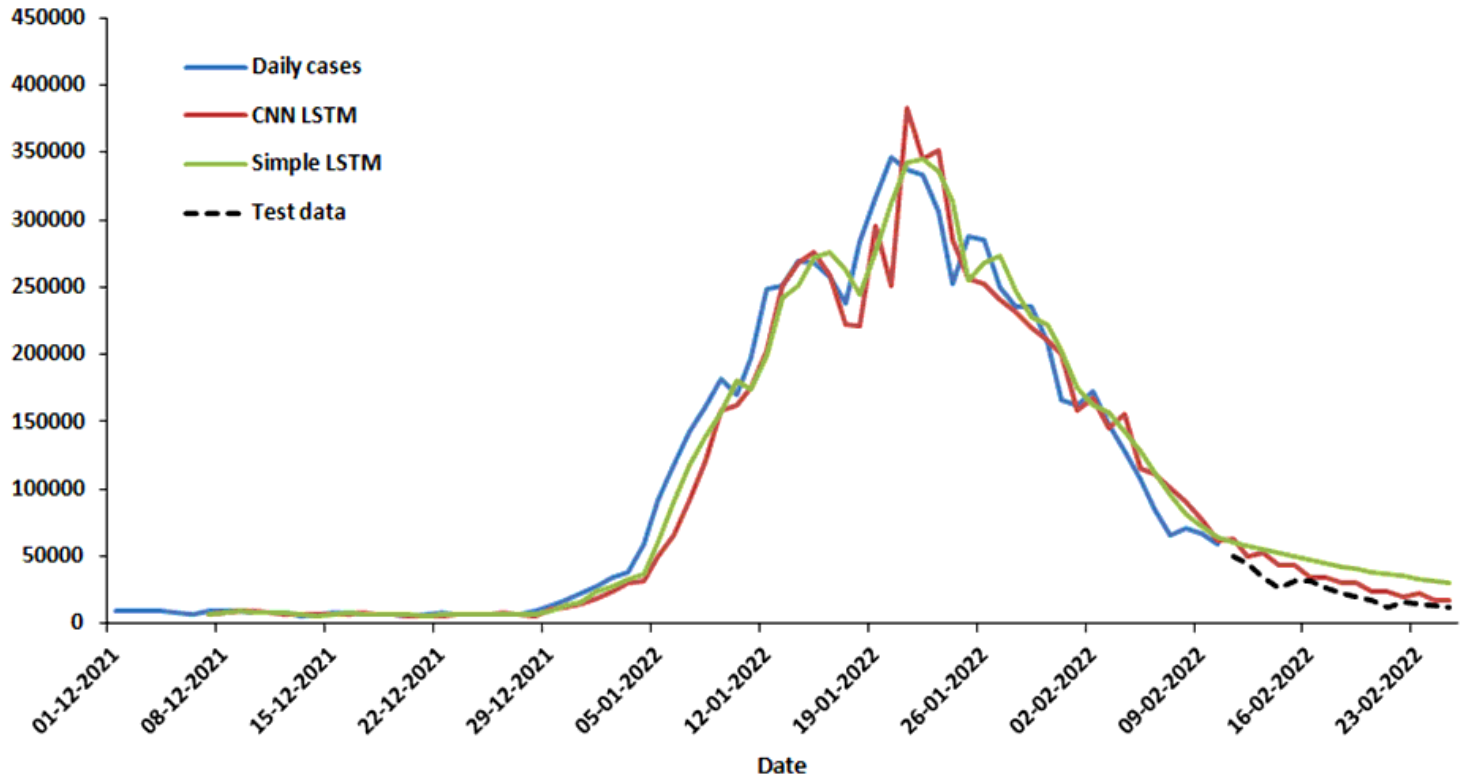


Figure 4

COVID-19 Projection for India

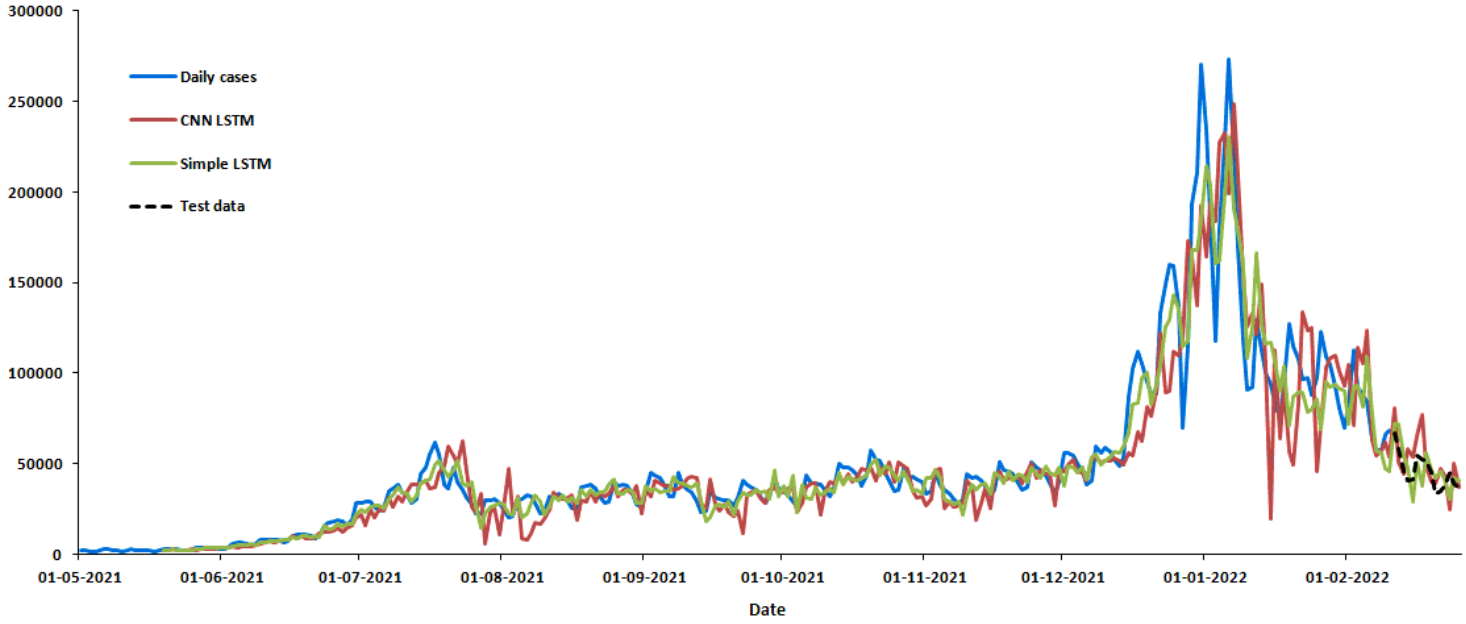


Figure 5

COVD-19 Projection for the United Kingdom