Letters in Applied NanoBioScience
Platinum Open Access Journal (ISSN: 2284-6808)

Article
Volume 10, Issue 2, 2021, 2044 - 2055
https://doi.org/10.33263/LIANBS102.20442055

Forecasting COVID-19 Transmission in India Using Deep Learning Models

Ashutosh Anand [1], Yash Lamba [2], Arpita Roy [3,*]

[1] Delhi Technological University, Delhi, India
[2] Cluster Innovation Center, Delhi University, Delhi, India
[3] Department of Biotechnology, School of Engineering & Technology, Sharda University, Greater Noida, India
* Correspondence: arbt2014@gmail.com;

Scopus Author ID 57203962973

Received: 23.09.2020; Revised: 14.10.2020; Accepted: 15.10.2020; Published: 17.10.2020

Abstract: In the past 6 months, the world has come to a standstill due to an escalation in the number of cases of COVID-19. COVID-19 is an infectious disease that was formerly called as 2019-nCoV or the novel Coronavirus 2019. COVID-19 first originated in Wuhan, China, in late December 2019, and subsequently, the World Health Organization declared it as a pandemic on 11th March, 2020. Lack of preparedness for the COVID-19 pandemic has put colossal stress on the healthcare systems of the world's largest economies. In a short period, the disease has spread to an unexpected number of people due to its high transmission rate and doesn't show a sign of slowing down in the near future. Estimating the rising number of cases via predictive modeling can help gauge the quantity of various medical amenities required for the treatment of patients as well as protective apparatus for essential workers and susceptible populations. In this paper, we have performed time series forecasting on the publicly available COVID-19 datasets of India using RNNs with LSTM and GRU. Additionally, we also employ the final models for analyzing similar data from different countries.

Keywords: COVID-19; deep learning; ANN; RNN; GRU; LSTM; time series analysis; forecasting.

1. Introduction

Starting in December 2019, a new coronavirus, namely 2019-nCoV, which was later changed to SARS-CoV-2, was identified, causing flu and pneumonia-like symptoms in people of Wuhan, China [1]. The novel coronavirus disease of COVID-19 was declared a pandemic on 11th March, 2020 by the World Health Organization. As of 24th September, 2020, 31,870,904 confirmed cases all over the globe, with 976,311 confirmed deaths of COVID-19 (WHO) [2,3]. Infectious diseases like COVID-19 tend to follow certain patterns, and these patterns can be used to predict and prepare for the future stages of the disease[4]. The major problem that brings the light to this pandemic is how fragile the medical systems of the world are at present [5]. Maintaining the availability of medical equipment, along with having enough staff and infrastructure ready to handle increasing cases, is crucial to prevent deaths. Predicting this increase can help to get an idea of how much preparation is required and taking timely steps, along with measuring the impact of the outbreak [6,7].

In India, the first case of COVID-19 was reported on 30th January 2020. Within five days, there were 3 confirmed cases of COVID-19. However, the number of cases in the month of February was fairly stagnant. The confirmed cases started escalating during the month of

March. On 24th March 2020, the Prime Minister of India instated the first phase of nationwide lockdown, followed by three more such phases [8]. To control an epidemic on such a scale, countries like India, with a population of 1.35 billion, need a clear idea of the challenges they are going to face. Forecasting can assist in the formation of decisive planning to quantify and allocate the important resources to tackle this disease effectively. The aim of this paper is to forecast the situation implementing Recurrent Neural Networks (RNNs) using Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU).

Traditional data-driven approaches use linear methods for analysis [9]. These methods often neglect the temporary state/components in the data. The linear functions used in regression with these methods can't effectively map the time series of an infectious disease. Popular statistical models like Auto-Regressive Integrated Moving Average (ARIMA), Moving Average (MA), Auto-Regressive (AR) are also difficult to use for real-time forecasting and depend on multiple assumptions [10]. Deep learning can overcome all this problem as it uses multiple processing layers to extract higher-level feature data, and RNNs can help map the temporal components of the data for forecasting.

Deep learning is a subclass of machine learning models that involves deep artificial neural networks, i.e., artificial neural networks with multiple hidden layers. Deep learning models can learn a representation of data and can progressively extract higher-level feature data using multiple processing layers [11,12]. In this paper, a subtype of deep neural networks called RNNs was used. RNNs are used majorly in tasks involving sequential input data like speech and language. RNNs process sequences of data and simultaneously maintain information about the history of sequences through hidden state vectors. This particular feature of RNNs can help in time series forecasting and understand the underlying patterns in the data [4].

Time series data is a sequence of data points taken at equally spaced "points in time" or discrete-time. Time series analysis includes a method to analyze data in order to extract meaningful observations from previously seen data for future forecasting or understanding the underlying patterns. Stock price forecasting, weather predictions, econometrics, etc. are some applications of time series analysis and forecasting [13,14]. Since the outset of COVID-19, various government and private institutions are recording daily statistics of the disease to measure its impact, strategize, and take precautionary measures accordingly.

A time series could be broken into two main classes: stationary and non-stationary. A time series is considered to be stationary when: (1) there doesn't exist any trend, (2) the mean and variance remain constant over a period of time, and (3) there is no autocorrelation. As opposed to a stationary time series, non-stationary time series exhibits a trend, and its statistical parameters, such as mean and variance, change with time. Augmented Dickey-Fuller (ADF) test is carried out to establish the nature of the input time series. If the p-value comes between 0.01 (or 1%) and 0.05 (or 5%) then the series is considered to be stationary otherwise if p-value is greater than 0.05 (or 5%) then it is considered nonstationary [13].

Before introducing the various architectures, it is imperative to learn about time series classification (TSC). A univariate time series can be defined as the ordered set of real values such that $X = \{ x_1, x_2, x_3, \dots , x_t \}$. A multivariate time series can be defined as an ordered set of real values having M-dimensions such as $X = [X^1, X^2, \dots, X^T]$ consisting of M different univariate series having $Xi \in R^T$. A model is tasked to map the inputs from a dataset D to the possible outputs.

Deep learning frameworks are designed to capture the underlying temporal pattern from the data. Neural networks are composed of layers where the output from the previous layer acts as the input for the current layer. A transformation or an activation function is applied in a layer to calculate its output. Apart from the transformation function, weights within the neural network layer associate the input from the previous layer to the output of the current layer. Hence for a fully connected layer l_i, the output can be written as:

$$f_i(x) = f_i(W_{i-1}^T X_{i-1}) + b_i \quad (1)$$

Where,

W^T is the weight matrix of the previous layer

X is the output matrix of the previous layer

b is the bias in the current layer

While training, a model is given a set of inputs and their corresponding outputs. At first, weights are initialized randomly, followed by a forward pass through the network, and the output is calculated via the activation function, *f* on the given input. Error in the predictions by the model is computed according to a function known as cost function, such as mean squared error. The error is then propagated backward through the network in order to update the weights accordingly using a gradient descent algorithm. Hence, by iterative forward and backward propagation through the network, a cost function is minimized on the data in which the network is being trained.

Previously unseen values are passed to the network during the inference phase, followed by the predictions by the network. The performance of the network can be evaluated using a given metric. There are various types of neural networks such as RNN, RNN with LSTM, and RNN with GRU [15].

RNNs are well suited for time series data because they incorporate sequential data into the training and processing of input data. RNNs have loops in their architecture, which allows the information to persist in being used by subsequent modules. Figure 1 shows the unrolled representation of RNNs. In Figure 1, X_t is the input at time step t, y_t is the output at time step t, and H is the hidden state. RNNs have been used in various problems dealing with time-series data such as speech recognition, air pollution, etc. During the learning phase, backpropagation of errors is carried out by an algorithm known as Back Propagation Through Time (BPTT). In theory, RNNs can be used to understand long-term dependencies in temporal data, but in practice, it is not often observed. This is because BPTT, in long term dependencies, is vulnerable to either vanishing of gradient or exploding of the gradient. Due to the high number of derivative passes to propagate error backward, the gradient becomes so small that the model is unable to capture the underlying pattern in the data [16]. LSTM, a special type of RNN, solved the shortcomings of traditional RNNs [13]. LSTMs, introduced in 1997 by Hochreiter & Schmidhuber [17], is a special type of RNNs developed to overcome the problem of vanishing and exploding gradient in traditional RNNs. They solved the problem of error propagation by using gates, which enables them to keep or forget information based on whether it is being used or not. This maintains enough error to allow significant updates in the gradient. In an LSTM cell, data is passed through various layers and undergo pointwise operations, as shown in Figure 2 [13].

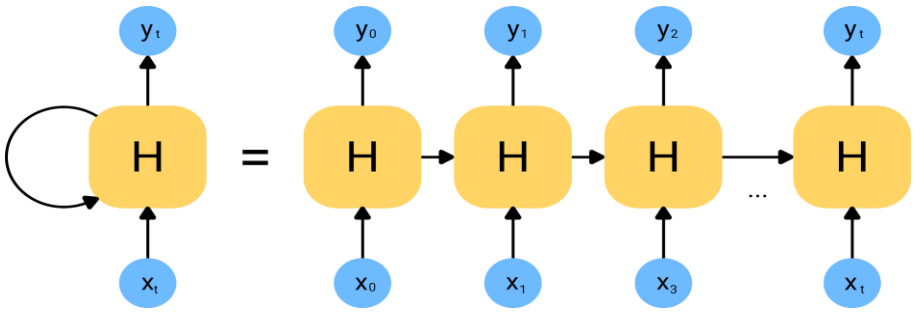

Figure 1. Unrolled structure of recurrent neural network.

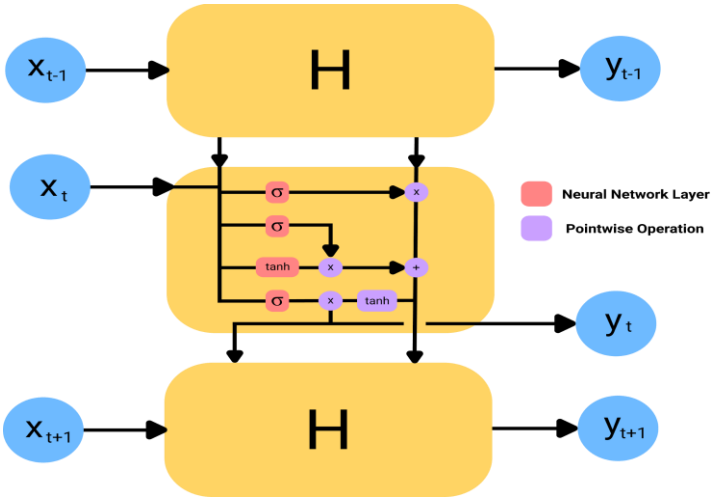

Figure 2. Chain structure of LSTM.

An LSTM cell consists of three gates, i.e., input gate, output gate, and forgets gate. Forget gate decides whether information should be passed or neglected. It takes into account the input at time step t, X_t, and output of the timestep t-1, y_{t-1}, and produces an input between 0 and 1 for each element in the cell state. This can be represented by:

$$f_t = \sigma\left(W_f[y_{t-1}, x_t] + b_f\right) \quad (2)$$

Then the input gate decides which information is to be updated in the cell state, which can be represented by the following equation:

$$i_t = \sigma(W_i[y_{t-1}, x_t] + b_i) \quad (3)$$

The output gate combines the output from the cell state and output of the forget gate at time t to generate the output. Similarly, the output gate can be represented by the following equation:

$$o_t = \sigma(W_i[y_{t-1}, x_t] + b_o) \quad (4)$$

This structure of LSTM enables the sharing of weights for longer durations, which prevents the vanishing of the gradient.

GRU was designed to make each unit to capture time dependencies of different scales. Unlike LSTM, GRU does not have a dedicated memory cell bringing down the number of gates from three in LSTM to two in GRU. The update gate, z_t decides which information to be updated in the current layer. For a layer l_i, it can be written as:

$$z_t = \sigma(W_{zi}x_t + U_{zi}h_{t-1}) \quad (5)$$

From eq (5), it can be observed that the linear combination of recently computed state and existing state in GRU is similar to that in LSTM.

Similar to update gate, reset gate, $r_t$for a layer $l_i$can be written as:

$$r_t = \sigma(W_{ri} + U_{ri}h_{t-1}) \quad (6)$$

It has been noted in multiple studies like [18], and references therein show that the performance of GRU is comparable to LSTM, in some cases even surpassing them.

In this paper, we compare multiple deep learning architectures that are generally used for time series analysis and forecasting for COVID-19 predictions in India. Forecast models can help prepare beforehand for the future phases of the outbreak, which would help people effectively combat this epidemic. This would also help in the quantification and allocation of important resources.

2. Materials and Methods

2.1. Data source and features.

For the time series analysis and forecasting of COVID-19 in India, the data was collected from COVID-India API and Our World in Data. As of 25th July 2020, India has over 1.3 million confirmed cases of COVID-19, and with over 31 thousand deaths. India has the sixth-highest COVID-19 related deaths in the world. However, with over 849 thousand recoveries, India also has the third-highest number of recovered cases. The final data has about 175 data points from 30th January to 21st July 2020, and 9 features, out of which 6 are used in this paper. The final used features/columns are total cases, active cases, daily deaths, total deaths, totally recovered, daily recovered. Also, the dataset does not have any missing values for the mentioned time period. However, for training, testing, and error calculation, we removed the data of the first 45 days as there were zeros in some rows. Thus, we were left with 130 data points to train and test the model. These data points were further split, where we used 85% of the data to train the models. However, the models were tested on both the testing and training data points, and the average percentage error was calculated. The data had to be converted in the right shape to be fed into GRU and LSTM models. Therefore, the features were converted into a 3-dimensional matrix of shape (no. of samples, timestep, no. of features). A timestep of 3 was considered in this paper. This 3-dimensional matrix was used for training the models and testing them.

2.2. Feature selection and data preparation.

For time series analysis and forecasting, the following features were selected:

1. Total Cases/Total Confirmed Cases
2. Total Deaths
3. Active Cases

The selected features can be considered as a univariate time series. ADF Test was performed on the time series selected to determine whether they were stationary or non-stationary.

We further performed multivariate time series analysis to understand the effect of multiple features on a given label. The various features and labels considered are shown in Table 1.

Table 1. Features and labels took for multivariate time series analysis.

S.No	Features	Label
1.	Total Cases, Total Deaths, Total Recovered	Total Cases
2.	Active Cases, Daily Deaths, Daily Recovered	Active Cases

2.3. Neural network architectures.

In this paper, we have three architectures that are:

1. Vanilla LSTM: This model consists of 3 layers with 32, 32, 1 neuron(s), respectively. The first layer is made up of LSTM cells with ReLU activation; the second is a simple dense layer with ReLU activation followed by an output layer.
2. Stacked LSTM: This model consists of 4 layers with 32, 32, 32, 1 neuron(s), respectively. The first and second layers are made up of LSTM cells with ReLU activation; the third is a simple dense layer with ReLU activation followed by an output layer.
3. GRU: This model consists of 2 layers with 64, 1 neuron(s), respectively. The first layer is made up of GRU cells with ReLU activation; the second is a simple dense output layer.

The networks were created using Tensorflow.

3. Results and Discussion

The first case of COVID-19 in India was reported on 30th January 2020. However, during February 2020, the total number of confirmed cases reported was 3 and stayed the same throughout the month. It was in the month of March 2020 when the disease started to spread throughout the country. The first death due to COVID-19 was reported on 12th March 2020. Even though the disease was spreading at a high rate, the death toll was low. On 24th March 2020, The Government of India enforced a complete nationwide lockdown till 14th April as a preventive measure against COVID-19. The first lockdown was announced when the total confirmed cases were 571, and the total death toll was 10. The lockdown was extended thrice; first from 14th April to 3rd May, second from 3rd May to 17th May, and lastly from 17th May to 31st May. The data used for analysis and forecasting is taken from 13th March 2020 to 21st July 2020. We first established the nature of the univariate time series using the ADF Test. Then we trained three different deep learning models, and the subsequent results obtained are presented in the following sections.

In order to establish the nature of the time series considered in this paper, we perform ADF Test. To test the nature of the time series, we put forward the following hypothesis:

1. Null Hypothesis (H_0): - time series is non-stationary
2. Alternate Hypothesis (H_a): - time series is stationary

As mentioned previously, the p-value should be less than 0.05 (or 5%) to reject the Null Hypothesis. Table 2 shows the results of the ADF Test on the time series studied in this paper.

Table 2. Results of ADF for various time series.

S.No	Time Series	p-value
1.	Total Cases	0.995
2.	Total Deaths	1.0
3.	Active Cases	1.0

From Table 2, it can be observed that the p-value of all the time series selected for analysis and forecasting is greater than 0.05 (p-value>0.05). Hence, we fail to reject the Null Hypothesis. Thus, we conclude that the time series taken into consideration are non-stationary time series and show trends in their statistical parameters.

3.1. Univariate time series analysis and forecasting.

Table 3 shows the performance of three deep learning models trained on three different univariate time series. The mean percentage error is calculated on the training and testing data. We also compare the forecast made by the models to the observed values for five days from 22nd July 2020 to 26th July 2020, as shown in Table 3. The total number of confirmed cases

of COVID-19 obtained from different models can be seen in Figure 3. Similar observations have been made for total deaths and active cases, as seen in Figures 4 and 5, respectively. These results imply that the proposed models are capable of producing good results that could be used for estimating the supplies that would be required in the near future to combat COVID-19 effectively.

Table 3. Mean percentage error on training, testing, and forecast data using different RNNs on various time series.

S.No.	Model Description	Feature	Label	Mean %error on training data	Mean %error on testing data	Mean %error on forecast
1.	GRU	Total Cases	Total Cases	6.484%	0.194%	5.825%
2.	Vanilla LSTM	Total Cases	Total Cases	7.5%	0.684%	6.001%
3.	Stacked LSTM	Total Cases	Total Cases	9.016%	0.657%	5.894%
4.	GRU	Total Deaths	Total Deaths	9.002%	0.965%	4.343%
5.	Vanilla LSTM	Total Deaths	Total Deaths	5.205%	2.348%	5.661%
6.	Stacked LSTM	Total Deaths	Total Deaths	6.461%	0.435%	4.025%
7.	GRU	Active Cases	Active Cases	7.171%	0.895%	5.009%
8.	Vanilla LSTM	Active Cases	Active Cases	11.013%	3.215%	6.275%
9.	Stacked LSTM	Active Cases	Active Cases	8.402%	1.798%	6.045%

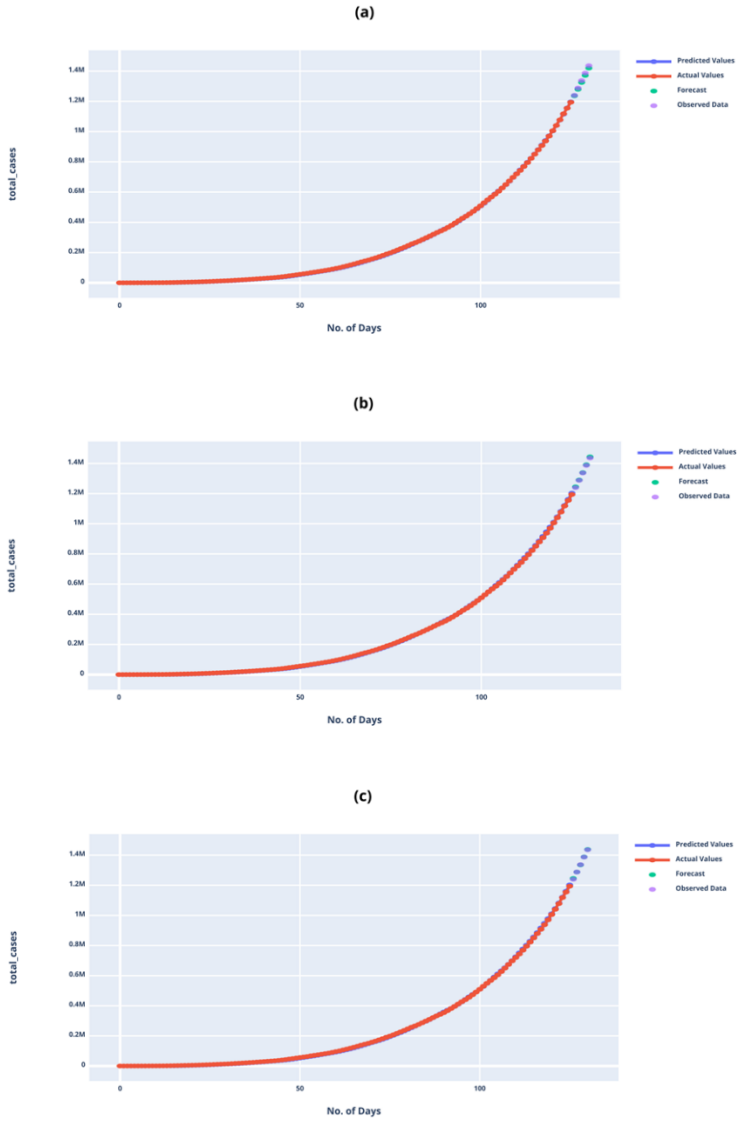

Figure 3. Total number of confirmed cases prediction using a) GRU b) Vanilla LSTM c) Stacked LSTM.

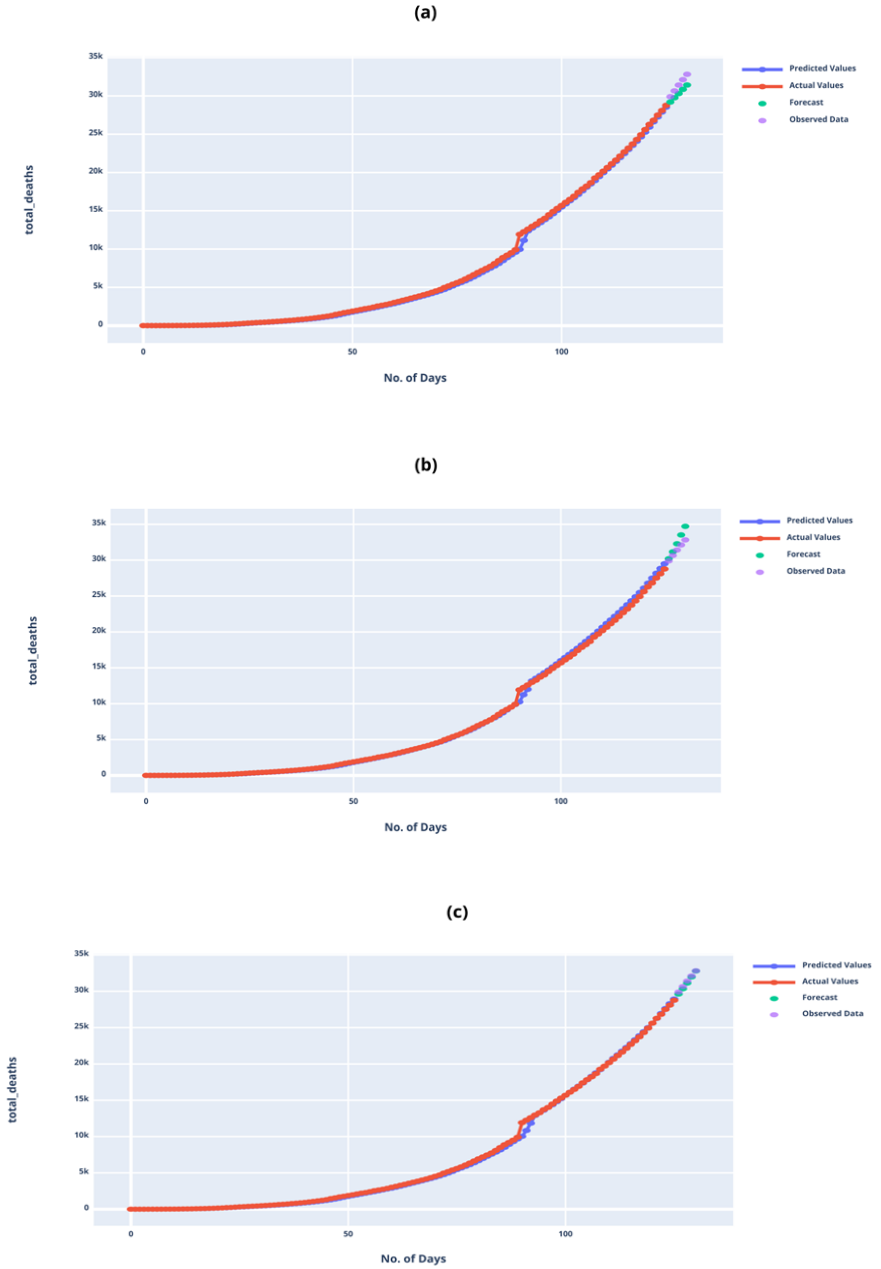

Figure 4. Total deaths prediction using a) GRU b) Vanilla LSTM c) Stacked LSTM.

Table 4. Mean percentage error on training and testing using various models on multivariate time series.

S.No	Model Description	Features	Label	%error on training data	%error on testing data
1.	GRU	Total Cases, Total Deaths, Total Recovered	Total Cases	15.334%	0.505%
2.	Vanilla LSTM	Total Cases, Total Deaths, Total Recovered	Total Cases	20.245%	4.975%
3.	Stacked LSTM	Total Cases, Total Deaths, Total Recovered	Total Cases	6.272%	0.708%
4.	GRU	Active Cases, Daily Deaths, Daily Recovered	Active Cases	8.707%	2.119%
5.	Vanilla LSTM	Active Cases, Daily Deaths, Daily Recovered	Active Cases	10.44%	2.914%
6.	Stacked LSTM	Active Cases, Daily Deaths, Daily Recovered	Active Cases	9.508%	10.18%

3.2. Multivariate time series analysis.

In this section, we present the evaluation of the proposed models on multivariate time series (as mentioned in Table 1) by analyzing the effect of various features on a given label. Table 4. presents the performance of proposed models on two multivariate time series by showing their mean percentage errors on the training and testing data. The resulting graphs for total confirmed cases and active cases are shown in Figure 6 and Figure 7, respectively.

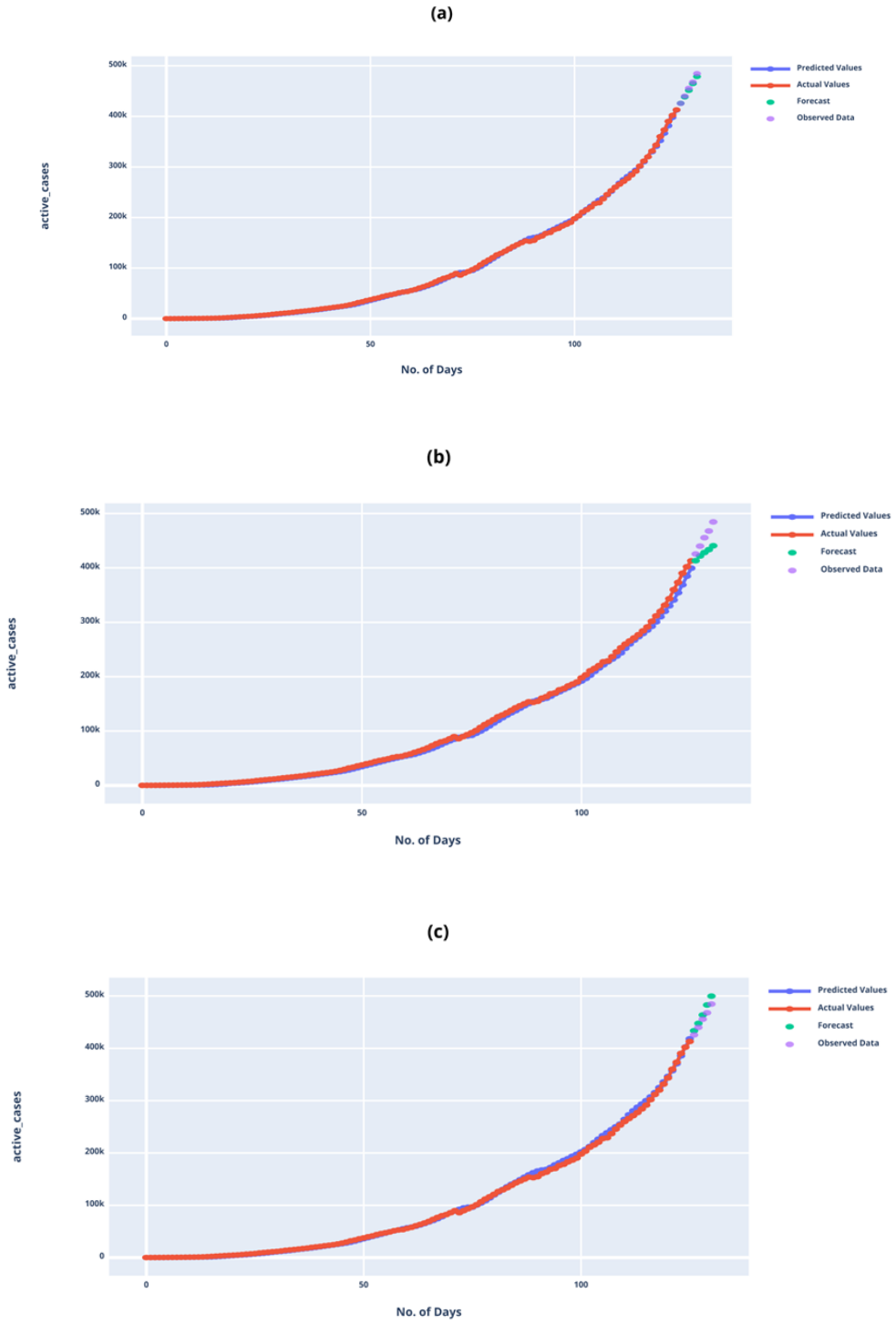

Figure 5. Active cases prediction using a) GRU b) Vanilla LSTM c) Stacked LSTM.

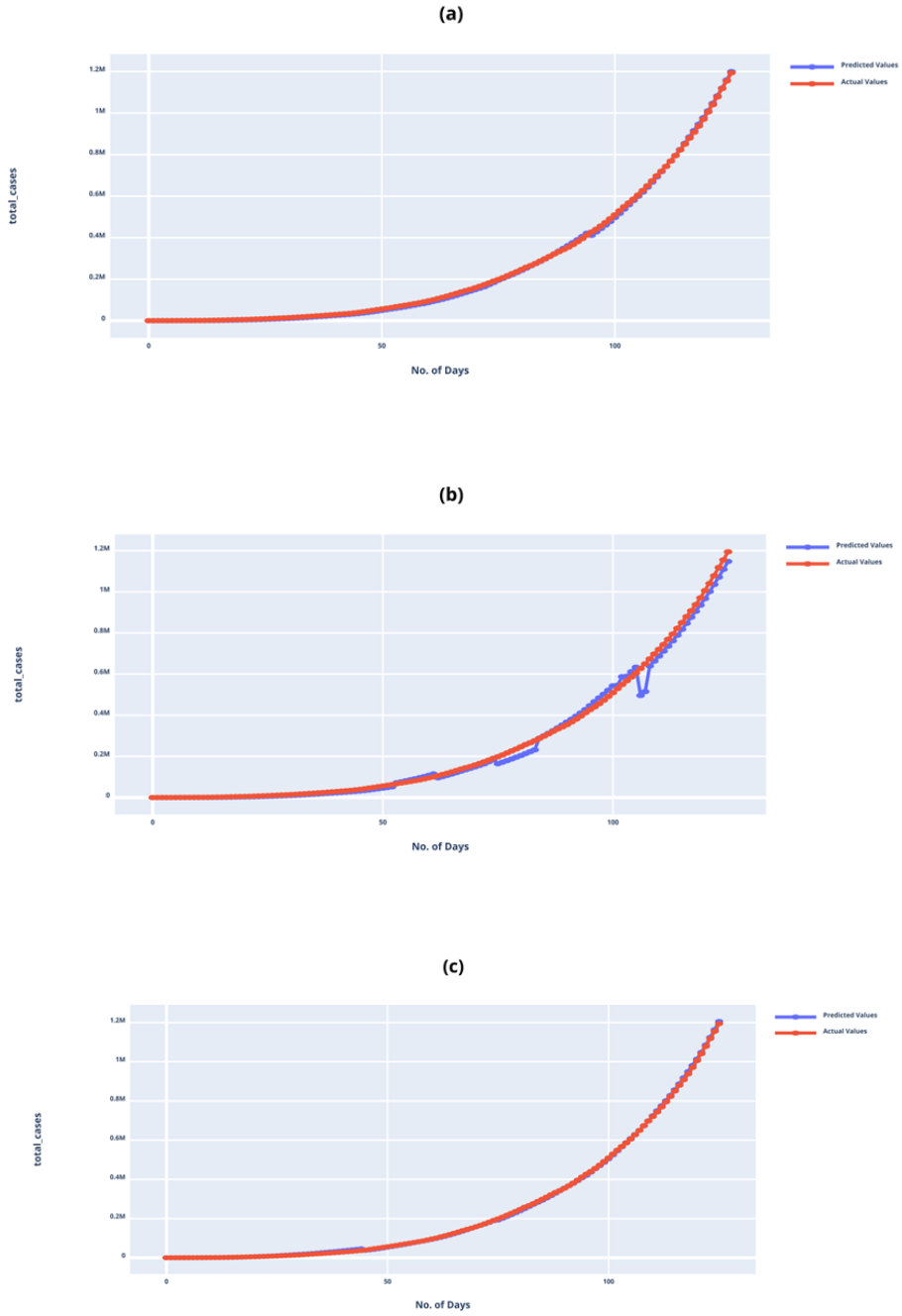

Figure 6. Multivariate time series analysis of total confirmed cases using a) GRU b) Vanilla LSTM c) Stacked LSTM.

4. Conclusions

In this work, RNNs based on GRU and LSTM is used for univariate and multivariate time series analysis and forecasting. The study is conducted on various time-series taken from the COVID-19 India dataset. This study compares various models that can be employed in the real-world for forecasting tasks to fight against COVID-19. It is observed that in univariate time series analysis, GRU performs better than Vanilla and Stacked LSTM for forecasting total confirmed and active cases while Stacked LSTM works better in forecasting total deaths. However, in multivariate time series analysis, Vanilla LSTM performs better than the other two models. GRU comes out to be an all-rounder performing satisfactorily in all cases.

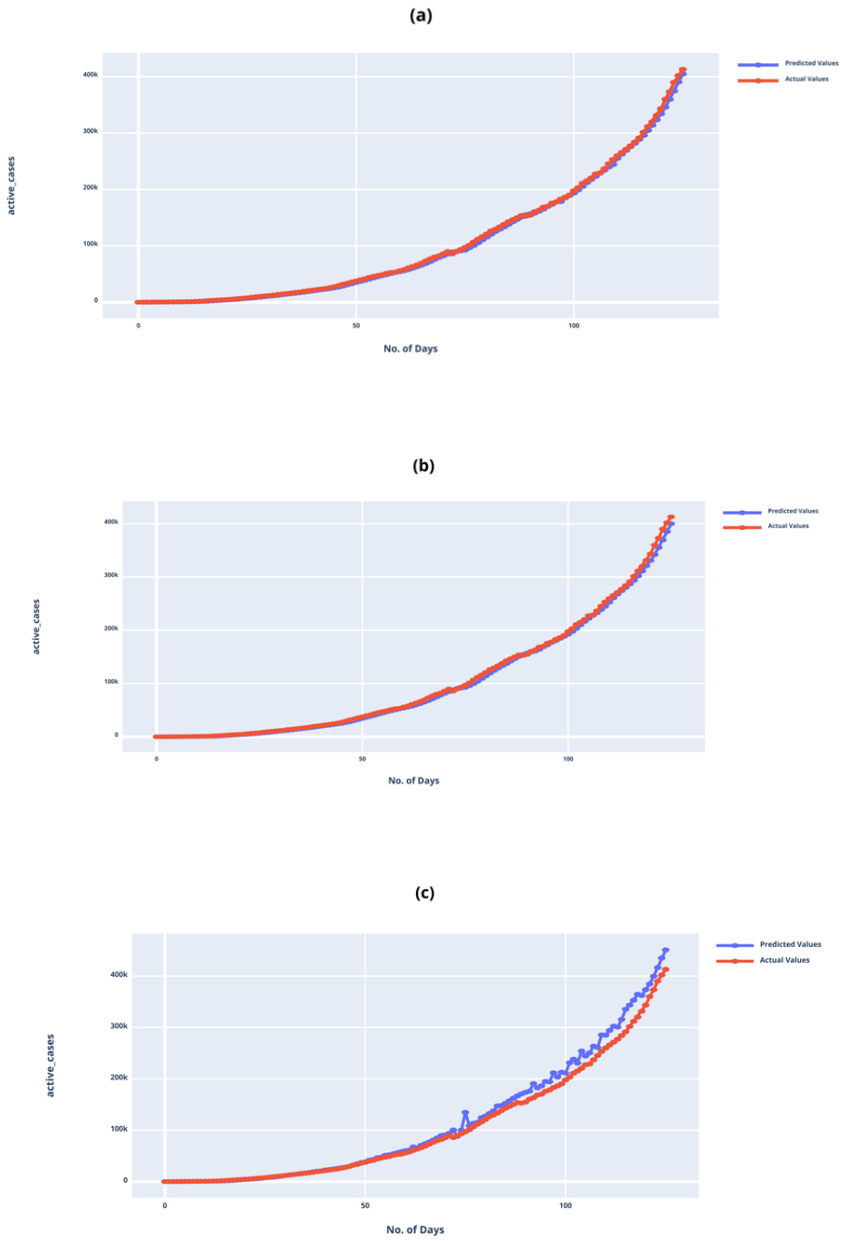

Figure 7. Multivariate time series analysis of active cases of COVID-19 using a) GRU b) Vanilla LSTM c) Stacked LSTM.

Funding

This research received no external funding.

Acknowledgments

We would like to thank our respective University.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Liu, J.; Liao, X.; Qian, S.; Yuan, J.; Wang, F.; Liu, Y.; Wang, Z.; Wang, F.S.; Liu, L.; Zhang, Z. Community Transmission of Severe Acute Respiratory Syndrome Coronavirus 2, Shenzhen, China, 2020. Emerg Infect Dis 2020, 26, 1320-1323, https://doi.org/10.3201/eid2606.200239.
2. Chan, J.F.-W.; Yuan, S.; Kok, K.-H.; To, K.K.-W.; Chu, H.; Yang, J.; Xing, F.; Liu, J.; Yip, C.C.-Y.; Poon, R.W.-S.; Tsoi, H.-W.; Lo, S.K.-F.; Chan, K.-H.; Poon, V.K.-M.; Chan, W.-M.; Ip, J.D.; Cai, J.-P.; Cheng, V.C.-C.; Chen, H.; Hui, C.K.-M.; Yuen, K.-Y. A familial cluster of pneumonia associated with the 2019 novel coronavirus indicating person-to-person transmission: a study of a family cluster. The Lancet 2020, 395, 514-523, https://doi.org/10.1016/S0140-6736(20)30154-9.
3. Huang, C.; Wang, Y.; Li, X.; Ren, L.; Zhao, J.; Hu, Y.; Zhang, L.; Fan, G.; Xu, J.; Gu, X.; Cheng, Z.; Yu, T.; Xia, J.; Wei, Y.; Wu, W.; Xie, X.; Yin, W.; Li, H.; Liu, M.; Xiao, Y.; Gao, H.; Guo, L.; Xie, J.; Wang, G.; Jiang, R.; Gao, Z.; Jin, Q.; Wang, J.; Cao, B. Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. The Lancet 2020, 395, 497-506, https://doi.org/10.1016/S0140-6736(20)30183-5.
4. Ong, S.W.X.; Tan, Y.K.; Chia, P.Y.; Lee, T.H.; Ng, O.T.; Wong, M.S.Y.; Marimuthu, K. Air, Surface Environmental, and Personal Protective Equipment Contamination by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) From a Symptomatic Patient. JAMA 2020, 323, 1610-1612, https://doi.org/10.1001/jama.2020.3227
5. Wu, C.; Chen, X.; Cai, Y.; Xia, J.a.; Zhou, X.; Xu, S.; Huang, H.; Zhang, L.; Zhou, X.; Du, C.; Zhang, Y.; Song, J.; Wang, S.; Chao, Y.; Yang, Z.; Xu, J.; Zhou, X.; Chen, D.; Xiong, W.; Xu, L.; Zhou, F.; Jiang, J.; Bai, C.; Zheng, J.; Song, Y. Risk Factors Associated With Acute Respiratory Distress Syndrome and Death in Patients With Coronavirus Disease 2019 Pneumonia in Wuhan, China. JAMA Internal Medicine 2020, 180, 934-943, https://doi.org/10.1001/jamainternmed.2020.0994.
6. Lu, R.; Zhao, X.; Li, J.; Niu, P.; Yang, B.; Wu, H.; Wang, W.; Song, H.; Huang, B.; Zhu, N.; Bi, Y.; Ma, X.; Zhan, F.; Wang, L.; Hu, T.; Zhou, H.; Hu, Z.; Zhou, W.; Zhao, L.; Chen, J.; Meng, Y.; Wang, J.; Lin, Y.; Yuan, J.; Xie, Z.; Ma, J.; Liu, W.J.; Wang, D.; Xu, W.; Holmes, E.C.; Gao, G.F.; Wu, G.; Chen, W.; Shi, W.; Tan, W. Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding. The Lancet 2020, 395, 565-574, https://doi.org/10.1016/s0140-6736(20)30251- 8.
7. Mohammadhassan, R.; Fallahi, S.; Mohammadalipour, Z.; ADMET and pharmaceutical activity analysis of caffeic acid diversities by in silico tools. Letters in Applied NanoBioScience2020, 9, 840-848, https://doi.org/10.33263/LIANBS91.840848.26
8. Mahato, S.; Pal, S.; Ghosh, K.G. Effect of lockdown amid COVID-19 pandemic on air quality of the megacity Delhi, India. *Sci. Total Environ.* **2020,** *730*, https://doi.org/10.1016/j.scitotenv.2020.139086.
9. Knight, G.M.; Dharan, N.J.; Fox, G.J.; Stennis, N.; Zwerling, A.; Khurana, R.; Dowdy, D.W. Bridging the gap between evidence and policy for infectious diseases: How models can aid public health decision-making. *Int. J. Infect. Dis.* **2016**, *42*, 17–23, https://doi.org/10.1016/j.ijid.2015.10.024.
10. Chimmula, V.K.R.; Zhang, L. Time series forecasting of COVID-19 transmission in Canada using LSTM networks. *Chaos, Solitons & Fractals* **2020**, *135*, https://doi.org/10.1016/j.chaos.2020.109864,
11. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015,** *521*, 436-444, https://doi.org/10.1038/nature14539.
12. Deng, L. Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing* **2014**, *7*, 197–387, https://doi.org/10.1561/2000000039
13. Freeman, B.S.; Taylor, G.; Gharabaghi, B.; Thé, J. Forecasting air quality time series using deep learning. *J Air Waste Manag Assoc.* **2018**, *68*, 866–886, https://doi.org/10.1080/10962247.2018.1459956.
14. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* **2014**.
15. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: a review. *Data Min Knowl Discov.* **2019**, *33*, 917–963, https://doi.org/10.1007/s10618-019-00619-1
16. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Netw Learn Syst* **1994**, *5*, 157–166.
17. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **1997**, *9*, 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735
18. Cho, K.; van Merrienboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* **2014**, 103–111.