Long Short-Term Memory (LSTM) Based Deep Learning Models for Predicting Univariate Time Series Data

Ashish Kharel*, Zeinab Zarean, and Devinder Kaur

EECS Department, University of Toledo, Toledo, USA
Email: akharel@rockets.utoledo.edu (A.K.); zzarean@rockets.utoledo.edu (Z.Z.); dkaur@rockets.utoledo.edu (D.K.)
*Corresponding author

Manuscript received June 8, 2023; revised July 28, 2023; accepted August 24, 2023; published March 20, 2024

***Abstract*—This paper explores the three variants of Long Short-Term Memory (LSTM) deep learning models for the analysis and prediction of univariate time series data to develop better understanding of the spread of COVID-19 pandemic. The COVID-19 pandemic continues to significantly impact public health, medical and industrial infrastructure, and the economy. Many researchers are working on various computational and mathematical models to analyze the underlying causes of transmission and spread of COVID-19. Accurate predictive models for COVID-19 will be significant as they will help us better manage the resources of health care professionals and medication to combat the COVID-19 pandemic. This study explores three deep learning models based on the Long Short-Term Memory (LSTM) cells to analyze and predict sequential data. These models are vanilla LSTM, bidirectional LSTM, and stacked LSTM. The models were trained and tested using the univariate time series data of daily trends in number of COVID-19 cases in the United States. Data was collected from the Centers for Disease Control and Prevention website (CDC) from January 23, 2020, to May 25, 2022. The models were trained using the first 743 samples and tested on the last 104 samples. The models were implemented using TensorFlow, and KerasTuner was used to optimize the hyperparameters of LSTM networks. The prediction accuracy of three models was compared using the metric of Mean Absolute Percentage Error (MAPE). It was found that bidirectional LSTM gave the best accuracy.**

***Keywords*—Long Short-Term Memory (LSTM), deep learning, univariate time series, data, COVID-19, comparison of different LSTM models**

I. Introduction

In 2019, scientists discovered a novel coronavirus, later called COVID-19 in Wuhan, China. COVID-19 is a highly transmissible respiratory disease that transfers from person to person through contact [1]. Soon after its discovery, the disease rapidly spread across the world. In March 2020, the World Health Organization (WHO) proclaimed COVID-19 a global pandemic, given that it was affecting all countries worldwide. The Centers for Disease Control and Prevention (CDC) confirmed the first case of COVID-19 in the United States on January 20, 2020, in Washington, DC. More than 486 million COVID-19 cases and over six million deaths were documented.

Accurate prediction of the COVID cases will enable the medical community to better manage the staffing and other infrastructure needed to handle the new patients. The conventional techniques for times series prediction are based on mathematical and statistical methods and other generic machine learning models. The shallow neural network-based techniques are not efficient in handling the long-term dependencies in sequential data [2].

Deep learning techniques have been deployed effectively in many complex real world prediction issues, including time series forecasting [3]. Deep learning has proven its ability to be successful at navigating the noise and chaos in time series predictions. LSTM is not only effective at capturing information about sequence, but also can be customized to only use relevant features from the training data and incorporate temporal correlations. In this research, several forecasting models are evaluated for time series prediction of confirmed cases, deaths, and recoveries.

II. Literature Review

Different mathematical and machine learning models have been used to investigate the pandemic parameters and eventually guide the pandemic response. For instance, Cooper, Mondal, and Antonopoulos [4] analyzed the effectiveness of modeling on the pandemic by proposing the well-known SIR model. The SIR model can provide insights about and predict the spread of the virus in communities and populations that recorded data alone cannot provide. The researchers looked at the evolution of different populations over time and tracked a variety of factors to see how the disease progressed in these groups. They concluded that COVID-19 can be contained if appropriate limits and strong policies are put in place to control infection rates early in the spread of disease.

Several machine learning and deep learning models have been trained to diagnose and predict COVID-19. [5] applied statistical and machine learning approaches using data collected through an online questionnaire to predict potential patients of COVID-19. Based on their research, the Multi-Layer Perceptron and the Support Vector Machine have proven to be most accurate. Domenico Benvenuto [6] found that an autoregression ARIMA-based model to predict the spread of COVID-19 is successful in short-term prediction but not long-term prediction. [7] investigated the accuracy of the 2 recurrent neural network model of long short-term memory (LSTM), bidirectional LSTM, and encoder-decoder LSTM models for predicting short-term COVID-19 infection. Their results revealed that the univariate encoder-decoder-LSTM model delivers the best test performance compared to the rest. In other comparisons too, LSTM has proven to have a strong forecasting accuracy. In looking at various models' ability to forecast the stock market, including LSTM, SVM, back propagation, and Kalman filter for different epochs varying from 10 to 100, LSTM demonstrated high accuracy and low variance [8]. [9]

compared the performance of different LSTM architectures for COVID-19 dataset of India. [10] compared different LSTM architectures and hybrid CNN-LSTM for Indian Covid-19 dataset. But the hyperparameters were tuned manually. However, both [9, 10] relied on Indian COVID 19 dataset.

III. Background

A. Time Series Data

Time series is a sequence of data points recorded over time generally equally spaced. It tracks how a given set of data changes over time and specifies what factors influence it from period to period. Examples of a time series are the daily value of the stock market or monthly rainfall over several years.

Any time series can have statistical properties. They may include the mean, median, mode, and standard deviation. If the statistical properties don't vary over time, this refers to a stationary time series. If they do vary over time, the time series is said to be nonstationary. For example, monthly totals of international air passengers are considered a stationary time series, while the price of bitcoin is a nonstationary time series due to many fluctuations over time. Time series data is often used in statistical and machine learning applications for various tasks such as prediction, classification, and clustering. In this research, we have used time-series data to make a scientific prediction.

B. Recurrent Neural Network

Recurrent neural networks are a type of neural network with a feedback loop which enables the output of the current node to be fed back into the input. RNNs allow the network to keep information from previous steps and use them to predict future values. They apply a recurrence relation at every time step to process a sequence. RNNs have hidden states which function as memory stores for all the information computed. RNN is one of the most widely implemented deep learning techniques. It has been utilized in large applications such as Google Voice, Siri, and Google Translate [11].

Fig. 1 represents the folded and unfolded RNN. The folded version illustrates the general structure of RNN, which has input (xt), hidden state (ht), and output (yt).

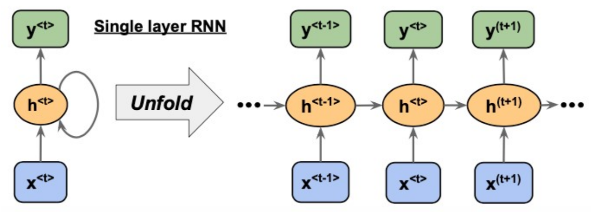

Fig. 1. Folded and unfolded RNN [11].

A loop allows information to be passed from one step of the network to the next. The unfolded diagram represents the repeating module in an RNN. The individual's time steps are unrolled across time. Even though the unfolded RNN shows several hidden states, it is always within the same block. The hidden state can capture past information about the sequence up to the current time step. It carries past information by applying a linear combination of the previous step and the current input. Then it runs them through an activation function to obtain the current time output. The key feature of RNN is that it uses the same function with the same set of parameters at every time step. RNN can model sequential data by propagating through time, i.e., both forward and backward propagation between units, from the last processing level to the first level.

C. Vanishing Gradient Problem

The recurrent neural network model can be effective when working with short-term contexts but has a long-term dependency issue called the "vanishing gradient". The vanishing gradient was first discovered by Joseph Horichter in 1991 and later by Yashua Benjio in 1994.

In a short-term model, the gradient of a loss function in each iteration is used to update neural network parameters, thus allowing the network to learn. It measures how much the function's output changes if a small change occurs at the input. The vanishing gradient problem happens when the gradient from backpropagation cannot reach the earlier states, and its values become too small, causing the model to stop or slow down learning. The RNN doesn't learn long-range dependencies and suffers from short-term memory problems. LSTM can overcome these problems effectively [12].

D. LSTM

Long short-term memory, known as LSTM, is a kind of recurrent neural network capable of learning long-term dependencies. Sepp Hochreiter and Juergen Schmidhuber [12] introduced it to address the problem of long-term dependencies. LSTM networks have the same control flow as a re current neural network. It processes data sequentially, passing on information as it propagates forward. Unlike RNNs, the operations within the LSTM cell allows the network to forget or retain information for a long period of time.

1) LSTM cells

The core concept of LSTM is cell state and its various gates. Fig. 2 shows a basic LSTM cell.

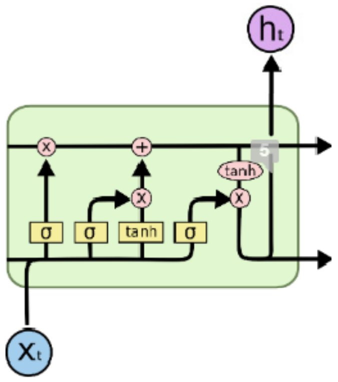

Fig. 2. The LSTM cell [13].

The cell state behaves like long-term memory, which stores longer term dependency and patterns. It transfers relative information throughout the sequence chain; even information from an earlier time step can be transmitted all the way to the last time step, thus reducing the effect of short-term memory. In the cell state, the gradient is stabilized due to the gate mechanism. It is therefore less likely to have a vanishing gradient issue, which is the main problem with training RNN. The cell state consists of three gates: an input, a forget, and an output gate. The gates are the mechanisms that control how information flows into and out of the cell state. They allow or deny data the ability to pass through during the training process based on the data's importance in making predictions. Gates consist of a sigmoid neural net layer with a point-wise multiplication operation.

2) *Repeating LSTM structure*

Fig. 3 shows an illustration of the repeating LSTM module.

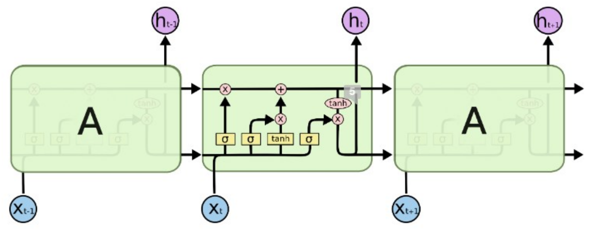

Fig. 3. The repeating LSTM structure [13].

E. *Stacked LSTM*

The original LSTM model consists of a single hidden LSTM layer followed by a feedforward output layer. An LSTM model with more than one LSTM layer is a stacked LSTM architecture. Fig. 4 shows an example of a stacked LSTM model. Stacking hidden LSTM layers make the model deeper. The depth of neural networks is generally related to greater accuracy. The more hidden layers there are, the more each layer can recombine learned representation from preceding layers and create new representations with higher degrees of abstraction [12].

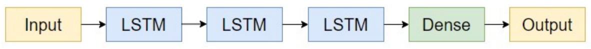

Fig. 4. Stacked LSTM [13].

F. *Bidirectional LSTM*

A bidirectional LSTM is a type of recurrent neural network that is trained in two directions (forward and backward) at once and can utilize information from both sides. Fig. 5 shows a bidirectional LSTM. It duplicates the LSTM layer so that there are now two layers that are aligned side-by-side.

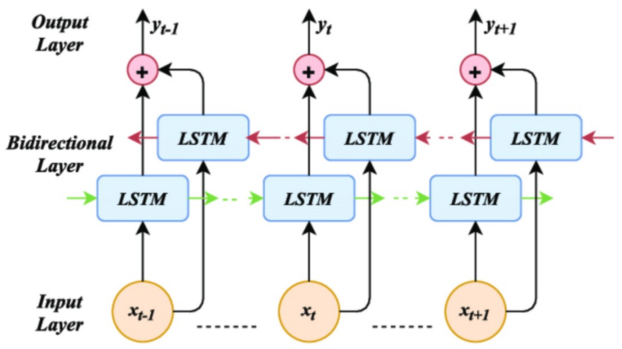

Fig. 5. The unfolded bidirectional LSTM. [14]

Bidirectional LSTM structures use two independent LSTM networks. One network is trained in the forward data sequence. The other one is trained in a reversed data sequence. The individual LSTM cells can learn the context from future information by providing a reversed copy of the input data. It allows the network to understand the relationships between values in both directions. At any timestep, the output from each forward and backward cell output is concatenated through the activation function to produce a single output. Hence the network, at any time, can process both the past and future information compared to the unidirectional LSTM's ability to process only past information.

IV. DATASET AND PREPROCESSING

This section shows how the LSTM network was used to predict the daily new cases of COVID-19. The vanilla LSTM, stacked LSTM, and bidirectional LSTM were used to develop the neural network models. All three models have been applied to the same dataset. The accuracy of the models is compared by calculating the minimum absolute percentage error (MAPE). The models will be fitted using the efficient ADAM optimization algorithm and the mean squared error as a loss function. The proposed LSTM models were implemented in Python 3.9 using Keras API running on top of TensorFlow 2.6.

Table 1. Tabular sample of new daily cases in the United States

Date	New Cases	7-Day Moving Avg	Historic Cases
May 25, 2022	160580	110304	0
May 24, 2022	134330	108925	0
May 24, 2022	113095	108294	0
May 23, 2022	49097	109067	0
May 22, 2022	41447	109740	0

A. *Dataset*

The dataset represents the COVID-19 daily trend in the United States from January 23, 2020, to May 25, 2022, [1]. In addition to the number of new cases, the dataset includes Date, 7-day moving average, and Historical Cases. Table 1 shows the tabular sample of the new daily cases in the United States between for 4 days. The dataset is used in the univariate time-series format since we only considered new COVID-19 cases that are reported daily. Fig. 6 shows the raw data of new daily COVID cases as retrieved from the CDC website. It shows it peaked in early 2021, that is attributed to the Alpha variant, and then it peaked again in January 2022 that is attributed to the Delta variant.

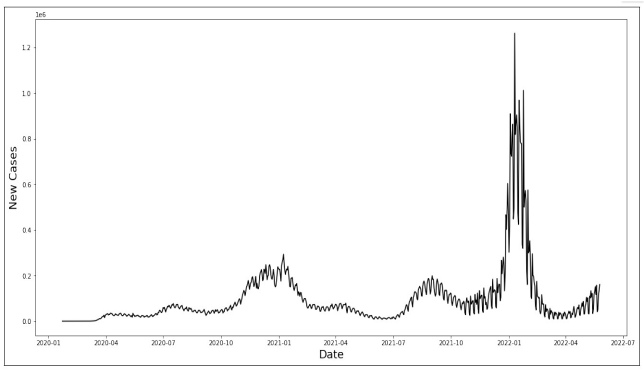

Fig. 6. Graph of new daily cases in the USA.

B. Data Pre-processing

Data preprocessing is the process of preparing raw data in a proper format that the model can accept. In this research, data preprocessing consists of moving average, scaling data, splitting data, and processing the time series data for supervised learning.

1) Moving average

Moving average, also called rolling average, also known as a rolling mean, is a statistic that is calculated by taking the mean of a data set over a certain period of time. It is used to smooth data performance by filtering out "noise" caused by random data fluctuations. Moving average on time series data is used to smooth out irregularities (peaks and valleys) to observe overall trends better. There are different moving averages, but the simple moving average (SMA) is the most common. The SMA is calculated by taking the average of the past n data points, where n is the number of time steps. A time step is the basic unit of measure used in time series data. A time step can be any length of time, such as 1 year, 1 month or even 1 hour, which is daily in our case. For example, if the time steps are 20, the SMA will be the average of the past 20 data points. The moving average can be used to identify the direction of the trend, as well as support and resistance levels. If the value is above the moving average, it is typically seen as an upward trend. Similarly, if the value is below the moving average, it is seen as an indication of a downward trend. Fig. 7 shows the moving average graph of the COVID data.

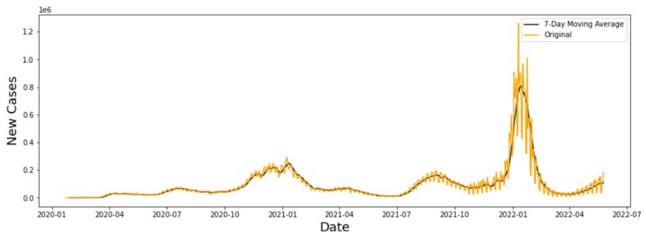

Fig. 7. Smoothng of a noise data (yellow curve) with a moving average (black curve).

2) Data scaling

Data preprocessing is essential to achieve better data quality before further processing. When a model receives non-scalable data, network learning and convergence may be reduced and impact the performance of the model in an adverse fashion. Data processing transforms the raw data into a more appropriate format for modeling. One common practice is to normalize the data, that involves scaling the data within a specified range. Scaling the data helps the LSTM model to learn and converge faster. Scaling also reduces the complexity of our models and can make our results easier to interpret.

3) Train/test sets

Machine Learning models are designed to predict the future value of a given data. The goal is to estimate the machine learning model's performance on new data that the model has not previously seen. The original data is divided into two subsets: training and testing sets. A training set is used to train and build the model by learning the parameters of the model based on the training set. train. Testing data is a data set used to evaluate the performance of a machine learning model. This data is used to assess the model's performance on unseen data. The data was parsed as 90% training and 10% testing for developing and training the model. An additional 3% of the data was used for validation of the model which was not used for testing or training the model.

V. Design and Implementation

A. Sliding Window Technique

Most machine learning problems use a supervised learning algorithm in order to make predictions, since the input and output are given. Using previous time steps to predict the next time step is called the sliding window method or lag method. We use the sliding window technique to restructure data into a machine learning problem in order to use all the tools and techniques available to train and optimize our model [15]. Table 2 shows the data which has a moving window of size 4 to parse the CDC data as four inputs and one output.

Table 2. Sliding window for time series. The window size is four

Date	Input1(t-4)	Input2 (t-3)	Input3(t-2)	Input4(t-1)	Output
Mar 1, 2020	51	70	78	130	146
Mar 2, 2020	70	78	130	146	214
Mar 3, 2020	78	130	146	214	390
Mar 4, 2020	130	146	214	390	498
Mar 5, 2020	146	214	390	498	530

The training dataset for our deep neural network required sliding windows x (input) and y (output) of a window size of 20. Both the x and y windows slide by a single increment to generate training data, as illustrated in Fig. 8.

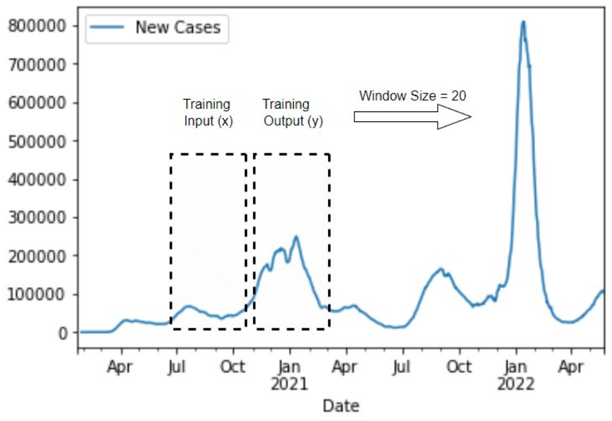

Fig. 8. The COVID-19 data showing the x and y sliding window.

B. Hyperparameters of LSTM Network Architecture

LSTM network architecture has a significant impact on network performance. There are parameters and hyperparameters in every machine learning task. The network learns parameters by training and adjusting them through backpropagation. However, there is no rule for designing and selecting hyperparameters, even though they significantly affect the accuracy of the model [16]. They are either adjusted by the user employing optimization algorithms, or trial and error. The typical hyperparameters are as follows:

1) Number of hidden layers

The number of layers between the input and output layers are called hidden layers.

2) Number of neurons in the hidden layers

The number of neurons in the input layer is equal to the number of features in the data, which is one in our case. The number of neurons in the output depends on whether the model is used as a regressor or classifier. Since we are dealing with a regression problem, we only have one output. But the number of neurons in hidden layer needs to be determined.

3) Number of nodes in dense layer

A dense layer is basically a layer where each node receives input from all of the nodes in the previous layer. Therefore, they are densely connected.

4) Number of time steps

It is the Sequence Length that the LSTM will receive when making predictions.

5) Number of epochs

One Epoch is when an entire dataset is passed forward and backward through the neural network once.

6) Batch size

Total number of training examples present in one batch. Batch Size is the number of samples we send to the model at a time.

Table 3 shows the hyperparameters that were selected for optimization and their optimized values.

Table 3. Optimal hyperparameters after tuning

Hyperparameters	Range	Optimal Hyperparameters
Number of hidden layers	1 to 5	3
Number of nodes per LSTM layer	50 to 500	150,50
Number of nodes in dense layer	32 to 512	32
Time Step	5 to 50	20
Epochs	1 to 1000	200
Batch Size	1 to 64	16

C. Keras Tuner

KerasTuner has developed a hypertuner which configures a space search in order to deploy a search algorithm, which finds the best hyperparameter combination [17]. We implemented a hypermodel using the model building function. It tries to find the values of the hyperparameters which minimize the loss function of the machine learning model. A model-building function is a function that takes the argument 'hp', which stands for hyper parameters. It defines the hyperparameters as inputs and returns a compiled Keras model. We applied the same method to optimize the number of nodes in the dense layer, number of epochs, batch size and time steps. Given the model, the next step is to initiate the tuner to perform hyper tuning. There are four different tuners in KerasTuner - RandomSearch, Hyperband, BayesianOptimization, and Sklearn. We chose the RandomSearch tuner, which is a space search algorithm that randomly samples values for the hyperparameters of a machine learning model. We chose RandomSearch because our study is focused on the performance comparison of three different LSTM architectures.

D. Vanilla LSTM Model

Fig. 9 depicts the structure of vanilla LSTM. First, the LSTM layer receives the input sequence. It passes them to the LSTM cells, which are equal to the sequence length. Then, the output of LSTM cells passes through all the 32 nodes of dense layer and the nodes in final layer predict the target value. LSTM receives 3-dimensional input shape of [batch size, timestep, feature]. A batch is comprised of one or more samples. In our models we have 16 batches comprised of 20 samples (timesteps). The time step is 20, which means that the LSTM will take input data with 20-time steps. There is only one feature which is number of new cases since we are dealing with univariate time series. As Fig. 10 shows the number of LSTM cells is same as number of timesteps and number of units is the dimension of the hidden state (or the output). The model has 20 LSTM cells and 150 nodes in each cell.

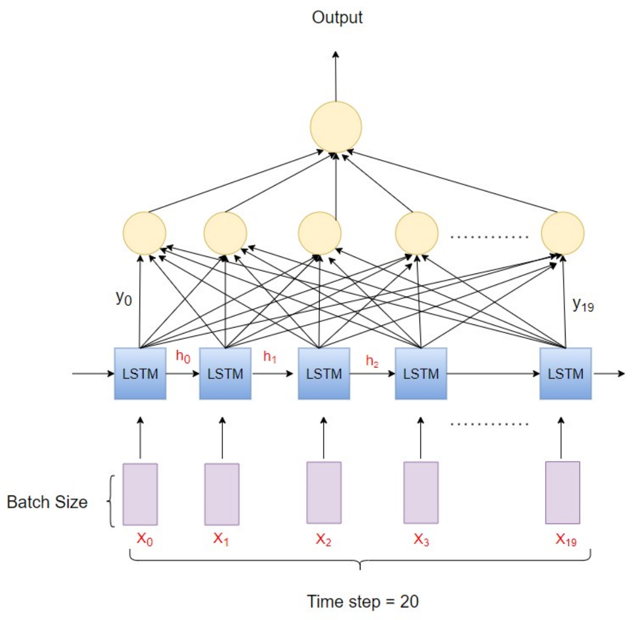

Fig. 9. The proposed vanilla LSTM architecture.

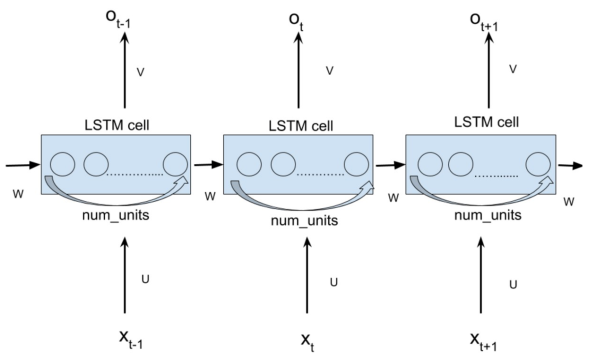

Fig. 10. Interpreting LSTM cells and number of units [18].

E. Stacked LSTM Model

Multiple hidden LSTM layers can be stacked one on top of another in what is referred to as a "stacked" LSTM model. Fig. 11 shows a stacked LSTM structure.

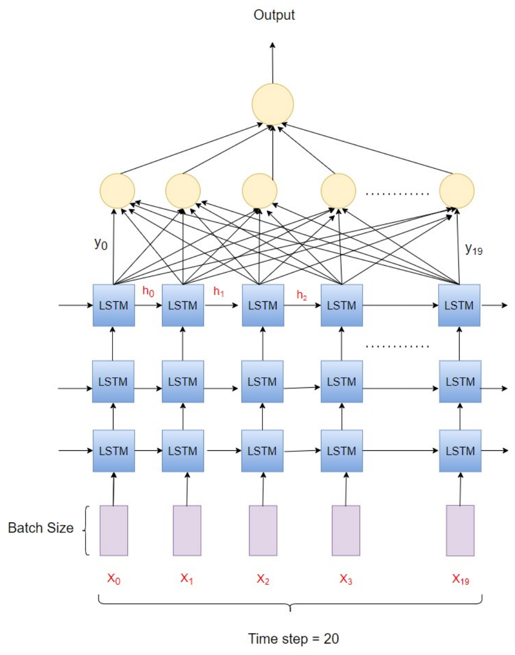

Fig. 11. Proposed stacked LSTM structure.

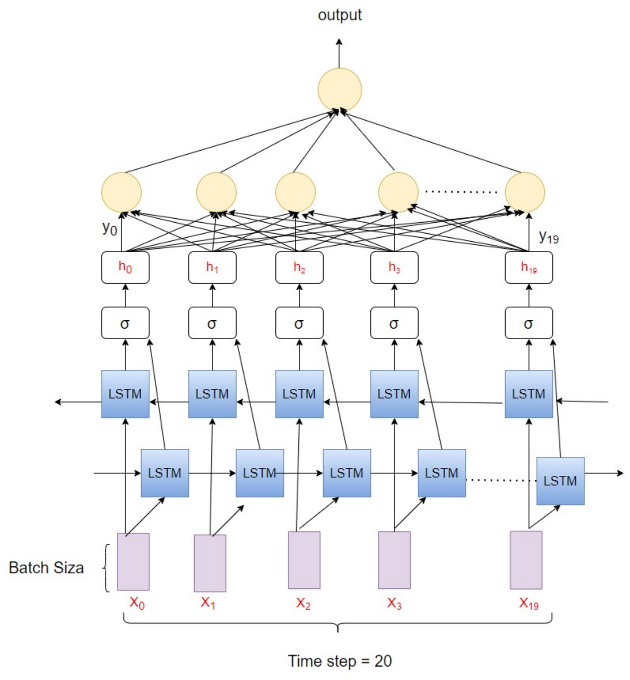

Fig. 12. Proposed bidirectional LSTM structure.

Each LSTM layer requires a three-dimensional input, and it will produce a two-dimensional output as an interpretation of the end of the sequence. Since we have more than one LSTM layer, the output should pass to the next layer but in the form of a three-dimensional input. We can address this issue by having the LSTM output a value for each time step in the input data by setting the 'return sequences=True' argument on the layer. This allows us to have 3D output from the hidden LSTM layer as input into the next. We defend a stacked LSTM model with three LSTM layers. All the model configurations are same as Vanilla LSTM model, except for the number of layers and the number of nodes. The first LSTM layer contains 150 neurons, the second and third layers are set to 50 neurons, and they are followed by two fully connected layers (dense layers).

F. *Bidirectional LSTM Model*

We can implement a bidirectional LSTM for a univariate time series forecast by wrapping the first hidden layer in a wrapper layer called Bidirectional layer. Fig. 12 shows a bidirectional LSTM structure. Bidirectional LSTM can read input both forward and backward. The main advantage of using a bidirectional LSTM is that it can learn long-term dependencies from both the left and right context. A bidirectional LSTM is a neural network that contains two hidden layers, one for the forward context and one for the backward context. The hidden layers are connected to the input and output layers. The hidden layers are also connected to each other, so that the network can learn from both the past and future contexts.

G. *Flowchart*

Fig. 13 shows our proposed implementation of the LSTM networks for each of the three different architectures: Vanilla, Bidirectional, and Stacked LSTM architectures.

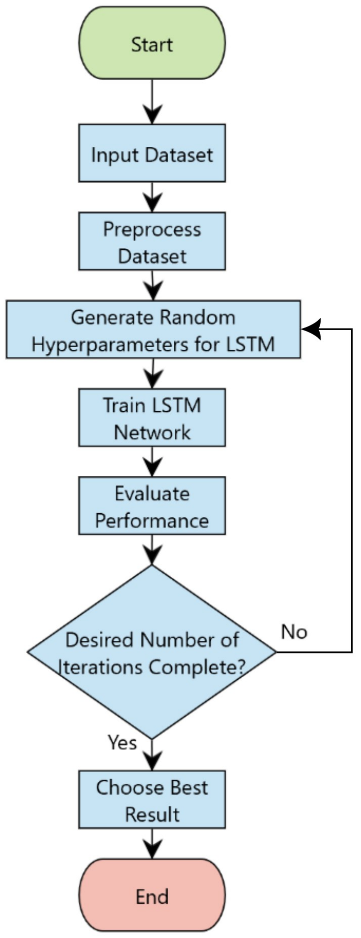

Fig. 13. Proposed implementation for comparing three different LSTM architectures.

VI. Results

A. *Accuracy of Vanilla LSTM*

Fig. 14 represents the predicted values vs actual values. It can be seen that while the model has followed the trend properly, the accuracy of the model is not very good, as shown by the non-overlapping lines.

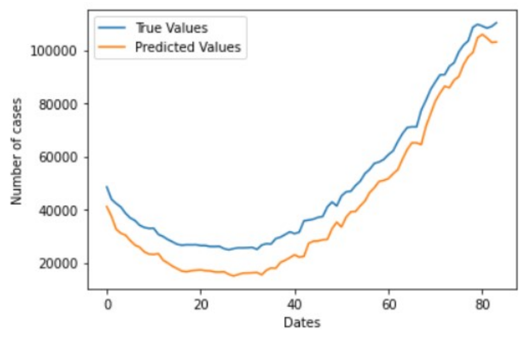

Fig. 14. Performance of vanilla LSTM.

B. *Accuracy of Stacked LSTM*

Fig. 15 represents the predicted values vs actual values resulting from the stacked LSTM. It can be seen that some regions in the figure still do not overlap. However, the accuracy is better compared to the vanilla LSTM models.

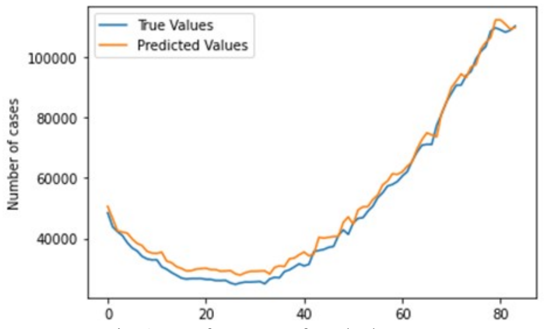

Fig. 15. Performance of stacked LSTM.

C. *Accuracy of Bidirectional LSTM*

Fig. 16 represents the predicted values vs actual values resulting from the bidirectional LSTM.

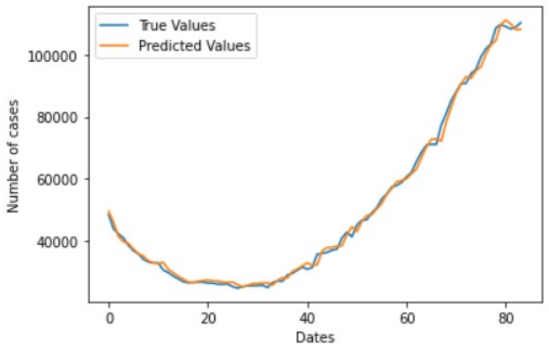

Fig. 16. Performance of bidirectional LSTM.

D. *Predictions for the Next 50 Days*

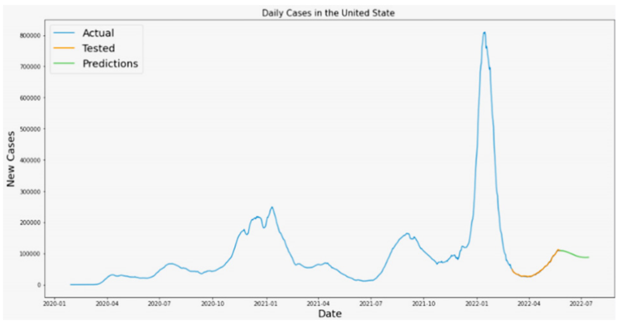

Fig. 17. Predictions.

The bidirectional model was used to predict the daily new cases of COVID-19 in the next 50 days after the trained database cut-off date (May 25, 2022), as shown in Fig. 17.

VII. Comparison of Accuracy of All Three Models

We found that bidirectional-LSTM has the best performance accuracy of all the tested models. Table 4 shows the performance comparison of all three models.

Table 4. Accuracy comparison of various LSTM architectures

Model	MAPE Scores
Vanilla LSTM	0.01
Stacked LSTM	0.003
Bidirectional LSTM	0.002

VIII. Conclusion

The performance analysis of three different deep neural network architectures based on LSTM viz. Vanilla LSTM. Stacked LSTM and Bidirectional LSTM was carried out using the univariate time series data, pertaining to COVID-19. The models were implemented in Python platform using Keras API built on TensorFlow. The models were compared using metric of mean absolute percentage error (MAPE). The performance of the Bidirectional LSTM model was the best.

In future work, these models can be tested on multivariant time series data sets. In COVID-19 data we can include vaccination data and lockdowns. Current models in this research can be compared with other models like LSTM Encoder-Decoder.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Dr. Kaur proposed the main idea related to this work. Ashish Kharel proposed the research plan and wrote the paper and Zeinab Zarean executed the research plan and performed statistical analysis on the data. Ashish Kharel and Zeinab Zarean worked under the supervision of Dr. Kaur.

References

[1] Centers for Disease Control and Prevention. Trends in number of COVID-19 cases and deaths in the US reported to cdc, by state/territory. [Online]. Available: https://covid.cdc.gov/covid-data-tracker/#trends_dailycases

[2] R. Ma, X. Zheng, P. Wang, H. Liu, and C. Zhang. (2021). The prediction and analysis of COVID-19 epidemic trend by combining lstm and markov method. [Online]. Available: https://www.nature.com/articles/s41598-021-97037-5

[3] G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. D. Filippo, A. D. Matteo, and M. Colaneri. (2020). Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32322102/

[4] I. Cooper, A. Mondal, and C. G. Antonopoulos. (2020). A sir model assumption for the spread of COVID-19 in different communities. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7321055/

[5] E. Fayyoumi, S. Idwan, and H. AboShindi. (2020). Machine learning and statistical modelling for prediction of novel COVID-19 patients case study: Jordan. [Online]. Available: https://thesai.org/Publications/ViewPaper?Volume=11Issue=5Code=IJACSASerialNo=18

[6] D. Benvenuto, M. Giovanetti, L. Vassallo, S. Angeletti, and M. Ciccozzi. (2020). Application of the arima model on the COVID-2019 epidemic dataset. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S2352340920302341

[7] R. Chandra, A. Jain, and D. S. Chauhan. (2022). Deep learning via lstm models for COVID-19 infection forecasting in India. [Online]. Available: https://doi.org/10.1371/journal.pone.0262708

[8] D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble. (2019). Comparison of predictive algorithms: Back-propagation, svm, lstm and kalman filter for stock market. [Online]. Available: https://ieeexplore.ieee.org/document/8701258

[9] A. Raj, N. R. Umrani, S. G. R. S. Audichya, A. Kodipalli, and R. J. Martis, "Forecast of COVID-19 using deep learning," presented at 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2021, pp. 1-5. doi: 10.1109/CONECCT52877.2021.9622721

[10] H. Verma, S. Mandal, and A. Gupta. Temporal deep learning architecture for prediction of COVID-19 cases in India. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422001038

[11] S. Raschka and V. Mirjalili, *Python Machine Learning. 3rd Edition. Birmingham*, UK: Packt Publishing, 2019.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. [Online]. Available: https://dl.acm.org/doi/abs/10.1162/neco.1997.9.8.1735

[13] C. Olah. (2015). [Online]. Available: Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[14] Ihianle, I. Nwajana, A. Ebenuwa, S. Otuka, R. Owa, K. Orisatoki, and Mobolaji, "A deep learning approach for human activities recognition from multimodal sensing devices," *IEEE Access,* 2020.

[15] A. Vidhya. (2022). Explained deep sequence modeling with rnn and lstm.

[16] F. Shahid, A. Zameer, and M. Muneeb. (2020). Predictions for COVID-19 with deep learning models of lstm, gru and bi-lstm. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077920306081

[17] Sandha, S. Singh *et al.*, "Mango: A python library for parallel hyperparameter tuning," presented at ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020.

[18] S. Verma. Understanding input and output shapes in LSTM — keras. (2019). [Online]. Available: https://shiva-verma.medium.com/understanding-input-and-outputshape-in-lstm-keras-c501ee95c65e/