Predicting Daily Confirmed Cases in Midwestern Central States in U.S. by Using AIMA and LSTM

Yi zheng
Marquette University

PREDICTING DAILY CONFIRMED CASES IN MIDWESTERN CENTRAL STATES IN U.S. BY USING ARIMA AND LSTM

by

Yi Zheng

A Thesis submitted to the Faculty of the Graduate School,

Marquette University,

in Partial Fulfillment of the Requirements for

the Degree of Master of Science

Milwaukee, Wisconsin

August 2021

ABSTRACT

PREDICTING DAILY CONFIRMED CASES IN MIDWESTERN CENTRAL STATES IN U.S. BY USING ARIMA AND LSTM

Yi Zheng

Marquette University, 2021

Covid-19 is an epidemic disease caused by SARS-Cov-2 virus, which is a type of coronavirus. This virus is highly contiguous, and the confirmed cases of this disease have increased rapidly in a short period. After one month of the first reported case, the World Health Organization (WHO) claims that the Covid-19 will become an international public health emergency.

The main purpose of this thesis is to predict the daily confirmed cases of Covid-19 in the midwestern central states in the U.S, by using Autoregression Integrated Moving Average (ARIMA) model and Long Short-Term Memory network (LSTM), which is a type of recurrent neural network. We compare the Root Mean Square Error (RMSE) for the prediction to determine the performance of the two methods.

In this thesis, we show that the LSTM network has a smaller prediction RMSE. Also, both models capture the seasonality of the dataset. LSTM captures the trend of the dataset and has a higher prediction than expected values. ARIMA does not capture the trend of the dataset and will have a larger range. Therefore, we can conclude that LSTM is a better method for predicting daily confirmed cases of Covid-19 in the Midwestern central states in the U.S.

TABLE OF CONTENTS

CHAPTER

I.INTRODUCTION……1

II. LITERATURE REVIEW……4

III.THEORY……8

A. ARIMA……8

B. SEASONAL ARIMA……9

C. ANN……9

D. RNN……13

E. LSTM……16

IV. Modeling COVID-19 DATA based on ARIMA and LSTM and Results……21

A. DATASET……21

B. ARIMA……23

C. LSTM……30

V.CONCLUSION……36

BIBLIOGRAPHY……37

Chapter 1: Introduction

Coronaviruses are a common group of RNA viruses that can cause disease among animals and humans. There are hundreds of known coronaviruses and 7 of them can infect humans. Four of those viruses cause mild respiratory infection such as the common cold, which are highly contagious. Two of those viruses, SARS and MERS, infect the lungs and cause more severe disease. For syndrome coronavirus (SARS-CoV-2), it not only spreads easily, but it can also cause a severe infection on the lungs [17]. This SARS-CoV-2 virus was first identified in Wuhan, China, in December 2019, so the World Health Organization named this virus as COVID-19 on 01/09/2020 [1]. The incubation period for COVID-19 is around 14 days. The symptoms of this virus included fever, cough, breathing difficulties and loss of smell and taste. Most people develop a mild symptom after exposure to the virus. Some people do not develop any noticeable symptoms after they are infected, but they are still contiguous. The rest of them, especially the elders or people suffer from other diseases such as heart disease or cancer, may develop more severe symptoms and might be critical [2].

The virus can be transmitted from human-to-human, and it is mainly spread by infected person's nose, mouse, and eye through droplets in close contact. It might also spread via contaminated surfaces. Since this is a highly contagious virus, the confirmed cases increase rapidly and cause a global pandemic in a short time.

In the middle of January 2020, one month after the first diagnosed case, China reported nearly 140 new confirmed cases daily. On 01/14/2020, the first confirmed case outside China was reported in Thailand. On January 30, 2020, the WHO declared the Covid-19 a public health emergency of international concern [2].

In the United State, the COVID-19 virus has greatly impacted people's daily lives as well. On 01/20/2020, the first confirmed case in the U.S was reported in Washington State, but it did not cause severe influence because the confirmed cases were still under control. However, the viruses rapidly spread to the whole country and the outbreak started to happen during March 2020. By 03/03/ 2020, two months after the first confirmed case in the U.S, 164,620 confirmed cases and 3,170 deaths were reported. The infected cases have grown rapidly since then. As of the end of April 2020, there were 1.04 million confirmed and 60,966 deaths reported. After that, the confirmed cases rose around 1 million per month on average. The situation became even worse after October 2020. The total confirmed cases increased 1 million every week on average. As of 03/08/2021, the total confirmed cases reported in the U.S had already passed 29 million cases and the number is still rising [3].

In order to predict future number of the infected cases, it is important to build a statistical model. This will provide valuable information of the virus and response to the situation. The future forecasting can be used to inform the public health officials about how severely the virus will influence people's lives and make decision to control the outbreak. Also, it can provide information and experience ahead if a similar virus discovers in the future.

In this thesis, the prediction of the confirmed cases in the Midwestern central states will be focused based on Autoregression Integrated Moving Average (ARIMA) and long short-term memory (LSTM) network, which are two popular methods to deal with time series dataset. ARIMA is used for stationary process and LSTM does not have such requirement. Also, the performance of these two models will be compared as well.

This thesis includes five chapters: Introduction, Literature review, Theory, Modeling Covid-19 data, and Conclusion. In chapter 2, literature review of existing Covid-19 forecasting articles will be presented. In chapter 3, theoretical details and mathematical explanation of ARIMA, neural network, recurrent neural network, and LSTM will be presented. In chapter 4, ARIMA and LSTM methods fitting in a Covid-19 daily confirmed cases dataset will be discussed and the forecasting performances of the two approaches will be compared.

Chapter 2: Literature Review

The issue of how to forecast the quantities related to Covid-19, such as death, confirmed cases and hospitalize rate, has been widely discussed in the literature since the pandemic began, and there have been several studies and articles published already [18][19][20].

The Autoregression Integrated Moving Average model (ARIMA) is one of the most traditional and popular approaches to explore time series datasets. It is widely used because it has great statistical properties. There have been a lot of studies and published articles on analyzing weather, financial, or epidemic disease dataset using ARIMA.

In the article by Sarbhan et al. in 2020 , the authors present the prediction of daily confirmed Covid-19 cases in Malaysia at an early stage using an ARIMA model [4]. The dataset used in this article was collected by Johns Hopkins University and the Malaysia government from 01/22/2020 to 03/01/2020 to predict daily confirmed cases from 04/18/2020 to 05/01/2020. Since the dataset is relatively small with a short period of time, the authors ignored the effect of seasonality and choose an ARIMA model instead of a Seasonal ARIMA model. In this article, several ARIMA models have been built and compared by choosing the lowest mean squared percentage error (MAPE). Authors choose ARIMA (0,1,0), where 0 are coefficients for autoregression and moving average and 1 is the coefficients for integrated. as the model to perform prediction. The accuracy of this model is high. The difference between accurate cumulative confirmed cases in the forecast period and prediction made by the model is 17 cases.

Long short-term memory is a type of recurrent neural network (RNN) which is good at analyzing sequence data. RNN is widely used in image reorganization and language

processing. There are also a lot of studies using LSTM in predicting time series dataset in other areas such as financial and signal processes. There are several published articles on analyzing Covid-19 dataset using LSTM as well. In the article published by Novanto in 2020, the growth of Covid-19 globally is predicted. Author uses dataset from 100 regions as training data and four regions as validation data [5]. The parameters inside dataset include the confirmed cases, death cases, recovered cases latitude and longitude. In this paper, the author uses 30 hidden states chosen by using a validation method. The result presented in the article shows that LSTM does not show a perfect quantitative prediction but shows a robust growth pattern. Also, the latitude and longitude shown to be a significant factor to predict the confirmed cases. The author also builds a RNN as a comparison with the performance of LSTM and LSTM is shown to have a lower RMSE on testing dataset.

Another paper written by Rahele and Roya et al. in 2021 presents the forecasting of Covid-19 in Iran by comparing the performance of several types of deep learning method including LSTM and several types of modified version of LSTMs [6]. The dataset used in this study contains the daily confirmed cases, daily death cases and daily recover cases, collected by John Hopkins University. The training dataset used was from 01/22/2020 to 07/30/2020. The testing data was from 08/01/2020 to 08/31/2020 to make the prediction. The authors build five different networks, including random forest, multilayer perception, long short-term memory, long short-term memory with extended features and multivariate LSTM. Then they compare the performance on testing data by using RMSE and MAPE is presented in the article and multivariate LSTM has shown the best performance among those deep learning methods.

In the article published by Munish and Surbhi et al. in 2020, a prediction of Covid-19 in India, Italy, Japan, Spain, UK and US using ARIMA and LSTM model is presented [7]. In this

study, dataset of confirmed and death cases in each of those six countries are collected from 02/22/2020 to 04/23/2020. The ARIMA models for confirmed cases and death cases are built individually for each of the countries. Then the estimated parameters with best performance was chosen for the prediction. An ARIMA model with retraining was also applied to make the prediction. The dataset was also implemented to a well-tuned LSTM model. The result shows that ARIMA with retraining has a better performance than LSTM model because of the lower RMSE. The LSTM network has a high accuracy of predicting the trend of the confirmed cases and death cases with a higher value. Also, LSTM model have a better performance on capturing a sudden rise happening in the dataset.

Since LSTM and ARIMA are both widely used in financial dataset, there are studies comparing the performance of those two methods using a financial dataset. In the article published by Sima and Akbar in 2018, the performance of forecasting financial time series on those two methods is compared [8]. The dataset used in the acritical was monthly financial situation from Yahoo finance website, including stock information from Jan 1985 to Aug 2018. The structure of the dataset used in the article is similar with Covid-19 dataset since they are all univariate time series datasets including a seasonality. The authors used the adjusted close variable of 12 stock to fit ARIMA and LSTM individually and showed the performance by using Root Mean Square Error (RMSE). In this acritical, authors fit a ARIMA(5,1,0) model and a standard LSTM network with loss function of MSE and optimizer of Adaptive Moment Estimation (Adam). For every stock, LSTM has a better performance. The performance of the prediction improves by 85% on average compare with ARIMA. This article also discussed if the training time in LSTM influence the accuracy of the model. The authors tried epoch values between 1-100 and record the error rate each time and there is

no significant test error improved. As the training time increases, there might even have an overfitting problem that could worsen the performance.

Chapter 3. Theory about ARIMA and LSTM

3.1.1 ARIMA

Autoregression Integrated Moving Average (ARIMA) model was generalized by Box and Jenkins in 1970 and has become one of the most traditional and wildly used model to analyzing time series dataset [8]. Autoregression integrated moving average model is a generalized model which includes three components to ARIMA (p, d, q):

Autoregression AR(p) is a regression model that includes its own lagged values. This process is used for making prediction on the time series dataset based on lagged values. The expression of this section can be written as followed:

$$X_t = c + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + Z_t,$$

where X_t is a stationary variable, c is a constant, ϕ_i is autocorrelation coefficients at lags i, 2…p and Z_t is the white noise of this dataset with mean 0 and variance σ^2.

Moving Average MA(q) is a method that weigh the prediction errors and apply to lagged prediction errors. This process is used for reducing error. The expression of this section can be written as followed:

$$X_t = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_p Z_{t-p},$$

where θ_i are the weights applied to the variables.

AR and MA components are applied to stationary time series. Integrated I(d) is used to make the time series stationary by differencing the time series at different time d. The expression of this section can be written as followed:

$$(1 - B)^d X_t$$

After combining the three components, the full model can be written as:

$$X_t{}^d = \phi_1 X_{t-1}{}^d + \cdots + \phi_p X_{t-p}{}^d + \theta_1 Z_{t-1} + \cdots + \theta_p Z_{t-p} + Z_t$$

where $X_i{}^d$ means X_t after different time d. [9]

3.1.2 Seasonal ARIMA

Seasonality commonly occurred on time series dataset and it has influence on analyzing time series dataset. An autocorrelation (ACF) plot and a partial autocorrelation (PACF) plot can be used to determine if seasonality occurred on the time series dataset. When seasonality is involved in the model, SARIMA model, which stands for seasonal autoregressive integrated moving average, is used instead.

The general form for a SARIMA model is ARIMA (p, d, q) × $(P, D, Q)_m$, where the p is the non-seasonal AR order, d is the non-seasonal differencing, q is the non-seasonal MA order, P is the seasonal AR order, D is the seasonal differencing, Q is the seasonal MA order and m is the number of seasonality. [9]

When implementing a SARIMA model on a dataset, the most important thing is to determine the values of p, d, q, P, D and Q. ACF plot and PACF plot can be used to find the values.

3.2.1 ANN

Artificial Neural Network (ANN) is a computer system inspired by the human brain and mimic how brain analyze and process information. The idea of ANN was first proposed by Warren McCulloch and Walter Pitts in 1943 [10]. After that, a lot of researchers added properties to the neural network to improve the performance and the artificial neural network became the more modern-like network that people widely use nowadays. ANN contains three types of layers: input layer, hidden layers, and output layer. All those layers are formed by nodes which is called artificial neurons. Every node in one layer is connected

with the nodes in next layer and the information is only pass forward. For each artificial neural network, there are only one input layer and one output layer. The number of hidden layers can be variable base on different design. As the number of hidden layers increases, the network becomes more complicated. The general idea of how artificial neural network work is straight forward. For each node, it will take the weighted sum as its input. Then add a bias b and pass it through a non-linear activation function, usually tanh, sigmoid or ReLu. Then the output of this node can be discovered, which will be used as the input for the next layer. This process is called forward propagation, which is the method training all types of neural network.

The next question is how to decide the weights and biases for all the nodes to optimize the network. In order to make more clear explanation on this question, Loss function, optimizer and backpropagation are needed which are defined below.

3.2.2 Loss function

Loss function, which is also called cost function, is a method to measure how unsatisfied the network perform, using the parameters chosen, by comparing the output with the actual values. The value of loss will be high if the network performs poorly, and it is low if the performance is good. Therefore, the goal of optimization is to find weights that will minimize the loss function. There are several types of loss functions that can be chosen due to different designs and dataset. When dealing with classification dataset, two of the most common lost function people used are support vector machine (SVM) loss and softmax. However, in the time series case such as predicting the daily confirmed cases of Covid-19, the outcome variable is quantitative. The least square loss function will be used instead of a linear classifier loss function. In the regression case, one of the commonly used

loss functions is the mean square error (MSE) loss. The mathematic expression of MSE is shown as followed:

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}$$

Some other regression loss functions such as mean absolute error is also available.

3.2.3 Optimizer

The goal of this algorithm is to minimize the loss function $L(\theta)$ by updating the parameters in the network, which are the weights and biases. In this study, Adaptive Moment Estimation (ADAM) will be used. This optimizer was first published in 2014 by Kingma and Jimmy [22]. This optimizer is widely used in practice and shown to have great performance in different deep learning architectures. Adam optimizer is a method that involve the first and second moment of the gradient to minimize the loss function, which is MSE in this study.

Frist, the gradient on a mini batch at time step t, which is the derivative of the function, needs to be calculated. This can be expressed as $g_t = \frac{\partial f(x,W)}{\partial}$. Then, calculate the first moment of the gradient $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ and the second moment of the gradient $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$. The commonly used defaulted value for hyper parameters β_1 and β_2 is 0.9 and 0.999. Also, m_t and v_t will be initialized as 0 at the first iteration. In Adam, the bias correction will be used to implement this method. The mathematical expression of the bias correction of m_t and v_t can be expressed as followed:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Then, the weight matrix can be updated by using the following function:

$$W_{t+1} = W_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

In this equation, α is the step size and ϵ is term that prevent the denominator to be 0. Usually, ϵ will be defaulted as 10^{-8}. α will control the speed of the gradient decent and will be defaulted as 0.001.

3.2.4 Backpropagation

Backpropagation is an algorithm of computing gradients and updated weights through a recursive application of chain rule, first used in neural network in 1986 by David Rumelhart, Geoffrey Hinton, and Ronald Williams [15]. In the paper, authors applied back propagation to several neural networks and compare the performance of tuning parameters with other approaches used before. The result shows that backpropagation algorithm is more effective than earlier approaches. Nowadays, the backpropagation algorithm is used in training every types of neural networks.

The general idea of backpropagation is straight forward. It starts from the end of the network and go through the network backward and compute all gradients along the way. The math of backpropagation algorithm depends on the following five equations [11]:

Partial Derivation : $\frac{\partial L}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1}$, where L is the loss function, w_{ij}^k is the weight for node *j* in layer *k* for incoming node *i*, δ_j^k is the error of node *j* in layer *k* and o_i^{k-1} is the output for node i in layer *k-1*.

For the final layer's error term: $\delta_1^m = g_o'(a_1^m)(\hat{y} - y)$, where g_0 is the activation function for the output layer nodes, a_1^m is the weighted sum plus bias in final layer, $\hat{y}$ is the computed output and y is target output.

For the hidden layers' error term: $\delta_j^k = g'(a_j^k)\sum_{l=1}^{r_{k+1}} w_{jl}^{k+1}\delta_l^{k+1}$, where r_{k+1} is the number of nodes in layer *k+1* and *g* is the activation function for hidden layer nodes.

Combining the partial derivatives for each input-output pair: $\frac{\partial L}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial}{\partial w_{ij}^k}(\frac{1}{2}(\hat{y}-y)^2)$

Finally, the weights can be updated by $\Delta w_{ij}^k = -\alpha\frac{\partial L}{\partial w_{ij}^k}$

After knowing those methods, a general method on how to train a neural network can be described as follows. The first step to train a neural network is to choose an architecture of neural network and initialize the weights and biases. Then perform the forward propagation and let all inputs go through hidden layers to get the output. The next step is to implement the loss function for optimizer. Then use back propagation to compute the gradient of loss. Finally, an optimizer can be used to tune the parameters. This process will be repeated until the smallest loss is reached.

3.2.5 RNN

Recurrent Neural Network (RNN) is a popular type of neural network first designed in 1985 by David Rumelhart, Geoffrey Hinton, and Ronald Williams [12]. Different from a feed-forward neural network, which is a network that information only moves in one direction and will not come back into the nodes, RNN recurrent the information. In a RNN network, it updates the hidden state, h_t, by considering the current input vector x_t and the hidden state that learns from previous time step (h_{t-1}) and passing through some function with parameters *w*. Also, the recurrent neural network has the same weight (w) during each time step when the model recurs. Because of the properties of RNN, this network can remember the information through time and is good at analyzing sequence dataset for

making predictions. The output does not have to be applied to each hidden state which gives RNN more flexibility and the ability to work in different area. In practice, recurrent neural network is commonly used in the area such as image recognition, classification, and language processing.

A more detailed figure showing how standard RNN works is described in Figure 1.

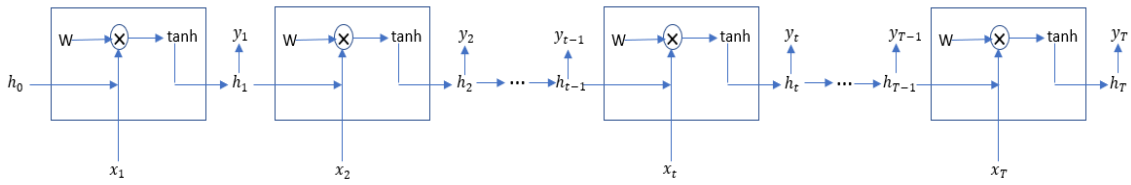

Figure 1(slide of cs231, fei-fei li & Justin Johnson & Serena Yeung) [13]

In this graph, each square where all calculations happen is called a RNN cell. By combing those cells together and the chain-like structure is the hidden layer in recurrent neural network. Other notations in the graph are explained as followed: t = 1,2,…,T will be the time step, $H = [h_0, h_1, \dots, h_T]$ is the hidden state, where h_0 is the initial hidden state, which is usually set to be 0, W is the weight matrix, tanh is the activation function, $X = [x_0, x_1, \dots, x_T]$ is the sequence of input, and $Y = [y_0, y_1, \dots, y_T]$ is the output of the network.

The equation expression of recurrent neural network can be written as followed:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h)$$

$$= \tanh\left([W_h \quad W_x \; 1]\begin{bmatrix} h_{t-1} \\ x_t \\ b_h \end{bmatrix}\right)$$

$$= \tanh\left(W\begin{bmatrix} h_{t-1} \\ x_t \\ b_h \end{bmatrix}\right)$$

$$y_t = g(W_y h_t + b_y),$$

where h_t is the hidden state, x_t is the input vector at current time step, h_{t-1} is the hidden state from the previous time step, W is the weighted matrix, b_h and b_y are the biases and g is an activation function. As moving forward through time step, the hidden state h_t will be passed into the same function and the next input x_{t+1} is read until all x_t in the sequence of input is consumed.

In the recurrent neural network, the loss of output y_t can be calculated in each time step. The loss function can be chosen differently based on types of RNN or dataset. The MSE loss is frequently used in regression loss. The final loss for the entire training step is the sum of those individual losses. The general expression of the loss step can be expresses as followed:

$$L(\hat{y}, y) = \sum_{t=1}^{T} L(\hat{y}_t, y_t)$$

The back propagate process in the recurrent neural network is similar with what have been explained before in the Artificial Neural Network section. It is called back propagation through time in RNN. The expression of the loss L with respect to weight matrix W at time step T can be expressed as followed:

$$\frac{\partial L_T}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_T}{\partial W}\Big|_t$$

Even though the RNN is powerful and can be used in a lot of different areas, there is a critical disadvantage of this type of network. When backpropagation from h_T to h_0 is used to compute gradients of loss, the transpose of W will be multiplied several times because it must go back through the whole sequence and do back propagate on every RNN cells included in the network.

If the largest singular value of W is greater than 1, the gradient on h_0 will rapidly increase in a short period and will cause exploding gradient problem. Otherwise, if the largest singular value of W is smaller than 1, the gradient on h_0 will rapidly decrease and will cause vanishing gradient problem. The gradient problem will cause difficulties in tuning parameters. In order to solve this problem, a new architecture of recurrent neural network called LSTM is designed and will be explained in the next section.

3.2.6 LSTM

Long short-term memory network (LSTM) is a type of Recurrent Neural Network (RNN) with a different architecture. This model is first introduced in 1997 by Hochreiter and Schmidhuber [14]. This model is design to avoid the vanishing and exploding gradient problem and will have better performance on long sequence.

Similar with traditional standard recurrent neural network, long short-term memory maintains a chain structure which will be recurrent at every time step. LSTM modified the architecture of RNN by adding another hidden layer at every time step that is transporting throughout the layer and kept inside the layer. This new hidden layer is called cell state which is the core idea behind LSTM. The cell state is the main reason why long short-term memory network solves the vanishing and exploring vanish problem, which will be explained later in this section when back propagation in LSTM is explained.

In RNN, the old information in memory will be refresh in every time step after recursion. Because of this structure, RNN usually does not have good performance if the sequence is long because the information from earlier time step might be forgotten. LSTM introduces three type of gates to reduce or increase information to the cell state, which are forget gate, input gate and output gate, which add flexibility to the network. Those gates

work similar as small and simple hidden layers mathematically. Those gates can control how much information to keep or remove in the network so the memory will last longer than recurrent neural network. Because of this structure of LSTM, this network will have better performance on long sequence. This is also the reason why it is called long short-term memory.

The figure of LSTM will be similar with the Figure 1 of RNN. The difference is that instead of using a RNN cell, a LSTM cell will be used to form the chain structure. The detailed figure of a single cell in long short-term memory is included in the Figure 2. The mathematical explanation will be described after explaining the figure.

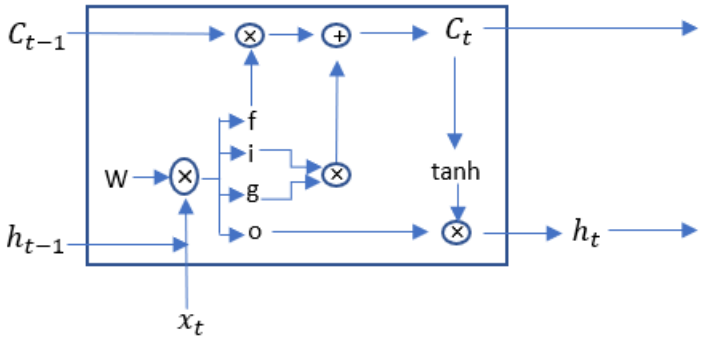

Figure 2(slide of cs231, fei-fei li & Justin Johnson & Serena Yeung)[13]

In this single cell figure, C_{t-1} is the cell state from previous time step t-1, C_t is the updated cell state at time step t, h_{t-1} represents the hidden state in previous time step t-1, h_t is the hidden state at time step t, x_t is the input vector at t time step, W is the weight matrix, f is the forget gate to decide whether to erase the cell, i is the input gate to decide if the information will be processed into the cell, g is the gate to control how much to write in

the cell state which will be used together with input gate, and o is the output gate to decide how much information in the cell state will be revealed.

The mathematical expression of this network base on the graph in a matrix form can be written as followed:

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

$$C_t = f_t \cdot C_{t-1} + i_t \times \tilde{C}_t$$

$$h_t = o_t \cdot \tanh(C_t)$$

$$y_t = \tanh(W_y h_t + b_y)$$

The more detailed explanation of each step and gates used in training a LSTM network will be shown as followed.

-forget gate

In this layer, what information will be reduced is considered. It takes the previous hidden state h_{t-1} and x_t weighted by weight w_f and then adds bias b_f. Finally, the data will be activated by sigmoid function. So, the forget gate will return a number between 0 and 1. If the number is 0, this means this information will be completely forgot. If the number is 1, it means this information is completely kept.

The expression of the forget gate is:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

-Input gate

In this layer, what new information will be stored into the cell state is included. To achieve this purpose, another tanh layer will be introduced as well. The input layer is a sigmoid layer that shows how much values should be updated. Similar to the forget gate, it

takes y_{t-1} and x_t weighted by weight w_i then adds bias b_i. The input gate returns a number between 0 and 1. If number is 0, this value is ignored, otherwise, it is updated.

The expression of the input gate is:

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

For the tanh layer, a candidate value $\tilde{C}_t$ is created. This layer takes y_{t-1} and x_t as well. It has its own weight w_c and bias b_c.

The expression of candidate value is:

$$g_t = tanh(W_c[h_{t-1}, x_t] + b_c)$$

After multiplying i_t and $\tilde{C}_t$, the updated value to cell state can be decided. By combining the forget gate and input gate, the new cell state can be written as:

$$C_t = f_t \cdot C_{t-1} + i_t \times \tilde{C}_t$$

-output gate

In this layer, output is decided. This layer decides how much of the updated cell state as explained before will become the output. This layer also takes y_{t-1} and x_t weighted by weight w_o then adds bias b_o. Finally, put it in a sigmoid function so that the layer will return a number between 0 and 1.

The expression of the output layer is:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

The final output y_t can be calculated by multiply o_t with C_t after putting it in a tanh function, which can be written as:

$$h_t = o_t \cdot \tanh(C_t)$$

In the long short-term memory, the way to compute the gradient of loss with backpropagation algorithm is similar with RNN with a small twist. Instead of flowing back through h_T to h_0, the cell state is used. The gradient of loss is computed by flowing back through C_T to C_0. As explained before, the cell state in LSTM is the transpose line at the top of the cell. Instead of multiplying weight matrix W at each time step that causes gradient problem, when backpropagation performed on LSTM, the forget gate is multiplied several times. Since the forget gate will be updated at each time step, the exploring or vanishing gradient problem caused by multiplying same number repeatedly does not happen in the LSTM model.

Chapter 4: Modeling Covid-19 DATA based on ARIMA and LSTM and Results

4.1 Dataset

The dataset used in the experiment is a daily collected dataset for Covid-19 from the COVID tracking project at Atlantic from January 2020 to February 2021 [16]. The dataset is the tracking by states with cases information, PCR tests, antibody tests information, antigen test information, hospitalization, outcomes, and state metadata. The source of the data is provided from official government sites to control the accuracy of the dataset. The data for each state is also published on the official website [21]. The tracking project updates the dataset on a daily basis and it collects data manually, rather than relying on scrapers, to make sure the data point can be recorded correctly even the data is not machine readable. Also, we only collect data that is posted publicly and freely available to make sure the dataset is usable. The following Table 1 shows a summary of the original dataset.

Data Category	Description
Date	Date from January 25, 2020 to March 07, 2021
State	The states in the U.S.
Cases	The reported covid-19 cases condition, including total confirmed cases, daily confirmed cases, and probable cases
PCR tests	The reported PCR tests condition, including total number of people took the PCR tests, the number of people took tests and return negative, the number of people took the tests and return positive, the total number of people took the test, but the result is still pending
Antibody test	The situation of the antibody tests, including total number of people took the antibody test, the number of people took tests and return negative, and the number of people took the tests and return positive
Antigen test	The situation of the antigen tests, including total number of people took the antibody test, and the number of people took tests and return positive
Hospitalization	The situation of hospital situation, including the daily increased hospitalized patients, cumulative hospitalized patients, cumulative patients who has been hospitalized in ICU, cumulative patients who has been hospitalized under advanced ventilation, currently hospitalized patients, currently

	hospitalized patients in ICU, and currently hospitalized patients under advanced ventilation
Outcomes	The death condition caused by Covid-19, the number of people who are identified as recovered, and the number of patients who discharged from the hospital

Table 1 Summary of the dataset

For this study, some preparation is made before application. In this study, we only focus on the daily counted confirmed cases in the midwestern central states. The number of daily confirmed cases from Michigan, Wisconsin, Ohio, Illinois and Indiana states are summed up and recorded. Since the first data recorded for all five states is 03/05/2020, the dataset since then is used. In this study, all the analyses and the plots are drawn by using R package 'ggplot2'.

Figure 3 shows the time series plot for the dataset. In the Figure 3, the x-axis represents the date, and the y-axis represents the daily confirmed cases.

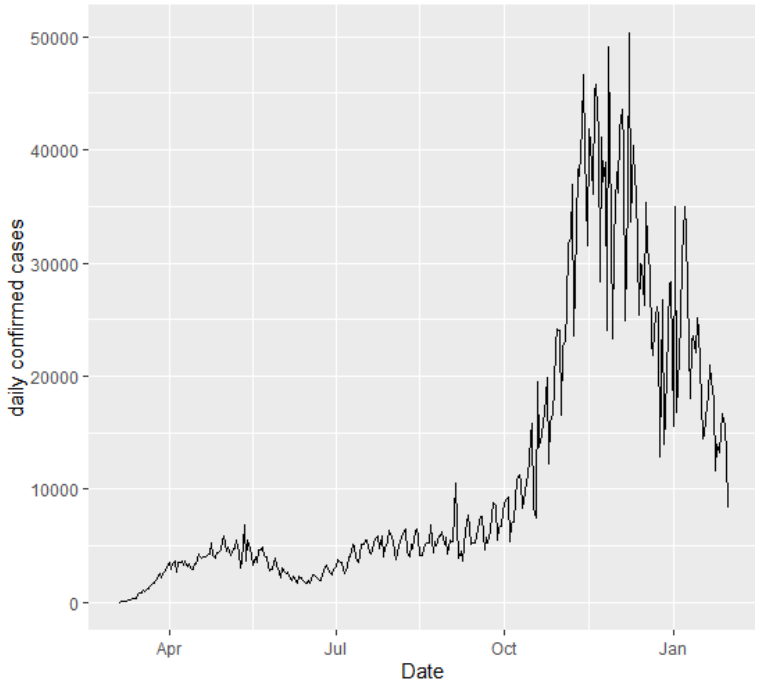

Figure 3 time series plot

4.2 ARIMA

For the ARIMA model, dataset from 03/05/2020 to 10/23/2020 is used to predict the daily confirmed cases from 10/24/2020 to 01/31/2021. In this study, 'astsa,' 'forecast,' and 'TSA' packages in R is used to implement ARIMA models.

Before applying ARIMA model, ACF plot in Figure 4 is drawn to check if the dataset is stationary and if any seasonality is involved.

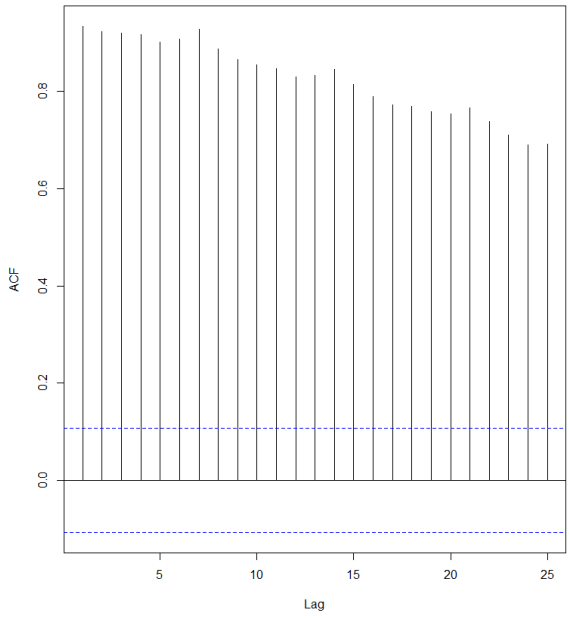

Figure 4 ACF plot.

From the plot, the acf values decrease linearly by increasing the lag. Therefore, the dataset is a non-stationary time series. A difference of order 1 is applied on the dataset to make it stationary. The ACF and PACF plot of the differences of order 1 is shown on Figure 5.

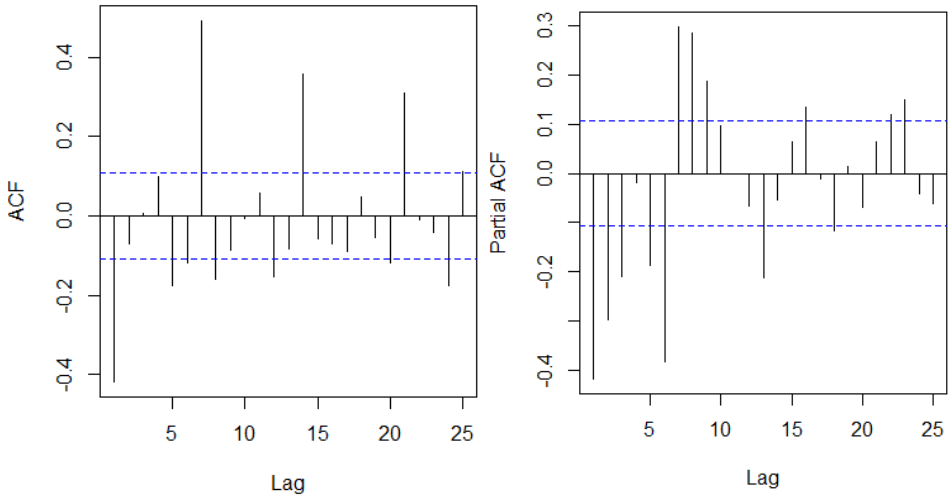

Figure 5: Autocorrelation function (ACF) plot and Partial Autocorrelation Function (PACF) plot involving differencing.

From the ACF plot after differencing, the spikes occur at lags 0, 7, 14 and 21. This shows a weekly periodicity included in the dataset so a ACF plot and PACF plot involves seasonality of 7 as shown in figure 6 and is used to decide the values of p, d, q, P, D and Q.

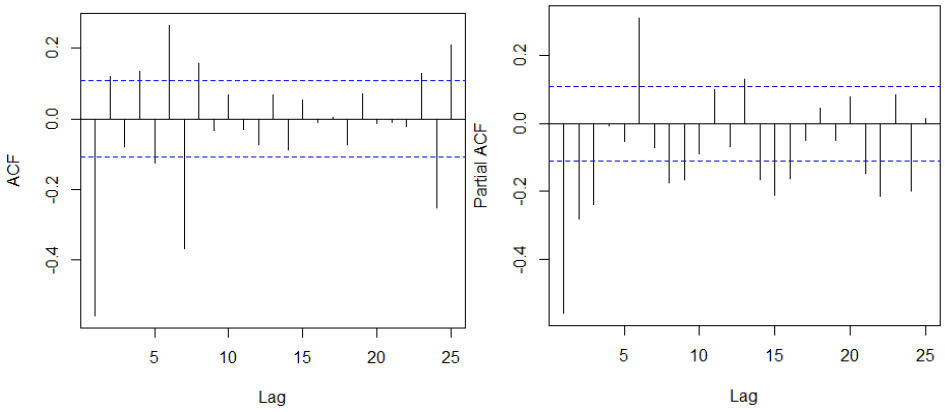

Figure 6 ACF plot and PACF plot involving seasonality.

By reading the acf plot and pacf plot, the preliminary model chosen for the dataset is ARIMA $(0,1,1)\times(0,1,1)_7$. The summary of this ARIMA model obtained by using R is shown in Table 2.

Coefficients:	**MA (1)**	**SMA (1)**			
	-0.7713	-0.5242			
Standard Error	0.0389	0.0928			
Training set error measures:					
ME	RMSE	MAE	MPE	MAPE	MASE
64.79	1020.87	571.05	0.177	12.36	0.736

Table 2 Summary of ARIMA $(0,1,1) \times (0,1,1)_7$

Then a z test is applied to test the significance of the corresponding coefficient. The result of the hypothesis test is shown in Table 3.

	Estimate	**Standard Error**	**z-value**	**p-value**
MA (1)	-0.7713	0.0389	-19.8064	$< 2.2 \times 10^{-16}$
SMA (1)	-0.5242	0.0928	-5.6517	1.59×10^{-8}

Table 3 The z test for ARIMA $(0,1,1) \times (0,1,1)_7$

The p-value for coefficient of MA and Seasonal MA is 2.2×10^{-16} and 1.59×10^{-8} respectively. Both p-values are smaller than the significant value 0.05, which means the null hypothesis can be rejected. This means that all the coefficients are significant.

After choosing the model, a Box-Ljung test was applied to make sure the residual is white noise. The output of the Box-Ljung test is shown as followed in Table 4.

X-squared	df	p-value
24.947	20	0.2035

Table 4 Box-Ljung test for ARIMA (0,1,1) × $(0,1,1)_7$

The p-value for the test is 0.2 which is higher than the significant value 0.05. This means the null hypothesis is accepted and there is no lack of fit in the model. The forecast equation for ARIMA model is written as followed:

$$\hat{y}_t = y_{t-7} + y_{t-1} - y_{t-8} + 0.77e_{t-1} + 0.52e_{t-7} + 0.4e_{t-8}$$

The result of the forecasting is shown below in Figure 7 and Table 5:

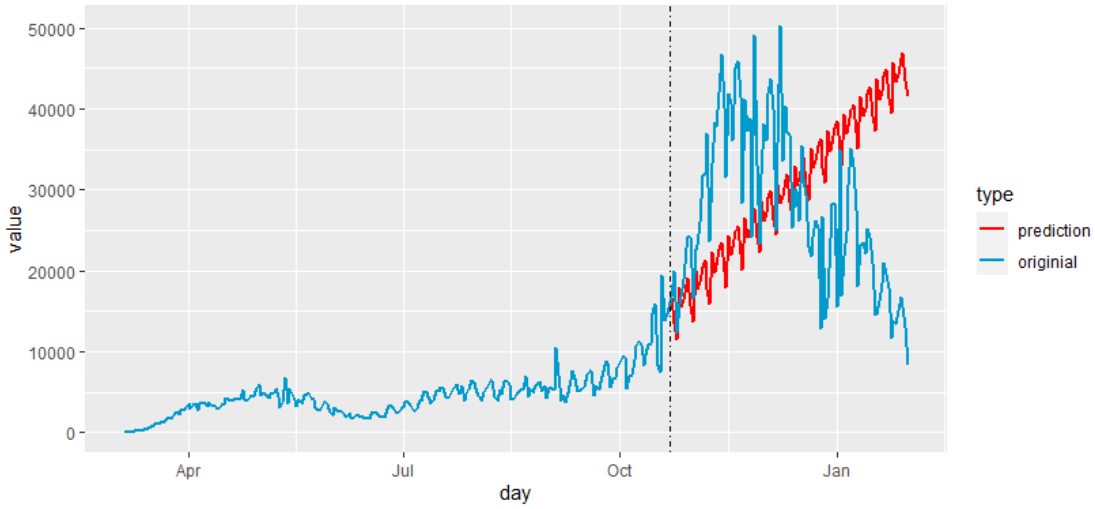

Figure 7 Prediction vs. Original Using ARIMA (0,1,1) × $(0,1,1)_7$

Data	*Original*	*Prediction*
10/24/2020	19923	13506.64
10/25/2020	12302	11544.52
10/26/2020	15887	17933.67
…	…	…
01/29/2021	15603	46966.09
01/30/2021	12994	43498.83
01/31/2021	8305	41536.71

Table 5 Prediction vs. Original Using ARIMA (0,1,1) × $(0,1,1)_7$

In Figure 7, the blue line is the original dataset, and the red line is the prediction made by using ARIMA $(0,1,1) \times (0,1,1)_7$. The black dash line separates the training dataset and the predictions.

The RMSE for this model is 15674.19. From the result presented in Table 2, the model does not have a good performance. This might be due to the change occurred in the time series between January 25, 2020 to March 07, 2021 . From Figure 3, which is the date versus daily confirmed cases plot, it shows a major change at 10/30/2020. After this date, the daily confirmed cases increase rapidly and show a totally different pattern then before.

Therefore, more recent dataset is used to do the prediction and check if it will have better performance. At this time, the dataset from 10/31/2020 to 01/31/2021 is used to pre dict the daily confirmed cases until 02/24/2021.

After repeating the process, the new fitted model is ARIMA $(1,1,1) \times (1,1,1)_7$. The ACF plot and PACF plot after involving seasonality is shown in Figure 8.

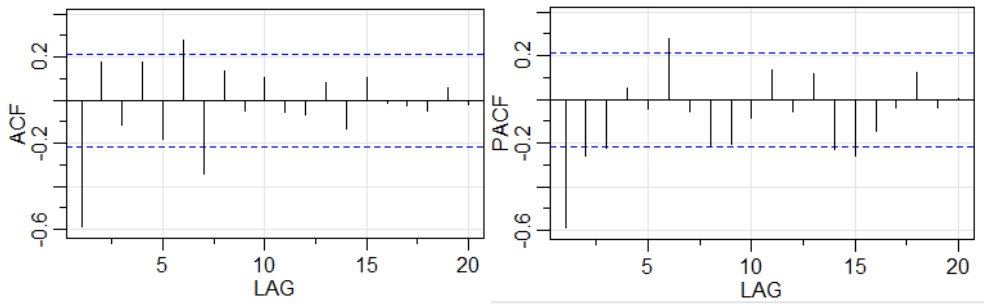

Figure 8 ACF plot and PACF plot involving seasonality

The R output of this ARIMA $(1,1,1) \times (1,1,1)_7$ is shown in Table 6.

Coefficients:	**AR (1)**	**MA (1)**	**SAR (1)**	**SMA (1)**
	-0.292	-0.4629	0.3084	-0.9999

Standard Error	0.1353	0.1102	0.1129	0.294	
Training set error measures:					
ME	RMSE	MAE	MPE	MAPE	MASE
-958.879	4691.353	3429.04	-4.941	13.6947	0.5891

Table 6 Summary of ARIMA $(1,1,1) \times (1,1,1)_7$

Then a z test is applied to check the hypothesis test for the coefficients. The results of the hypothesis test are shown below in Table 3.

	Estimate	**Standard Error**	**z-value**	**p-value**
AR (1)	-0.2919	0.1352	-2.1584	0.03
MA (1)	-0.4629	0.1102	-4.2023	2.64×10^{-5}
SAR (1)	0.3084	0.1129	2.7316	0.0063
SMA (1)	-0.9998	0.2939	-3.401	0.00067

Table 7 The z test for ARIMA $(1,1,1) \times (1,1,1)_7$

The p-values for coefficients of AR, MA and Seasonal AR and Seasonal MA are 0.03, 2.64×10^{-5}, 0.0063 and 0.00067 respectively. All p-values are smaller than the significant value 0.05, which means that the null hypothesis of 0 coefficients can be rejected. This means that the coefficients are significant.

Also, a Box-Ljung test is applied on the residual values to make sure the model capture most of the information. The result of the Box-Ljung test is shown in Table 8.

X-squared	**df**	**p-value**
19.156	20	0.5117

Table 8 Box-Ljung test for ARIMA $(1,1,1) \times (1,1,1)_7$

The p value for Box-Ljung test is 0.51, which is much larger than the significant value 0.05. This means the model does not exhibit lack of fit. The forecast equation for the new ARIMA model is written as followed:

$$\hat{y}_t = 28217 - 0.29y_{t-1} + 0.29y_{t-8} + 1.31y_{t-7} + y_{t-1} - y_{t-8} + 0.46e_{t-1} + 0.99e_{t-7} + 0.46e_{t-8}$$

The result of the prediction is shown below:

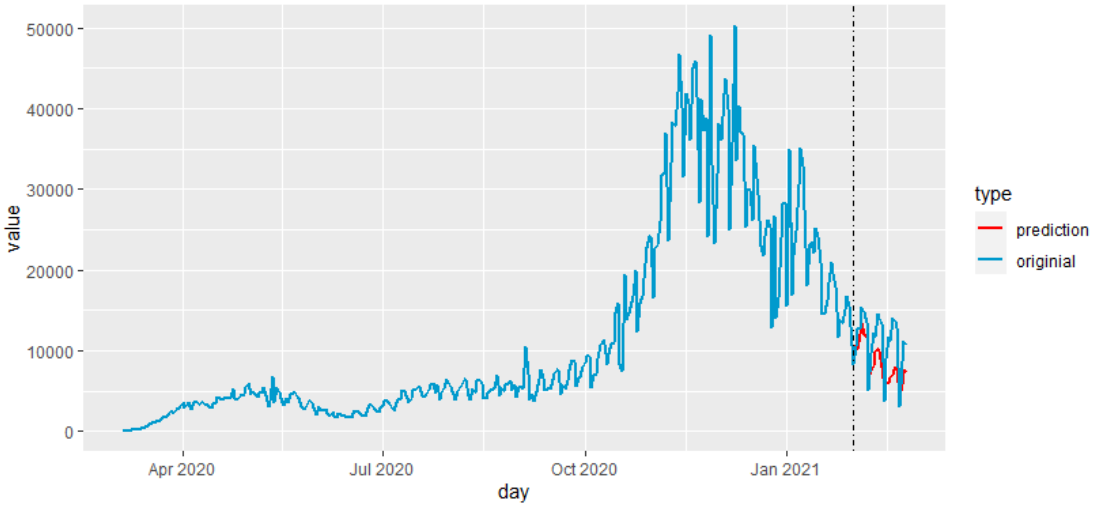

Figure 8 Prediction vs. Original Using ARIMA $(0,1,1) \times (1,1,1)_7$

Data	*Original*	*Prediction*
02/01/2021	10661	10436.13
02/02/2021	10202	12748.87
02/03/2021	11989	12708.08
…	…	…
02/22/2021	5805	8551.768
02/23/2021	7610	11152.78
02/24/2021	7269	10652.51

Table 9 Prediction vs. Original Using ARIMA $(1,1,1) \times (0,1,1)_7$

In Figure 8, the red line is the prediction line, and the blue line is the original dataset line. The vertical black dashed line separates the training dataset and prediction. The RMSE for this model is 3623.92, which is a significant improvement compared to the last model.

This change can also be discovered from the plot and table presented before. From the result, it can be discovered that the ARIMA model captures the seasonality of the dataset but does not capture the trend of the dataset very well. Also, the ARIMA makes the prediction with a larger range than the expected values.

4.3 LSTM

In order to have a fair comparison with the ARIMA model, the LSTM model takes dataset from 10/31/2020 to 01/31/2021 to make prediction on the daily confirmed cases until 02/24/2021.

Since the dataset is not a large or complicated dataset, the standard LSTM is used in this experiment to reduce the overfitting problem. Before training the dataset, some transformation of the dataset to make it acceptable in the LSTM model will be performed.

We used the packages 'keras' and 'tensorflow' to implement the LSTM network. The R code used to build and train the LSTM network is shown in Figure 9:

```
model <- keras_model_sequential()
model%>%
  layer_lstm(units,
             batch_input_shape,
             return_sequences = TRUE,
             stateful = TRUE)%>%
  layer_dropout(rate = 0.5)
model %>% compile(
  loss = 'mean_squared_error',
  optimizer = optimizer_adam(
    lr,
    beta_1 = 0.9,
    beta_2 = 0.999,
    epsilon = NULL
  ),
  metrics = c('accuracy')
)
for(i in 1:Epochs){
  model %>% fit (x = x_train_arr,
                 y = y_train_data,
                 Epochs = 1,
                 batch_size=1,
                 verbose = 1,
                 shuffle= FALSE)
  model %>% reset_states()
}
```

Figure 9 R code to build LSTM

In this study, MSE is used as the loss function since this is a univariate time series dataset. Adaptive Moment Estimation (Adam) is used as the optimizer because this is a non-stationary dataset. In the Adam optimizer, the default values of β_1 and β_2, which are 0.9 and 0.999, are used. The step size used in the optimizer is tuned in this study. The detailed method is introduced later. From the R code shown in Figure 9, in order to implement the LSTM network, units and epoch are two factors that needed to be included. In the R code, 'units' means the number of hidden nodes used in the network and 'epoch' means the number of times given to train the model. Epoch equals to 1 means the dataset is pass forward through the network and perform backward propagation ten times in this implement.

Before implementing the LSTM network as described in Figure 9, dataset need to be transformed in order to meet the requirement of the network. First, a MinMaxScaler is used to normalized dataset because LSTM is sensitive to normalization. The scaling function is expressed as followed:

$$X_{scale} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

LSTM models also expect 3-D array predictor X of size, structured as [number of samples, number of time steps, or number of features] and a 2-D array target variable Y, structured as [number of samples, number of features]. The number of samples means the number of observations in the set. This number is also called batch size. In this experiment the batch size will be defined as the size of the training dataset. The number of time step describe the method to separate time steps for given observations. In order to deal with seasonality of seven which is describe in the ARIMA part, seven-time steps will be checked to make the next prediction. The number of features is the number of predictors used in the

network. In this study, it should be 1 since this is a univariate time series dataset. The batch_input_shape required to build the network in R code, which is shown in Figure 9, is the same structure as the predictor X.

In order to decide the number of nodes and epoch used in LSTM layer, cross validation on a rolling basis is applied. First the dataset from 10/31/2020 to 01/31/2021 is divided into 5 folds. First, fold 1 to fold 4 will be used to build a model to predict for fold 5 and get the validation RMSE. Then, fold 1 to fold 3 are used with the same number of units used before to predict for fold 4 and get the validation RMSE. Repeat this process and the final validation RMSE of the specific number of units will be the average validation RMSE. The number of nodes were tested from 10 to 50, increasing 10 each time. During the cross validation, epoch of 5 and step size in optimizer 0.001 was used.

Number of units	Validation RMSE
10	8858.33
20	8516.87
30	*8333.18*
40	8350.04
50	8611.48

Table 4 Deciding Units

From the table, the number of units with smallest validation RMSE is 30. Therefore, units 30 is used to fit the model. After setting the units, the same method is applied to decide the epoch from 1 to 10, increasing 1 each time, with units equals to 30.

Epoch	Validation RMSE
1	8127.84
2	8292.74
3	8261.86
4	*8019.31*
5	8271.77
6	8428.89
7	8621.05
8	8196.92

9	8343.21
10	8347.11

Table 5 Deciding Epoch

From the table before, the number of the units 30 and epoch 4 were chosen to develop the model because they have the smallest validation RMSE. Therefore, there were totally 3840 parameters included in this network. Then the same cross validation method is used to tune the step size in the optimizer.

Step Size	**Validation RMSE**
0.0005	8666.59
0.0006	8572.24
0.0007	8207.08
0.0008	8220.80
0.0009	8260.52
0.001	*8156.20*
0.0011	8653.23
0.0012	8230.23
0.0013	8363.36
0.0014	8213.61
0.0015	8334.47

Table 6 Decide step size for optimizer

From the Table 6, the step size of 0.001 is used for the Adam optimizer because it has smaller validation RMSE. After building the network, the predict function in R was used to make the prediction by using the network built before. The R code to perform the prediction is shown in Figure 10.

```
lstm_forecast<-model %>%
  predict(x_pred_arr,batch_size)%>%.[,,1]
```

Figure 10 R code for prediction

The input value of prediction was the last 24 observation of the training set since the number of predictions is 24. The structure of x_pred_arr is as [number of samples, number of predictions, number of features]. In this study, the structure of x_pred_arr is

[1,24,1]. After making the prediction, the data was rescaled to restore the original values. The function to restore the values is shown as followed:

$$X = X_{scale}(X_{max} - X_{min}) + X_{min}$$

The result of prediction made by the network is shown as followed in Figure 11 and Table 7:

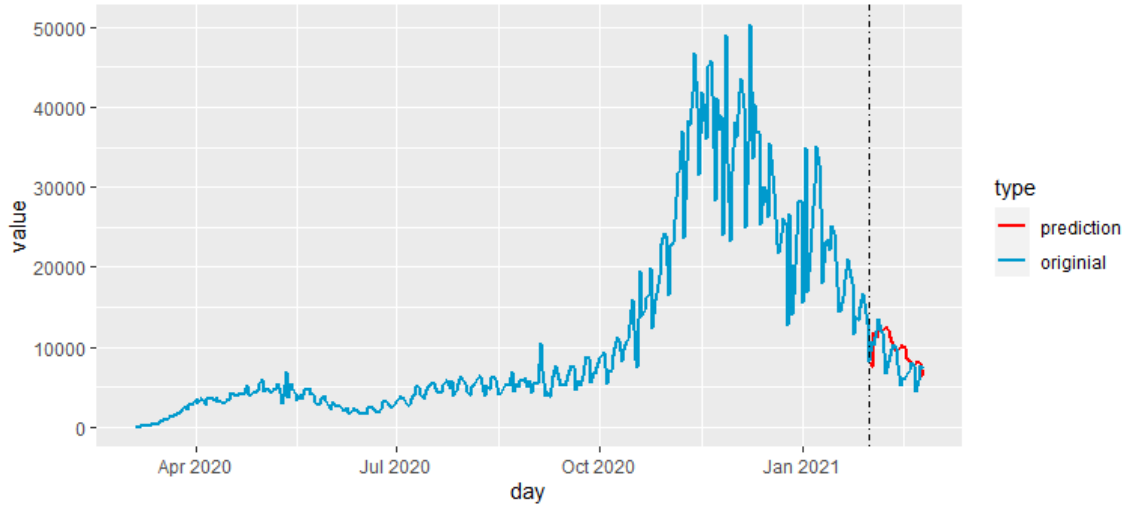

Figure 11 Prediction vs. original with units=30, epoch=4

Data	*Original*	*Prediction*
02/01/2021	10661	7417.29
02/02/2021	10202	11853.41
02/03/2021	11989	11171.981
…	…	…
02/22/2021	5805	8220.05
02/23/2021	7610	7775.37
02/24/2021	7269	6432.82

Table 7 Prediction vs. original with units=30, epoch=4

In Figure 8, the blue line is the original dataset line, and the red line is the prediction line. The vertical black dashed line separates the training dataset and the predictions. The test RMSE for this network is 2689.044. The LSTM network shows a higher prediction

number than expected values on average. Also, the network captures both the trend and the seasonality of the dataset.

Chapter 5: Conclusion and future discussion

According to the results of this study, the RMSE for ARIMA model is 3623.92 and the RMSE for the LSTM is 2689.04. Since LSMT has a smaller RMSE, LSTM has a better performance. Also, from Figure 8 and Figure 11, ARIMA has better ability to capture the seasonality and LSTM seems to have better ability to capture the trend of the dataset.

The future direction of the current research is to re-build the model with several approaches. First, in the current study, a univariate time series is used to make prediction. However, in the real life, there are many other factors that will influence the number of daily confirmed cases such as the policy published to prevent Covid-19 or traveling status. After choosing the significant factors to influence the daily confirmed cases and combining them into the prediction, the model will become more accurate and have better performance.

Another approach of the future direction is improving architecture of the models. Both ARIMA and LSTM are methods introduced long time ago. There have been several improvements on both methods. It is possible to have better performance such as weighted ARIMA and stacked LSTM. Those more modern methods might improve the performance as well.

BIBLIOGRAPHY

[1] https://www.who.int/health-topics/coronavirus#tab=tab_1

[2] https://en.wikipedia.org/wiki/COVID-19#cite_ref-WSJ-20210226_7-0

[3] https://en.wikipedia.org/wiki/Timeline_of_the_COVID-19_pandemic_in_the_United_States_(2020)

[4] Singh, S., Sundram, B. M., Rajendran, K., Law, K. B., Aris, T., Ibrahim, H., ... & Gill, B. S. (2020). Forecasting daily confirmed COVID-19 cases in Malaysia using ARIMA models. *The Journal of Infection in Developing Countries*, *14*(09), 971-976.

[5] Yudistira, N. Covid-19 growth prediction using multivariate long short term memory (2020). *arXiv preprint arXiv:2005.04809*.

[6] Kafieh, R., Arian, R., Saeedizadeh, N., Amini, Z., Serej, N. D., Minaee, S., ... & Haghjooy Javanmard, S. (2021). COVID-19 in Iran: Forecasting Pandemic Using Deep Learning. *Computational and Mathematical Methods in Medicine*, *2021*.

[7] Kumar, M., Gupta, S., Kumar, K., & Sachdeva, M. (2020). Spreading of COVID-19 in India, Italy, Japan, Spain, UK, US: A prediction using ARIMA and LSTM model. *Digital Government: Research and Practice*, *1*(4), 1-9.

[8] Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018, December). A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1394-1401). IEEE.

[9] Brockwell, P. J., Brockwell, P. J., Davis, R. A., & Davis, R. A. (2016). *Introduction to time series and forecasting*. springer.

[10] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

[11] https://brilliant.org/wiki/backpropagation/

[12] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

[13] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

[14] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[15] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

[16] https://covidtracking.com/data/download

[17] V'kovski, P., Kratzel, A., Steiner, S., Stalder, H., & Thiel, V. (2020). Coronavirus biology and replication: implications for SARS-CoV-2. *Nature Reviews Microbiology*, 1-16.

[18] Massonnaud, C., Roux, J., & Crépey, P. (2020). COVID-19: Forecasting short term hospi tal needs in France. *medrxiv*.

[19] COVID, I., & Murray, C. J. (2020). Forecasting COVID-19 impact on hospital bed-days, I CU-days, ventilator-days and deaths by US state in the next 4 months. *MedRxiv*.

[20] Anastassopoulou, C., Russo, L., Tsakris, A., & Siettos, C. (2020). Data-based analysis, modelling and forecasting of the COVID-19 outbreak. *PloS one*, *15*(3), e0230405.

[21] https://screenshots.covidtracking.com/wyoming/

[22] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.