Application and development of statistical and computational models for COVID-19 forecasting

Qian Liu[1,2,3*], Daryl L.X. Fung[1*], Leann Lac[2*]

[1]Department of Computer Science
[2]Department of Statistics
[3]Department of Biochemistry and Medical Genetics
University of Manitoba, Winnipeg, Manitoba, R3E 0W3, Canada,

Emails: QL: qianl@myumanitoba.ca; DF: lerhxind@myumanitoba.ca;
LL: lacl@myumanitoba.ca

*Equal contribution

Faculty member: Dr. Pingzhao Hu (pingzhao.hu@umanitoba.ca)

Abstract

The outbreak of 2019 novel coronavirus disease (COVID-19) has been raging around the world for almost one year. Analysis of previous data about COVID-19 might be useful to explore its epidemic regulars. Utilizing data mining and machine learning methods on COVID-19 forecasting might provide a better insight into the COVID-19. This study aimed to perform 5 months forecasting for three important indicators of COVID-19 in US, namely percentage of hospital admission that are COVID-19, number of daily confirmed COVID-19 cases, and number of death cases caused by COVID-19. Data were obtained from the website of Carnegie Mellon University Delphi Research Group. Autoregressive integrated moving average (ARIMA), echo state network (ESN), and long short-term memory (LSTM) models were used to predict the three indicators of COVID-19. In order to achieve better performance, an unsupervised time-series anomaly detection approach, matrix profile, was used to assist LSTM. In addition to matrix profile, attention mechanism was also considered to improve the LSTM. All models were evaluated using rep-holdout training and test strategy. Root mean square error (RMSE) was used as the performance metric. The matrix profile assisted attention-based LSTM model achieved relative best performance with RMSEs 1.02 for hospital admission, 43863 for daily confirmed cases, and 571 for daily death cases. Machine learning algorithms can be employed to forecast trends of COVID-19. Our forecasting can aid policymakers in prevention plan-making and guide healthcare managers to allocate health care resources reasonably.

Keywords: COVID-19 forecasting, ARIMA, ESN, LSTM, matrix profile, attention mechanism

1. Introduction

It has been more than one year since the first case of the novel coronavirus disease (COVID-19) came to light in December 2019 [1]. According to the interactive COVID-19 dashboard created and maintained by Johns Hopkins Center for Systems Science and Engineering (JHU-CSSE), COVID-19 has spread to 191 counties and caused 2,004,269 global deaths out of more than 93 million diagnosed cases by Jan 15th, 2021 [2]. COVID-19 was confirmed to be caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) as defined by the

International Committee on Taxonomy of Viruses (ICTV) [3]. SARS-CoV-2 coronavirus is a type of β-coronavirus with many potential hosts, leading to difficulties in prevention and treatment [4].

As COVID-19 is rapidly spreading and putting the world under a very distressing situation, The World Health Organization (WHO) declared COVID-19 a global pandemic in March, 2020 [5]. As a whole year's data (such as daily confirmed cases, daily hospital admission, daily death, and so on) are now available, some patterns of COVID-19 have been observed. Basically, COVID-19 follows dynamic transmission of the epidemic, with different magnitudes in terms of time, region, season, and weather, and exhibited as non-linear in nature. Since new case prevention and healthcare resource management become critical for every country, good time-series forecasting tools for COVID-19 are extremely important and necessary for estimating the number of cases in coming days.

A time-series is a collection of observations obtained sequentially in time order. To estimate the future values of the time-series is called forecasting. Time-series forecasting is a well-studied area in statistics, economics, epidemic, and so on. There is a classic time-series forecasting algorithm called autoregressive integrated moving average (ARIMA) [6], which is characterized by 3 terms: p, d, q. where, p is the order of the autoregressive term; q is the order of the moving average term; d is the number of subtracting required to make the time series stationary. ARIMA was applied for infectious disease prediction in public health [7][8]. ARIMA has also been applied to COVID-19 forecasting as early as February 2020, where only 22 time points were available at that time [9]. Ceylan et al. used ARIMA to predict the prevalence of COVID-19 for confirmed and deceased cases in Italy, Spain, and France from February 21 to April 15, 2020 [10]. Chintalapudi et al. have forecasted the number of registered and recovered cases after sixty-day lockdown in Italy by ARIMA (1,2,0) and ARIMA (3,2,0) with accuracy rate of 93.75% and 84.4%, respectively [11]. Researchers have also widely used ARIMA in comparison with other approaches in COVID-19 cases forecasting [12][13][14][15][16][17].

However, ARIMA is considered too simple to recognize complex patterns in the data. Although ARIMA is a popular and widely used model in time series forecasting, the fitted ARIMA model may not forecast accurately due to some limitations since the model only considered observations with respect to time. In principle, we can consider employing more complicated models which may include other significant variables/factors in disease prevalence. However, it is usually not possible and unreasonable to collect and involve all these factors into the forecasting models. But we could let the data "tell" us the influences of all these confounding factors. For example, unsupervised data-driven time-series anomaly detection algorithms could find significant abnormal patterns within the time series [18]. If we could incorporate the anomaly information into the forecasting models, the performance maybe increased. Matrix profile is one of these unsupervised data-driven time-series anomaly detection algorithms, and has been succeed impressively in many time-series anomalies detection problems. In 2016, Keogh et al. proposed the new data structure called matrix profile and developed a series of algorithms to detect the abnormal patterns within a time-series [19][20][21][22]. A matrix profile consists of two components: a distance vector and a profile index vector. The distance vector contains of the minimum Euclidean distances among the patterns within the time-series. The indexes of the nearest neighbours are stored in the profile index vector.

Some machine learning algorithms, such as Echo State Network (ESN) [23] and Long Short-Term Memory (LSTM) [24], have also been widely applied in time-series forecasting. ESN and LSTM all belong to Recurrent Neural Network (RNN). RNN is a broad term of neural network that has an internal memory (state) to process sequences of inputs. The output of the current input

of a RNN depends on the past ones. With the memory mechanism in it, LSTM is able to handle the time-series data very well, however, it suffers from short-term memory. If a time-series is very long, it will be difficult to pass information from earlier time steps to later ones. This is also called vanishing gradient problem [25]. ESN dose not suffer from this vanishing gradient problem, because hidden neurons in an ESN are very sparsely connected to form a network called reservoir. The weights of reservoir are randomly assigned and not trainable. Only the weights of the output are trainable. The information of earlier time points is randomly passed to the later points. No gradient backpropagation is happening within the reservoir, thus no gradient vanishing problem. Also due to the untrainable random hidden state, ESN has high computational efficiency and is good at dealing with chaotic time series data [26]. However, the untrainable random hidden state also reduced the power of the modeling, because the fact that only the output layer is trainable reduced the complexity of the model weights. While LSTM reduces the short-memory problem by making it easier to pass previous information throughout the state sequences, which still has some gates (input gate, forget gate, control gate, and output gate) to regulate the information flow [24].

However, the standard LSTM cannot detect which is the important time point for future prediction. LSTM system usually is constrained by the model structure, wich means all input time-series are forced to be encoded to a fixed length which is believed to limit the performance. Recently, the attention mechanism was proposed to overcome this limitation that allows the network to learn where to pay attention in the input series [27]. This is achieved by keeping the intermediate information from the LSTM units and training the model to learn to pay selective attention to the inputs and relate them to items in the output time-series [28]. Attention mechanism increases the computational burden but results in a more targeted and better-performing model. In addition, the model is also able to show how attention is paid to the input time-series when predicting the output. This can increase the expansibility of the LSTM model which is an important characteristic of gaining trust from end-users.

In this study, we aim to develop a matrix profile guided attention LSTM model for forecasting COVID-19 cases, including the percentage of hospital admission, the daily diagnosed cases, and the daily death cases in the United States (US).

2. Data

In this COVID-19 data analysis, we consider the data set of number of cases in United States from March 01, 2020 to January 07, 2021 [29]. In this project, we focus on three indicators: adjusted percentage of daily admitted hospitalized cases (adjusted.cases), daily confirmed cases (confirmed.cases), and daily death cases (death.cases). The adjusted percentage of daily admitted hospitalized cases is the estimated percentage of new hospital admissions with COVID-19. It is based on insurance claims data from health system partners, and smoothed in time using a Gaussian linear smoother. The data was were obtained from the website of Carnegie Mellon University Delphi Research Group.

3. Method

3.1 Data preprocessing and remaping

To improve model performance and consider the consistency in evaluating model performance between statistical and machine learning approaches, preprocessing raw data by normalization is necessary. The full data were first separated into training and testing sets. We then applied z-normalization or standardization to the training set. The formulation to transform observed raw data into z-score is $Z_i = (Y_i - \bar{Y})/s$, where $\bar{Y}$ and s are sample mean and standard

deviation. After building the model and obtaining predicted values, we remapped these values to the observed raw data scale by applying the formula: $Y_{i = +}s*Z_i + \bar{Y}$.

3.2 Matrix Profile for time-series data analysis

Matrix profile of the three indicators are calculated using Python package "matrixprofile" [30]. Several algorithms are provided by the package for computing the matrix profile, such as Scalable Time series Anytime Matrix Profile (STAMP), Scalable Time series Ordered-search Matrix Profile (STOMP). We selected STOMP function since it is faster. The window size is set as 7 (weekly anomaly). After the matrix profiles are calculated, top 10 discords are visualized. In addition to matrix profile, we added additional information from the matrix profile index that can be useful additional information for the model to learn --- global position matrix profile and relative position matrix profile. Global position matrix profile is the nearest neighbor of the closest pattern based on the current timestep. For instance, if the nearest neighbor of the current timestep is at the 15^{th} location, the value of this matrix profile will be 15. Relative position matrix profile is the relative nearest neighbour of the closest pattern based on the current timestep. If the current timestep is at 10^{th} location and the nearest neighbor is at the 15^{th} location, then the relative matrix profile will have a value of +5. The distances profiles and profile indexes are all passed to the later LSTM multivariable forecasting.

3.3 ARIMA

Nonseasonal ARIMA(p,d,q) is a generalized form of autoregressive moving average models ARMA(p,q) which is a combination of autoregression model of order p, AR(p) and moving average model of order q, MA(q). However, ARMA is designated for stationary data while time series data may not always be stationary. Therefore, to overcome this limitation, the integrated part of ARIMA models provide initially differencing process to transform non-stationary data to stationary ones.

Let y_t denote the d^{th} difference of Y_t, and Y_t refers to the observation at time t, the general equation of ARIMA(p,d,q) models are as follows.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \quad (1)$$

where $\phi = [\phi_1, \phi_2, \ldots, \phi_p]$ and $\theta = [\theta_1, \theta_2, \ldots \theta_q]$ are parameters of AR(p) and MA(q), respectively. Here, c is a constant and ε_t is the residual assumed to be uncorrelated in the final selected ARIMA model.

We built a statistical ARIMA model as baseline model to forecast the daily number of cases for each indicator. ARIMA is flexible for both stationary and non-stationary data, and the modeling procedure are described step-by-step as follows.

1. Check stationarity of the raw observed time series data. Characteristics of stationarity can be identified by time series plot and Augmented Dickey-Fuller (ADF) test with alternative hypothesis is for stationarity. Large p-value from ADF test indicates the time series data are not stationary which means data transformation is required.
2. Transform data by taking differences until data are stationary. Here, number of differencing d should be identified.
3. Determine orders p and q of AR and MA From autocorrelation function (ACF) and partial autocorrelation function (PACF) plots, respectively.
4. Select the good model based on lowest value of AIC.
5. Examine residuals of the selected model from ACF plot of residuals. If residuals look like white noise, the fitted model can be used for forecasting.

When values of d, p, q are identified in steps 2 and 3, we can obtain the estimates of c, $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. Classical regression parameter estimation approaches are maximum likelihood estimation (MLE) and least square estimation (LSE). To estimate parameters in ARIMA, we can use MLE, LSE, more details can be found in Box et al. (2015) [6].

Hyndman & Khandakar proposed Hyndman – Khandakar algorithm for automatic ARIMA modeling to select the best model and identify the optimal values of p, d, q based on smallest Akaike's information criterion (AIC) value [31]. This algorithm is summarized to R function auto.arima() in "forecast" package.

Statistical approach using ARMIA models for this COVID-19 data analysis uses R and designed to use R built-in packages for forecasting "forecast", "ffp2" and "tseries".

3.4 ESN

ESN has an input layer, a reservoir and an output layer. Considering the input layer has R input units, the reservoir has N neurons, and the output layer has L neurons, then at time step t (total training time step is T), the states of these layers and their updates are as follows.

$$X_t = [x_{1t}, x_{2t}, \ldots, x_{Rt}]' \quad (2)$$

$$H_t = [h_{1t}, h_{2t}, \ldots, h_{Nt}]' \quad (3)$$

$$Y_t^{predict} = [y_{1t}^{predict}, y_{2t}^{predict}, \ldots, y_{Lt}^{predict}]' \quad (4)$$

$$H_{t+1} = f\,(W^{in}X_{t+1} + WH_t + W^{back}Y_t^{predict}) \quad (5)$$

$$Y_{t+1} = g\,(W^{out}(X_{t+1}, H_{t+1}), Y_t^{predict}) \quad (6)$$

Where X_t, H_t, Y_t are input layer, hidden layer, and output layer at time point t, respectively. And *f* and *g* are activation functions. There are four weight matrices, W^{in}, W, W^{back}, and W^{out}, and the first three weight matrices are randomly initialized and fixed in the training step, so only W^{out} requires to be updated.

ESN could be updated using gradient descent approach. This kind of ESN is called ESN-GD. Squared error can be used as the lost function (7). And the readout weights can be updated with the gradient of the loss function with respect to the current weights (8).

$$L = (Y_t^{predict} - Y_t)^2 \quad (7)$$

$$W_t^{out} = W_{t-1}^{out} - \eta \frac{\partial L}{\partial W_{t-1}^{out}} \quad (8)$$

Where η is the learning rate to control the magnitude of the updating step. The model was built and trained using Python. Hyperparameters were finetuned manually.

3.5 LSTM

3.5.1 LSTM without attention

LSTM contains a forget gate, an input gate, and an output gate. The gates control how much of the information that is allowed into the next sequence of inputs. They are controlled using a sigmoid layer that outputs value between 0 and 1. They also receive the previous timestep of the hidden state and the cell state and to determine the next state. The first gate is the forget gate, it receives the hidden state from the previous timestep and concatenates with the input in the current timestamp, then passes into a linear layer with a sigmoid activation.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_t) \quad (8)$$

f_t refers to the output of the forget gate. $[h_{t-1}, x_t]$ is the concatenation between the input of the current timestep, x_t, and the hidden state of the previous timestep, h_{t-1}. W_f is the weight of the forget gate and b_t is the bias of the forget gate.

In the input gate, it will control how much of a new information will be passed into the current timestep, the equation is as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_t) \quad (9)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (10)$$

i_t is the output of the input gate and $\tilde{C}$ will be used to update the cell state of the current timestep.

The new cell state will be updated as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (11)$$

The forget gate will control how much of the previous cell state into the current cell state and the input gate will control how much of the new pre-cell-state into the current cell state.

The output gate will control and filter the information from the cell state. The equation is:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (12)$$

$$h_t = o_t * \tanh(C_t) \quad (13)$$

3.5.2 LSTM with attention

Motivated by the idea of attention, we added an attention mechanism to our LSTM to pay attend to important parts of the sequence to generate more insight into predicting and learning future forecast values. The proposed overall workflow could be found in **Figure 1**. As there exist outliers in the forecast values, attention mechanism can help LSTM models to focus on important parts of the sequence to prevent getting skewed values. We utilize multi-headed attention where there are several heads and each head contains a query, key and a value. Multi-headed attention has been shown beneficial with different learned linear projections [25]. The equation for the attention is:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (14)$$

Where Q is the query, K is the key, and V is the value. d_k is the dimension of the current layer. Each head will contain the attention equation:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (15)$$

The multi-headed attention will concatenate the attentions of the heads and output the combination of the attentions of the head like so:

$$MultiHead(Q, K, V) = Concat([head_1, head_2, \dots, head_k])W^O \quad (16)$$

3.5.3 LSTM models compared in the study

Given the combination of the different features, matrix profiles and the attention mechanism, we evaluated different LSTM models for each of the three indicators used in the study. Due to space limitation, we only discussed below LSTM related models:

1) *LSTM raw mo*del: using only raw normalized case data feature;
2) *LSTM raw attention model*: using only raw normalized case data feature with attention mechanism;
3) *LSTM raw matrix attention*: using raw normalized case data feature and matrix profile feature fed into the LSTM attention network;
4) *LSTM raw relative attention*: using raw normalized case data feature and the relative position of the matrix profile feature fed into the LSTM attention network.

4. Performance evaluation

After building the models for each indicator, it is important to evaluate the prediction accuracy and compare the forecasting performance to other proposed models in forecasting the number of

cases in COVID-19 data analysis. Traditional K-fold cross validation is designated for independent data. However, time series data is considered as dependent data in which we used past events to forecast the future ones. Therefore, we consider rep-holdout cross validation methods. In this method, we divide the time series data into training and testing sets by ascending time order. To conduct the model performance evaluation, we apply rep-holdout strategy in which the test sets will be the last (most recent) 10%, 20%, 30%, 40%, and 50% of all time points. We can measure forecast accuracy by root mean square error (RMSE) using the test set.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(y_t^{predict} - y_t)}{n}} \quad (17)$$

where n is the sample size for each testing set.

5. Results

5.1 Analysis of ARIMA model results

In data analysis, we consider R function auto.arima() from "forecast" package to estimate the model parameters and select the best model.

To evaluate the performance of the model, the selected ARIMA model for each holdout of each indicators and its measure of errors (overall RMSE for each holdout) are summarized in **Table 1**. For each indicator, we divided the data into training and testing sets with different proportions for different hold-out strategies. Since the training set was used to build the model, we need to pre-examine its stationarity. If the data is non-stationary, data transformation by d differencing will be considered. As we can see from **Table 1**, not all training sets required differencing (d=0).

While rep-holdout strategy (last 50% time points for testing set) outperformed among five strategies in hospital admission with lowest RMSE 1.92, and in death cases with lowest RMSE 604, the first rep-holdout strategy (last 10% time points for testing) gave the best performance in confirmed cases with RMSE = 41880. The overall performance across the 5 rep-holdouts is 3.19, 72484 and 786 for percentage of COVID-19 hospital admission, and daily death cases, respectively.

These values indicate that ARIMA models may be reasonably accurate in forecasting hospital admission and death cases, but weak and inaccurate in forecasting confirmed cases for the dataset.

5.2 Analysis of ESN model results

The results of the ESN_GD model could be found in **Table 1**. Each holdout has different performance, and the best one is observed in rep-holdout 3(30%), which has RMSE 1.47 and 51157 for hospital admission and the confirmed cases, respectively. The best RMSE for the death case is 468 observed in rep-holdout 2 (20%). The overall performance of RMSE across the 5 rep-holdouts is 1.95, 60961 and 587 for percentage of COVID-19 hospital admission, and daily death cases, respectively.

In conclusion, compared with the ARIMA model, the ESN models showed overall better performances on percentage of COVID-19 hospital admission, and daily death cases.

5.3 Analysis of LSTM model results

The result of the LSTM models can be found in **Table 1**. Comparing the LSTM model without attention mechanism (Model: raw), the LSTM model with attention mechanism (Model: raw attention) have achieved better overall performance in predicting the daily confirmed cases and daily death cases).

In order to further improve the performance of the LSTM model, we added attention mechanism to the LSTM such that the LSTM will be able to focus on important time points in the data. We achieve the best overall performance when LSTM is added with attention mechanism with the addition of matrix profile that is fed as an additional input feature (Model: raw matrix attention). It achieves the best overall performance in predicting the percentage of new hospital admission and the total daily confirmed cases with RMSE value of 1.02 and 43863, and the second-best overall performance in predicting the total daily death cases with RMSE value of 571, respectively.

Overall, the LSTM with attention mechanism fed together with the matrix profile outperforms the ARIMA model, the ESN models and the LSTM without the attention mechanism, achieving the best performance for hospital admission and confirmed cases. It is also able to achieve the second-best performance in predicting the total daily death cases.

6. COVID-19 case forecasting

After the models' performances are evaluated and the hyperparameters are fine-tuned using the rep-holdout strategy, the final prediction models for each indicator were build using the whole data (March 1, 2020 – January 7, 2021), which was used to forecast the future data between January 8 and May 31st, 2021 (**Figure 2**). Note: we can only access the data up to January 7, 2021 from the data download website at the time we submit the report. The overall forecasting trend by the LSTM raw matrix attention model is that the percentage of COVID-19 hospital admission and daily death cases will be decreasing but the number of confirmed cases will be still kept at a relatively high-rate level. For the ARIMA model, the forecasting trend is that the percentage of COVID-19 hospital admission and daily death cases will be still kept at a relatively high-rate level but the number of confirmed cases will be continuing increasing. The forecasting trend of the ESN model is between the trends of the two models.

7. Discussion and conclusion

ARIMA and ESN models, which are the baselines of this study, are able to achieve descent prediction results of the three indicators of COVID-19. The proposed LSTM combining matrix profile and attention mechanism achieved the best performance (**Table 1**). Different number of timepoints are assigned to train set according to the rep-holdout strategy. According to **Table 1**, the model performances are not associated with the size of the train sets, which means a larger train set may not guarantee a better performance. For example, the best performance of ARIMA model to predict the percentage of new hospital admission (RMSE = 1.92) is obtained when it is trained by only half of the observed data (which is the shortest training time-series) with rep-holdout 5 (50%).

Since the trend of COVID-19 cases follows a seasonal pattern, and current models, especially the ESN model, are not able to capture well the seasonal pattern. There are a lot of studies about modeling the seasonality of time series, such as seasonal ARIMA [32], seasonal ESN [33], seasonal LSTM [34] and so on. If we could involve the seasonality into our modeling, the forecasting performance might be improved further.

In conclusion, several forecasting methods have been applied and developed to model the possible percentage of COVID-19 hospital admission, cases of daily confirmed COVID-19, and cases of daily death because of COVID-19 in US, which were applied to forecast the cases for the next 144 days (Until May 31st, 2021). The proposed matrix profile combining with attention-based

LSTM model performed best among these methods. The forecasted data may guide society and policy prevention systems to control the consequences of COVID-19.

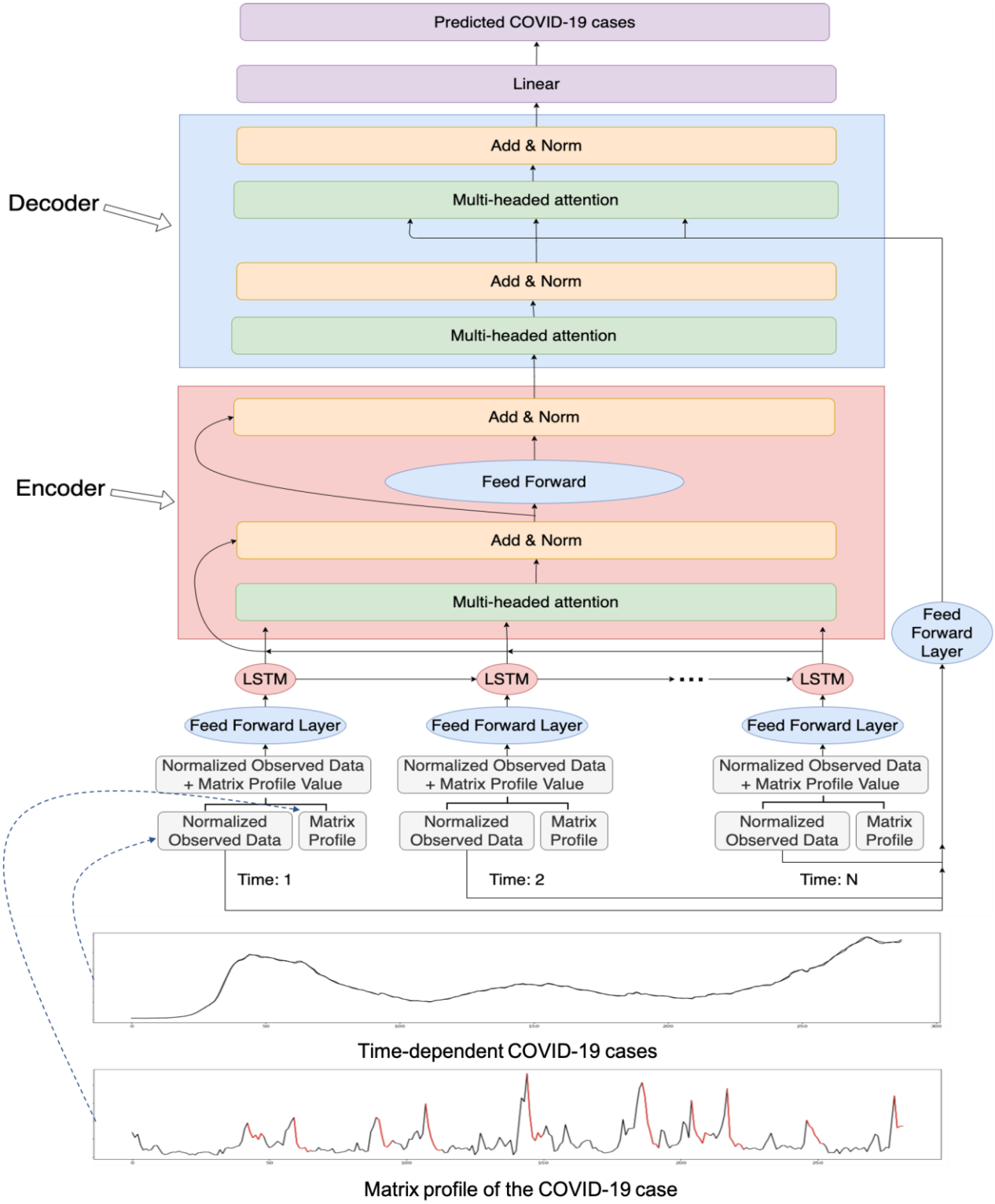

Figure 1. The overall workflow of the proposed Matrix profile attention LSTM model.

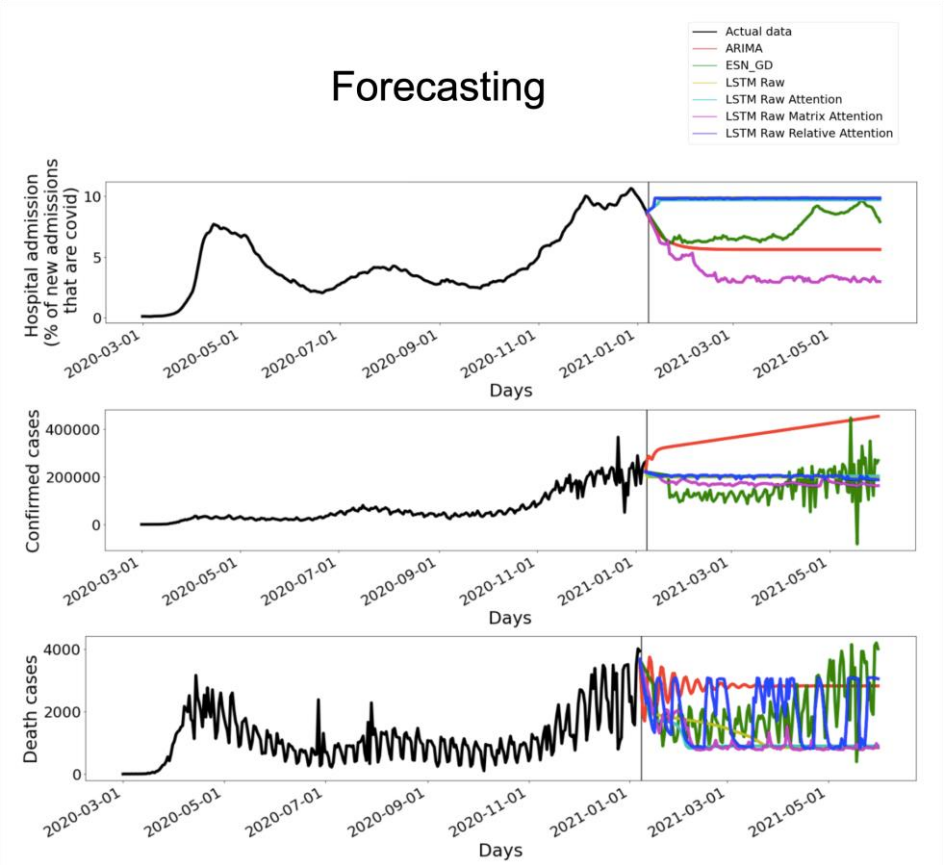

Figure 2. The forecasting results of the three COVID-19 indicators by six selected models.

Table 1. Model performance

Model		Indicator	Admission						Confirmed						Death					
		holdout	1	2	3	4	5	overall	1	2	3	4	5	overall	1	2	3	4	5	overall
ARIMA		p	4	2	2	0	1	NA	3	2	2	3	4	NA	4	4	1	4	5	NA
		d	0	0	0	2	2	NA	1	1	1	1	1	NA	1	1	0	1	1	NA
		q	2	1	1	1	0	NA	1	2	2	2	2	NA	2	2	2	2	2	NA
		RMSE	3.27	3.92	3.44	3.05	1.92	3.19	41880	98822	98005	53275	48031	72484	1058	876	679	612	604	786
ESN		RMSE	2.15	1.59	1.47	2.12	2.44	1.95	73332	58036	51157	55794	60961	59856	507	468	572	677	709	587
LSTM	Raw	RMSE	1.58	1.66	0.93	1.01	0.76	1.19	72073	93924	14498	33079	29027	48520	1106	571	523	559	407	633
	Raw relative attention	RMSE	0.49	2.36	0.53	1.16	0.66	**1.04**	64569	119964	22137	9504	8702	44975	987	780	301	350	344	**552**
	Raw matrix attention	RMSE	0.55	2.18	0.54	1.18	0.63	**1.02**	64803	114132	21391	9989	9001	**43863**	1090	762	309	345	347	**571**
	Raw attention	RMSE	1.45	2.34	0.51	1.01	0.81	1.22	71941	109818	14829	17495	13915	45600	1004	675	364	408	344	559

Reference

[1] Disease outbreak news, WHO | Novel Coronavirus – China, WHO. (2020). https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/ (accessed September 22, 2020).

[2] E. Dong, H. Du, L. Gardner, An interactive web-based dashboard to track COVID-19 in real time, Lancet Infect. Dis. 20 (2020) 533–534. https://doi.org/10.1016/S1473-3099(20)30120-1.

[3] A.E. Gorbalenya, S.C. Baker, R.S. Baric, R.J. De Groot, A.A. Gulyaeva, B.L. Haagmans, C. Lauber, A.M. Leontovich, The species and its viruses – a statement of the Coronavirus Study Group, Biorxiv (Cold Spring Harb. Lab. (2020) 1–15. https://www.biorxiv.org/content/10.1101/2020.02.07.937862v1.full.

[4] B. Vellingiri, K. Jayaramayya, M. Iyer, A. Narayanasamy, V. Govindasamy, B. Giridharan, S. Ganesan, A. Venugopal, D. Venkatesan, H. Ganesan, K. Rajagopalan, P.K.S.M. Rahman, S.G. Cho, N.S. Kumar, M.D. Subramaniam, COVID-19: A promising cure for the global panic, Sci. Total Environ. 725 (2020) 138277. https://doi.org/10.1016/j.scitotenv.2020.138277.

[5] D. Cucinotta, M. Vanelli, WHO declares COVID-19 a pandemic, Acta Biomed. 91 (2020) 157–160. https://doi.org/10.23750/abm.v91i1.9397.

[6] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, Time Series Analysis: Forecasting and Control, J. Mark. Res. 14 (1977) 269. https://doi.org/10.2307/3150485.

[7] S.H. Heisterkamp, A.L.M. Dekkers, J.C.M. Heijne, Automated detection of infectious disease outbreaks: hierarchical time series models, Stat. Med. 25 (2006) 4179–4196. https://doi.org/10.1002/sim.2674.

[8] K. CHOI, S.B. THACKER, AN EVALUATION OF INFLUENZA MORTALITY SURVEILLANCE, 1962–1979, Am. J. Epidemiol. 113 (1981) 215–226. https://doi.org/10.1093/oxfordjournals.aje.a113090.

[9] D. Benvenuto, M. Giovanetti, L. Vassallo, S. Angeletti, M. Ciccozzi, Application of the ARIMA model on the COVID-2019 epidemic dataset, Data Br. 29 (2020) 105340. https://doi.org/10.1016/j.dib.2020.105340.

[10] Z. Ceylan, Estimation of COVID-19 prevalence in Italy, Spain, and France, Sci. Total Environ. 729 (2020) 138817. https://doi.org/10.1016/j.scitotenv.2020.138817.

[11] N. Chintalapudi, G. Battineni, F. Amenta, COVID-19 virus outbreak forecasting of registered and recovered cases after sixty day lockdown in Italy: A data driven model approach, J. Microbiol. Immunol. Infect. 53 (2020) 396–403. https://doi.org/10.1016/j.jmii.2020.04.004.

[12] S.I. Alzahrani, I.A. Aljamaan, E.A. Al-Fakih, Forecasting the spread of the COVID-19 pandemic in Saudi Arabia using ARIMA prediction model under current public health interventions, J. Infect. Public Health. 13 (2020) 914–919. https://doi.org/10.1016/j.jiph.2020.06.001.

[13] V. Chaurasia, S. Pal, COVID-19 Pandemic: ARIMA and Regression Model-Based Worldwide Death Cases Predictions, SN Comput. Sci. 1 (2020) 288. https://doi.org/10.1007/s42979-020-00298-6.

[14] V. Chaurasia, S. Pal, Application of machine learning time series analysis for prediction COVID-19 pandemic, Res. Biomed. Eng. (2020) 1–13. https://doi.org/10.1007/s42600-020-00105-4.

[15] A. Hernandez-Matamoros, H. Fujita, T. Hayashi, H. Perez-Meana, Forecasting of

COVID19 per regions using ARIMA models and polynomial functions, Appl. Soft Comput. J. 96 (2020) 106610. https://doi.org/10.1016/j.asoc.2020.106610.
[16] A.K. Sahai, N. Rath, V. Sood, M.P. Singh, ARIMA modelling & forecasting of COVID-19 in top five affected countries, Diabetes Metab. Syndr. Clin. Res. Rev. 14 (2020) 1419–1427. https://doi.org/10.1016/j.dsx.2020.07.042.
[17] Y. Wang, C. Xu, S. Yao, Y. Zhao, Forecasting the epidemiological trends of COVID-19 prevalence and mortality using the advanced α-Sutte Indicator, Epidemiol. Infect. 148 (2020). https://doi.org/10.1017/S095026882000237X.
[18] V. Chandola, A. Banerjee, V. Kumar, Anomaly Detection : A Survey, ACM Comput. Surv. 41 (2009) 1–58.
[19] C.-C.M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H.A. Dau, D.F. Silva, A. Mueen, E. Keogh, Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets, in: Institute of Electrical and Electronics Engineers (IEEE), 2017: pp. 1317–1322. https://doi.org/10.1109/icdm.2016.0179.
[20] Y. Zhu, Z. Zimmerman, N.S. Senobari, C.-C.M. Yeh, G. Funning, A. Mueen, P. Brisk, E. Keogh, Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins, in: Institute of Electrical and Electronics Engineers (IEEE), 2017: pp. 739–748. https://doi.org/10.1109/icdm.2016.0085.
[21] C.-C.M. Yeh, H. Van Herle, E. Keogh, Matrix Profile III: The Matrix Profile Allows Visualization of Salient Subsequences in Massive Time Series, in: Institute of Electrical and Electronics Engineers (IEEE), 2017: pp. 579–588. https://doi.org/10.1109/icdm.2016.0069.
[22] C.C.M. Yeh, N. Kavantzas, E. Keogh, Matrix profile IV: Using Weakly labeled time series to predict outcomes, Proc. VLDB Endow. 10 (2017) 1802–1812. https://doi.org/10.14778/3137765.3137784.
[23] H. Jaeger, H. Haas, Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication, Science (80-.). 304 (2004) 78–80. https://doi.org/10.1126/science.1091277.
[24] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (1997) 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
[25] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, Int. J. Uncertainty, Fuzziness Knowlege-Based Syst. 6 (1998) 107–116. https://doi.org/10.1142/S0218488598000094.
[26] M.C. Oztuik, D. Xu, J.C. Principe, Analysis and design of echo state networks, Neural Comput. 19 (2007) 111–138. https://doi.org/10.1162/neco.2007.19.1.111.
[27] Y. Wang, M. Huang, L. Zhao, X. Zhu, Attention-based LSTM for aspect-level sentiment classification, EMNLP 2016 - Conf. Empir. Methods Nat. Lang. Process. Proc. (2016) 606–615. https://doi.org/10.18653/v1/d16-1058.
[28] J. Xu, T. Yao, Y. Zhang, T. Mei, Learning multimodal attention LSTM networks for video captioning, MM 2017 - Proc. 2017 ACM Multimed. Conf. (2017) 537–545. https://doi.org/10.1145/3123266.3123448.
[29] D.C. Farrow, L.C. Brooks, A. Rumack, R.J. Tibshirani, R. Rosenfeld, Delphi Epidata API, (2021).
[30] A. Van Benschoten, A. Ouyang, F. Bischoff, T. Marrs, MPA: a novel cross-language API for time series analysis, J. Open Source Softw. 5 (2020) 2179.

https://doi.org/10.21105/joss.02179.
[31] R.J. Hyndman, Y. Khandakar, Automatic time series forecasting: The forecast package for R, J. Stat. Softw. 27 (2008) 1–22. https://doi.org/10.18637/jss.v027.i03.
[32] W.B. M., H.L. A., Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results, J. Transp. Eng. 129 (2003) 664–672. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664).
[33] H. Siqueira, L. Boccato, R. Attux, C. Lyra, Echo State Networks and Extreme Learning Machines: A Comparative Study on Seasonal Streamflow Series Prediction, in: T. Huang, Z. Zeng, C. Li, C.S. Leung (Eds.), Neural Inf. Process., Springer Berlin Heidelberg, Berlin, Heidelberg, 2012: pp. 491–500.
[34] K. Bandara, C. Bergmeir, H. Hewamalage, LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns, ArXiv. (2019) 1–14. https://doi.org/10.1109/tnnls.2020.2985720.