Paper:

A comparative study of statistical models and univariate and multivariate RNN for forecasting the number of COVID-19 cases

Naoki Dohi, Namal Rathnayake, Yukinobu Hoshino

185 Miyanokuchi, Tosayamada, Kami City, Kochi 782-8502, JAPAN
E-mail: 255055c@gs.kochi-tech.ac.jp
[Received 30/06/22; accepted 22/07/22]

Abstract. This study forecast COVID-19 cases in Japan to assist in decisions. There are statistical and machine learning models for forecasting time series. Traditionally, Statistical models are used in this field, and many machine learning models are recently adopted. Therefore, both models are needed to be compared for forecasting COVID-19 cases. Auto-Regressive Integrated Moving Average (ARIMA) and Recurrent neural network (RNN) are compared by Akaike information criterion(AIC) and Root Mean Square Error (RMSE). Results show that RNN(with vector inputs and outputs) was better than statistical models. Specifically, the RMSE of RNN is 47 and that of ARIMA is 103.

Keywords: Statistical Model, SARIMA, Machine Learning, Deep Learning, RNN, Time Series Forecasting, COVID-19, Novel Corona Virus

1. Introduction

1.1. Background

Within a couple of years, SARS-CoV-2 has been influenced worldwide. According to World Health Organization (WHO), weekly reports [1], confirmed cases are over 514 million, and deaths are over 6 million globally.

Therefore, it is necessary to create a forecasting model for public health, early availability of hospital beds, vaccines, early administrative decision-making on like declaring a state of emergency, and etc. This affects many fields such as people's physical and mental health and social economy.

1.2. Purpose

This paper investigates Recurrent Neural Network(RNN) to minimize prediction error in forecasting the number of COVID-19 cases. Thus, three types of input-output relation's, one dimension input and one dimension output, three dimension input and three dimension output, and three dimension input and one dimension output, RNN is compared. Then compare three types range for scaling with multivariable inputs.

1.3. Related work

A Wu et al.[2] study compares statistical and neural networks in the plague forecasting field. In this study, forecast the ratio of the number of influenza-like illnesses in each country in the last four weeks to the previous week with a feature in the last ten weeks by using Transformer [3] and data sets from the Centers for Disease Control and Prevention (CDC). They compare Transformer, ARIMA, Long short-term memory (LSTM) [4], Sequence to Sequence with Attention. Note that they set the RMSE of ARIMA as the baseline for comparison. As a result of their paper, Transfomer most reduces RMSE from ARIMA Relatively.

Spyros et al.[5] have also verified and compared the prediction accuracy of both statistical models and machine learning with data from M3 competition; machine learning models such as the RNN and other machine learning models to verify if they can replace statistical models such as Auto-Regressive Integrated Moving Average (ARIMA) that have been used for a long time in the field of time series forecasting, The results showed that the statistical models tended to have lower prediction errors. However, it was also noted that the results might be dataset-dependent. Therefore, it is necessary to compare statistical models with machine learning models when performing time series forecasting.

Recently, transformer is applied in several application field. for example, vision transformer[6] is applied in computer vision, Transformer is applied in NLP[3].

In the time series forecasting field, Haoyi Zhou et al.[7] propose that informer: Transformer improved as time series. Informer forecast time series with one forward procedure. It is not auto-regressive.

For these reasons, RNN is compared with baseline model as ARIMA and experimented on both auto-regressive and one forward procedure forecasting with RNN in our study.

2. Methodology

2.1. Autoregressive Integrated Moving Average(ARIMA)

If it define L^d as Eq. 1, d-order differencing series as $\Delta^d y_t$ (where $\Delta^0 y_t = y_t$) is formulated as Eq.(2). Autore-

gressive Integrated Moving Average(ARIMA) is formulated as Eq.(3). Note that ϕ refers to the autoregressive coefficient (strength of correlation), ε_t refers to the error (following a normal distribution with mean 0 and variance σ^2), and θ refers to the moving average coefficient (strength of correlation).

$$L^d y_t = y_{t-d} \quad \ldots\ldots\ldots (1)$$
$$\Delta^d y_t = (1-L^d) y_t \quad \ldots\ldots\ldots (2)$$
$$(1-\sum_{i=1}^{p} \phi_i L^i)\Delta^d y_t = (1+\sum_{j=1}^{q} \theta_j L^j)\varepsilon_t \quad \ldots\ldots (3)$$

2.2. Seasonal Autoregressive Integrated Moving Average(SARIMA)

Seasonal Autoregressive Integrated Moving Average(SARIMA) applied to seasonal variation components such as one-week cycle and quarterly cycle with ARIMA. in this time, the model was applied to a one-week cycle. Fig.1 shows an overview of the SARIMA model. Note that y_t is the data at each time.

The data circled in blue are ARIMA terms. on the other hand, The data circled in red are SARIMA terms. From this, this can include long-term components in the model.

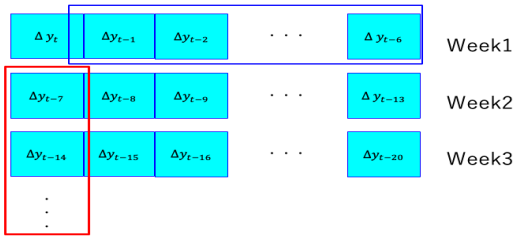


Fig. 1. SARIMA

If the order of seasonal auto regressive, the order of seasonal moving average, the lag operator for the seasonal variation component and D-order seasonal difference is define as P, Q, L^{sD} and $\Delta_s^D y_t$ (but $\Delta_s^0 y_t = y_t$), SARIMA is formulated as in Eq. (4). Note that ϕ is the auto regressive coefficient (strength of correlation), θ is the moving average coefficient (strength of correlation), Φ is the seasonal auto regressive coefficient (strength of correlation), Θ is the seasonal moving average coefficient (strength of correlation), and ε_t is the error (following a normal distribution with mean 0 and variance σ^2).

$$(1-\sum_{i=1}^{p} \phi_i L^i)(1-\sum_{I=1}^{P} \Phi_I L^{sI})\Delta^d \Delta_s^D y_t$$
$$= (1+\sum_{j=1}^{q} \theta_j L^j)(1+\sum_{J=1}^{Q} \Theta_J L^{sJ})\varepsilon_t \quad \ldots\ldots (4)$$

2.3. Recurrent neural network(RNN)

Rosenblatt propose MultiLayer Perceptron(MLP)[8] which is model as display neuron in human brain to the computer. But, this learn sequence data well.

So, Elman propose Simple RNN[9] for learning sequence data by inputting outputs from hidden layer recursively.

RNN is formulated as Eq.(1) and Eq.(2) Note that t refers to time, refers to $\boldsymbol{x}_t$ refers to Input,$\boldsymbol{h}_t$ refers to hidden state vector (It means information in past time),$\boldsymbol{W}_x$ refers to weight for Input, $\boldsymbol{W}_h$ refers to weight for hidden state vector,$\boldsymbol{b}$ refer to bias.

$$\boldsymbol{h}_t = \tanh(\boldsymbol{x}_t \boldsymbol{W}_x + \boldsymbol{h}_{t-1}\boldsymbol{W}_h + \boldsymbol{b}) \quad \ldots\ldots (5)$$
$$\boldsymbol{y}_t = f(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y) \quad \ldots\ldots\ldots (6)$$

In this study, five types of RNN is built. They are broadly classified into two categories. Some are Many to One RNN, The others are Many to Many(Synced) RNN. both means relationship between input and label data.

Three types Many to One RNN is built. The figure is shown below Fig.2 and Fig.3.

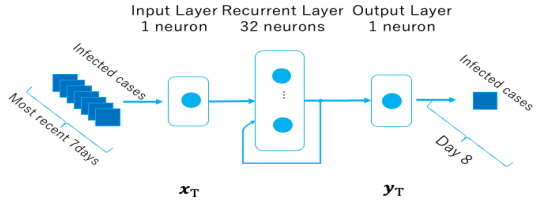


Fig. 2. Many to One with one dimension input and one dimension output

In case of Fig.2, 7days data is labeled by day8. both data is scalar values and applied to scaling condition in section3.2.1.

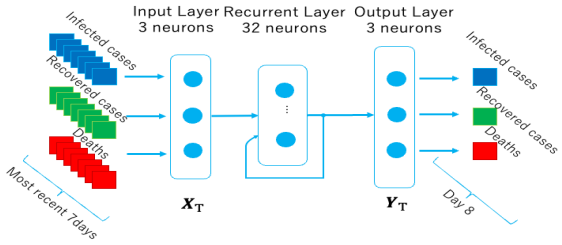


Fig. 3. Many to One with three dimension input and three dimension output

In case of Fig.3, 7days data is labeled by day8 such as Fig.2. However, both data is vector values. It is expected less prediction error than Fig.2 model, caused by using vector values. And also, is applied to scaling both condition in section3.2.2 and section3.2.3.

Furthermore, three types of Many to Many(synced) RNN is built. It shows these model below Fig.4 to Fig.6. RNN in Fig.4 has one dimension(newly cases in most recent one week) inputs and one dimension(newly cases in

next week) outputs. In this settings, is applied to scaling condition in section3.2.1. This model targets as forecasting next week cases by learning feature pattern at one week's sequence.

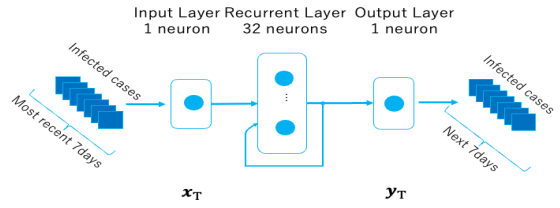


Fig. 4. Many to Many with one dimension input and one dimension output

RNN in Fig.5 has three dimension(newly cases, recovered and deaths in most recent one week) inputs and three dimension(newly cases, recovered and deaths) outputs. This expect better least prediction error than model of Fig.4. because this model use more variables.

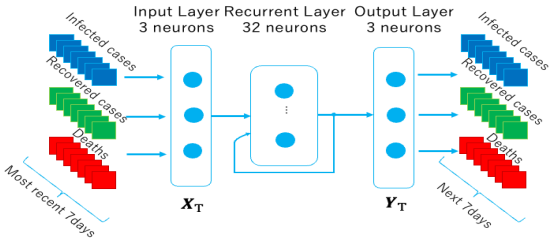


Fig. 5. Many to Many with three dimension input and three dimension output

RNN in Fig.5 has three dimension(newly cases, recovered and deaths in most recent one week) inputs and one dimension(newly cases) outputs. By mapping the 3-variable sequence input's information pattern to a 1D series, it is expected that a more accurate model will be constructed than the models in Figure4 and Figure5.

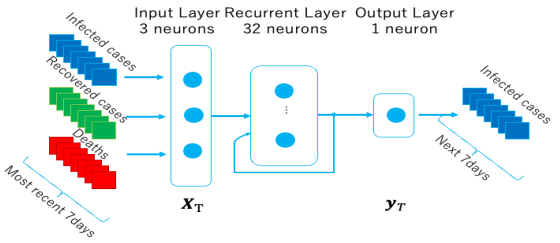


Fig. 6. Many to Many with three dimension input and one dimension output

In Fig.5 and Fig.6 settings, model is applied to scaling both condition in section3.2.2 and section3.2.2.

3. Experiment

In this study, Data sets[10] is used, from Center for Systems Science and Engineering(CSSE) at Johns Hopkins. This data sets include the cumulative number of confirmed cases, recovered cases, and death in each country.

And, new data sets is created to display number of newly infected cases, recovered cases and deaths due to preprocess data sets as differencing like $y_t - y_{t-1}$. Note that total terms is from January 23, 2020 to October 28, 2020. The training data terms was from January 23, 2020 to October 21, 2020, and the test data terms was from October 22, 2020 to October 28, 2020.

3.1. Conditions for statistical model

Parameters is a need to explore for ARIMA and SARIMA.

Parameters is explored in range at Table.1 and Table.2.

Table 1. Parameter search range for ARIMA(p,d,q)

p(min)	p(max)	d(min)	d(max)	q(min)	q(max)
1	3	0	1	0	3

Table 2. Parameter search range for SARIMA$(p,d,q,P,D,Q)_7$

p(min)	p(max)	d(min)	d(max)	q(min)	q(max)
1	3	0	1	0	3
P(min)	P(max)	D(min)	D(max)	Q(min)	Q(max)
0	3	0	2	0	3

3.2. condition for RNN

To avoid differences in model properties due to differences in the number of neurons in the hidden layer, the number of neurons in the hidden layer was set to 32 for all models. Also, Mean square error is used and it is optimized with Adam[11].

A forecasting model tends to overfit because the model has many parameters. Thus, L2-norm is used to regularize weight with in this study. Regularization is a method for keeping weight values fitting the model because significant weight values cause overfitting.

Model learning takes the form of determining the weights that minimize the value of the loss function E. Then, as in the second term of the Eq.(7), regularization is possible by adding a penalty term to the loss function to keep the weights small and minimizing the value of the new loss function E_w. In this case, the L2 norm was adopted. Note that λ is a parameter indicating the strength of regularization.

$$E_w = E + \lambda \sum_{k=1}^{n} w_k^2 \quad \ldots\ldots\ldots\ldots\ldots (7)$$

Since the size of λ is proportional to the size of the loss function value, it is adjusted to reduce its weight. In this case, λ is set as 0.001.

Input data is scaled with Eq.(8) and Eq.(9). Note that range for scaling is one from minus one.

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad \ldots\ldots\ldots\ldots \quad (8)$$

$$x_{scaled} = x_{std}(max - min) + min \quad \ldots\ldots \quad (9)$$

3.2.1. Scaling condition 1

This condition is using scalar inputs. In this condition, x_{min} is 0 which is the minimum value in the data sets and x_{max} is 1762 which is the maximum value in recovered cases.

3.2.2. Scaling condition 2

This condition is using vector inputs. In this condition, x_{min} is 0 which is the minimum value in the data sets and x_{max} is 3941 which is the maximum value in the data sets.

3.2.3. Scaling condition 3

This condition is using vector inputs. In this condition, x_{min} is 0 which is the minimum value in the data sets and x_{max} is 1762, 3941, and 29 which are maximum value each variable value.

3.3. Evaluation function

3.3.1. Akaike Information Criterion(AIC)

Akaike Information Criterion(AIC) is proposed by Akaike [12][13]. This value is for evaluation for model fitting. So, note that this is used error in train term for calculation. The smaller the this value, the better the model can be considered. AIC is formulated as Eq.(10) and Eq.(11). Note that k refer to number of parameters, L refer to likelihood, N refer to number of sample in train term, RSS refer to residual sum of squares.

$$AIC = 2k - 2\ln(\hat{L}) \quad \ldots\ldots\ldots\ldots \quad (10)$$

$$= 2k + N\ln 2\pi + N\ln\frac{RSS}{N} + N \quad \ldots\ldots \quad (11)$$

Indeed, The values is used as following Eq.(12) because N and $\ln 2\pi$ are constant.

$$AIC = N\ln\frac{RSS}{N} + 2k \quad \ldots\ldots\ldots\ldots \quad (12)$$

3.3.2. Mean Absolute Error(MAE)

It was used Mean Absolute Error(MAE) for evaluation of prediction error in test term at each model. MAE is formulated as Eq.(13). Note that n refer to number of data, $y_i(i = 1,2,3,...,n)$ refer to observed values, $\hat{y}_i(i = 1,2,3,...,n)$ refer to predicted value.

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad \ldots\ldots\ldots\ldots \quad (13)$$

MAE is less susceptible to outliers

3.3.3. Root Mean Square Error(RMSE)

It was used Root Mean Square Error(RMSE) for evaluation of prediction error in test term at each model. RMSE is formulated as Eq.(14)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad \ldots\ldots\ldots\ldots \quad (14)$$

RMSE is sensitive to the presence of outliers. Because prediction error is squared.

4. Result

As the exploration result about parameters of ARIMA and SARIMA, ARIMA(2, 1, 3) and SARIMA$(3,1,2)(0,2,3)_7$ is selected and used for forecasting. It is smallest AIC value in all parameters.

Model fitting result shows in Table.3 to Table.5. Note that if I define dimension as D, 1Dto1D, 3Dto1D, and 3Dto3D respectively mean 1D-input 1D-output, 3D-input 1D-output, and 3D-input 3D-output and number in round brackets of right side of model name mean scaling condition. For example, 3Dto1D(2) means 3Dto1D model in setting on scaling condition 2.

Table 3. Evaluation model fitting of Statistical model for COVID-19 cases

	ARIMA	**SARIMA**
AIC	3361	3299

Table 5. Evaluation model fitting of Many to One for COVID-19 cases

	1Dto1D(1)	**3Dto3D(2)**	**3Dto3D(3)**
AIC	4909	5694	5048

Forecasting result shows in Table.6 to Table.8. Note that gap in the these Table means that how many % error has increased or decreased with respect to the RMSE of ARIMA.

Table 6. Prediction error of statistical model for COVID-19 cases

	ARIMA	**SARIMA**
MAE(people)	86	53
RMSE(people)	103	70
Gap(%)	0.0	-32.0

Table 4. Evaluation model fitting of Many to Many(synced) with scaling condition 1, 2, and 3

	1Dto1D(1)	3Dto1D(2)	3Dto3D(2)	3Dto1D(3)	3Dto3D(3)
AIC	4951	5175	5366	4918	5082

Table 7. Prediction error of Many to Many(synced) with scaling condition 1, 2, and 3

	1Dto1D(1)	3Dto1D(2)	3Dto3D(2)	3Dto1D(3)	3Dto3D(3)
MAE(people)	101	48	37	84	46
RMSE(people)	110	59	47	92	52
Gap(%)	6.8	-42.7	-54.4	-10.7	-49.5

Table 8. Prediction error of Many to One for COVID-19 cases

	1Dto1D(1)	3Dto3D(2)	3Dto3D(3)
MAE(people)	95	121	101
RMSE(people)	110	128	121
Gap(%)	6.8	24.3	17.5

As the result, many to many(synced) RNN mapped to 3D from 3D is most minimized forecasting error; is also with the largest percentage decrease relative to the RMSE of ARIMA.

Table.9 shows the relative reduction of the influenza forecasting model to the RMSE of ARIMA in the study by Wu[2] et al.

Table 9. Model evaluation about influenza-like illnesses by Wu[2]et al.

	ARIMA	LSTM	Seq2Seq	Transformer
Gap (%)	0.0	-20.9	-37.1	-42.4

From the results in Table.6 to Table.8, and Table.9, the model mapped from 3-D to 1-D was as accurate as Transformer, and the model mapped from 3-D to 3-D was 12% more accurate than the Transformer. Note that the models created in this study cannot be compared because the model by Wu et al. is based on the ratio of the number of patients in the previous week.

5. Discussion

The following presents the results of the fitting. From Table.3 to Table.5, model composed by scalar tend to low AIC. Thus, this model is better fitting than the model composed by vectors. This is caused by the more variables, the more difficult to optimize the loss function. In Spyros et al.'s study[5], They state that LSTM tends to overfit. So, RNN has possible that it is hard to optimize for non-stationary time series. Thus, it needs to take the difference in data, such as ARIMA and SARIMA, to optimize the model. This has the potential to cause better fitting of RNN and minimize prediction error.

Furthermore, in setting on vector-input models, it is observed that it tends to more fit the model in setting on scaling condition 3 than that on scaling condition 2 from Table.4 and Table.5. However, in the point of view about RMSE, it is observed that it tends to more minimize the error in setting on scaling condition 2 than that on scaling condition 3 from Table.7 and Table.8. It also can observed. For example, in Fig.7 and Fig.8. it show forecasting result Many to Many RNN with 3Dto3D in scaling condition 2 and 3. in Fig.7, There is the big bias in training term. Thus, it cause big AIC value.

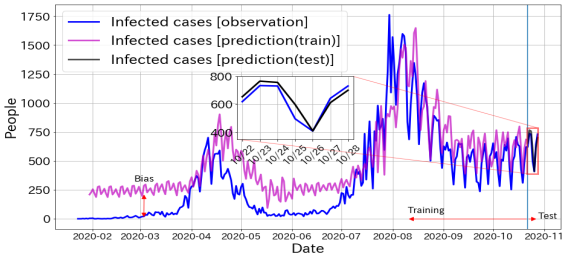


Fig. 7. Prediction result of 3Dto3D RNN at scaling condition 2

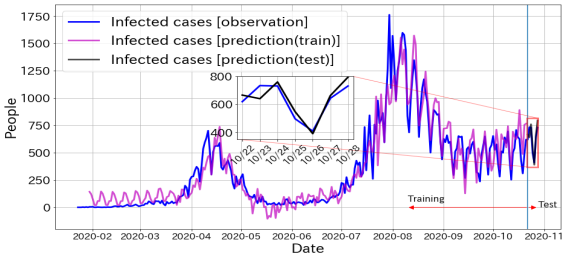


Fig. 8. Prediction result of 3Dto3D RNN at scaling condition 3

The next paragraph refers to the result of the forecasting error. From Table.6 to Table.8, Many to One RNN tends to grow the gap of RMSE from the baseline model(ARIMA) when it compares the multi-variable Many to Many(synced) RNN. This trend is confirmed because the relationship between training data and label data is not settled, making it difficult for learning to converge; Many to One is fitting one day(label data) with seven

days(training data).

Whereas, it is confirmed that multi-variable Many to Many(synced) RNN model tend to reduce the RMSE gap from baseline model(ARIMA) in result of Table.7. This trend is confirmed because the relationship between training and label data is settled, making it easy for learning to converge. Many to Many fits each day in seven days(label data) and each day in seven days(training data). In Many to Many settings, it is confirmed that scalar input grows RMSE from baseline, but vector input reduces RMSE from the baseline. This is thought to be due to the fact that the addition of the features successfully represented the relationship between each date by its weight.

In Many to Many with scaling condition 2 settings in Table7, the 3D to 1D RNN had the same accuracy as the Transformer created by Wu et al.[2] in Table.9 and the 3D to 3D RNN had a smaller prediction error. This is thought to be because Many to Many(synced) RNN with multi-dimensional input act as Self-Attention in the hidden layer, channeling feature with significant correlations to the next neuron on a day-by-day basis due to the efficacy of tanh in the hidden layer. It will be necessary to construct models using other data sets to verify this phenomenon in the future.

From Table.7, Comparing the RNN mapped from 3D to 3D with the RNN mapped from 3D to 1D, it is confirmed that mapping from a high-dimensional space to a same dimensional space yields better results than mapping from a high-dimensional space to a low-dimensional space. This can be considered as follows from the fact that the hidden layer in RNN is defined as in the Eq.(5). In the hidden layer, rather than recursively inputting COVID-19 cases, it is thought that recursively inputting the same dimension as the input does not cause a lack of time-series information. This is because the hidden state vector $\boldsymbol{h}_{t-1}$ and related weight are defined and calculated as a vector.

Future work includes fitting for differencing data; Hewamalage et al. say that RNN is easy to fit given a seasonal pattern and sufficient amount of data[14]. However, the data sets used in this study have different long-term amplitudes, which makes modeling difficult. Therefore, it has possible that modeling will be easier and forecasting accuracy will be improved by converting non-stationary time series data to stationary time series data by transforming the first-order difference.

Furthermore, there are different amplitudes in the short term. It has possible that this issue could be solved by scaling on the input term rather than on the entire training term.

Thus, it has possible that these improvements improve forecasting accuracy.

6. Conclusion

This study compares ARIMA, SARIMA, and many-to-many (synced) RNNs with three input-output relationships to forecasting time series and compares with Wu et al.

The ratio of increase/decrease of the error based on the RMSE of ARIMA is calculated and compared the results. As a result, the many-to-many (synced) RNN with 3D input for 1D output showed the same level of accuracy as Wu et al.'s Transformer. The many-to-many (synced) RNN with 3D input for 3D output showed 12% higher accuracy than Wu et al. The reason for this accuracy is thought to be that tanh, the activation function of the hidden layer in the multidimensional input-output RNN, functioned as Self-Attention.

As for future activities, whether or not it functions as Self-Attention using other data sets and constructing a model with differencing data is needed to verify.

References:

[1] World Health Organization and others. "COVID-19 weekly epidemiological update, edition 91, 11 May 2022," World Health Organization, 2022

[2] Wu, Neo and Green, Bradley and Ben, Xue and O'Banion, Shawn. "Deep transformer models for time series forecasting: The influenza prevalence case," arXiv preprint arXiv:2001.08317, 2020.

[3] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Łukasz and Polosukhin, Illia. "Attention is all you need," Advances in neural information processing systems, Vol.30, 2017.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory," Neural computation, Vol.9, No.8, pp.1735–1780, 1997.

[5] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and machine learning forecasting methods: Concerns and ways forward," PloS one, Vol.13, No.3, pp.e0194889, 2018.

[6] Dosovitskiy, Alexey and Beyer, Lucas and Kolesnikov, Alexander and Weissenborn, Dirk and Zhai, Xiaohua and Unterthiner, Thomas and Dehghani, Mostafa and Minderer, Matthias and Heigold, Georg and Gelly, Sylvain and others. "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020

[7] Zhou, Haoyi and Zhang, Shanghang and Peng, Jieqi and Zhang, Shuai and Li, Jianxin and Xiong, Hui and Zhang, Wancai. "Informer: Beyond efficient transformer for long sequence time-series forecasting," Proceedings of AAAI, 2021

[8] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological review, Vol.65, No.6, pp.386, 1958.

[9] Jeffrey L Elman. "Finding structure in time," Cognitive science, Vol.14, No.2, pp.179–211, 1990.

[10] https://github.com/CSSEGISandData/COVID-19

[11] Kingma, Diederik P and Ba, J. "Adam: a method for stochastic optimization 3rd Int," Conf. for Learning Representations, San,2014.

[12] Hirotugu Akaike. "A new look at the statistical model identification," IEEE transactions on automatic control, Ieee, Vol.19, No.6, p.716-p.723, 1974.

[13] Hirotogu Akaike. "Information theory and an extension of the maximum likelihood principle," Springer, p.199-p.213,1998.

[14] Hewamalage, Hansika and Bergmeir, Christoph and Bandara, Kasun. "Recurrent neural networks for time series forecasting: Current status and future directions," International Journal of Forecasting, Elsevier, Vol.37, No.1, p.388-p.427, 2021.