TEAM15 COVID-19 Prediction FINAL REPORT

Chenxin Yang
UCLA Math Department
Los Angeles, United States
chenxinyang@g.ucla.edu

Yiwei Liao
UCLA
Los Angeles, United States
tonyliao0406@gmail.com

Jieyi Wang
UCLA Math Department
Los Angeles, United states
jieyiwang@g.ucla.edu

Tianyi Bao
UCLA
Los Angeles, United states
enigmabao@g.ucla.edu

ABSTRACT

We study and predict the number of confirmed cases and deaths in each US state due to COVID-19 With the data of previous confirmed cases and death and related features. We use multiple models for time-series data, including ARIMA, Neural Networks with LSTM, Seq2seq and Seq2seq with attention, as well as PROPHET, a forecasting tool designed by Facebook, to predict the future confirmed cases and deaths due to Covid-19. We compare the MAPE(Mean Absolute Percentage Error) score for each model and select the Seq2seq with attention as our final model. We also compare and remark on their relative effectiveness in application to the dataset.

ACM Reference Format:
Chenxin Yang, Yiwei Liao, Jieyi Wang, and Tianyi Bao. 2018. TEAM15 COVID-19 Prediction FINAL REPORT. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Covid-19 has been a huge impact to people's lives since March 2020. Tha pandemic sirst started in China and then quickly spread to the entire world. US was one of those countries that have been heavily influenced by the virus. The number of confirmed cases has been increasing at an exponential rate without any sign of stopping. Up to now, US has over 17 million confirmed cases with 310 thousands deaths. Stopping the spread of Covid-19 has become an essential mission of humankind.

We will do Covid-19 prediction with a goal to provide better methods for estimates that can assist medical and governmental institutions to prepare and adjust as pandemics unfold.

We will use MAPE (Mean Absolute Percentage Error) to evaluate the performance of mour model, MAPE is calculate as MAPE = $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{p_i-a_i}{a_i}\right|$, where: n is the total number of datapoints, p_i is our prediction, a_i is the actual value.

Specifically, we are interested in predicting the cumulative number of confirmed cases, as well as the cumulative number of confirmed deaths for each of the 50 states in the U.S.A. The prediction has two round, with the first round's test data range from 9/1/2020 to 9/28/2020, and we could use many models to test the MAPE score on the testing dataset. The second round is the final round and we have to predict the number of confirmed cases and deaths from 12/7/2020 to 12/13/2020, and we have only one chance.

We will collect training and testing data from JHU CSSE, as well as SafeGraph. We thank their efforts for collecting these data and making them public.

We thank Professor Yizhou Sun, and TAs Junheng Hao, Shichang Zhang, Yue Wu, Zijie Huang for organizing this competition.

2 RELATED WORK

2.1 Time series data forecasting

A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Time Series analysis can be useful to see how a given asset, security or economic variable changes over time. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.[1] Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values.

3 DATA PREPOSSESSING

The data we have date from 4/12/2020 to 8/31/2020. It includes information of total confirmed cases and deaths in each of 50 states of US. Besdies, it provide information like Recovered, Active, Incident Rate, etc. There are many missing values in the data and the data are very noisy due to irregular updates of confirmed cases and deaths, we need to perform some prepossessing to make the data smoother and look more similar to a typical pandemic's growth.

[1]https://en.wikipedia.org/wiki/Time_series

- There are many missing columns in the training data, we prepare to do following to handle missing data:

  For some of the states with missing "recovered" data, it seems to be the case that they did not record the recovered data at all or over a certain period. This causes the number of active cases to be not very accurate. If a state shows a long period of missing recovered data, we will adjust the " active" data a bit based on the average recovery rate and then calculate the recovered data based on "recovered" = "confirm"-"death"-"active"

  The "Hospitalization Rate" and "people hospitalized" columns also have missing data for certain states. We plan to make up these missing data with the average hospitalization rate of other states which are having similar amounts of cases with it.

  Starting from Aug 28th, none of the states recorded "Hospitalization Rate" "people hospitalized" data, we are going to fill out each states' missing data by taking the average of its data over the past three days.

- We use one hot encoding to encode the state feature which is a categorical column.

- We split the data to training and validation set. The training set is from 4/14/2020 to 8/31/2020 and the testing set is from 8/3/2020 to 8/31/2020.

- The number of confirmed cases and deaths are cumulative, it is not good for time-series prediction models, so we changed the cumulative number to daily change.

- We perform smoothing techniques to reduce short-term volatility in data. Specifically We performed smoothing by n days to stabilize the data; our result is to calculate cumulative so the information lost in smoothing will not affect result that much.

- We perform Minimum-maximum scaling for each state; We have 50 min max scalers, and for a given state, $X_{scale} = \frac{max(X)-x_i}{max(X)-min(X)}$ where x_i is the number of cases in i^{th} day, X is the list of all number of cases in each day of the given state.

- For the convenience of LSTM neural network, we group the data into length 7 vectors. E.g.: it looks like [[day1, day2, ..., day7], [day8, ..., day14], [day15, ..., day21], ...].

- We use a pairing function that uses a sliding time window to create more samples and convert the problem into a supervised learning format.The pairing is between 7 days before and 7 days after.The first two samples look like this: [d1, d2, d3, d4, d5, d6, d7] -> [d8, d9, d10, d11, d12, d13, d14], [d2, d3, d4, d5, d6, d7, d8] -> [d9, d10, d11, d12, d13, d14, d15]

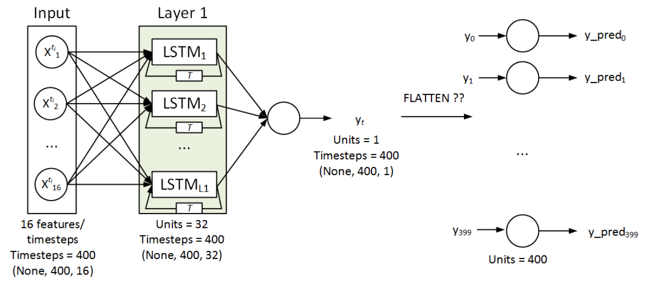

Figure 1: Neural Network with LSTM

4 NEURAL NETWORK WITH LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture[1] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections[2]. Moreover, since the mode considers the time sequence, so it could be utilized to model how the COVID-19 cases could change by time. The general network structure is visualized in Figure 1.

However, for normal LSTM, it has a major drawback; input and output length must be the same, we want longer input length to contain more information while output length remains the same (because longer output length increases error rate drastically). Due to this drawback, We get a MAPE score of 2.39 on testing data, which is not ideal.

We also tried bidirectional LSTM; The advantage of it is that it can be trained using all available input information from both backwards and forward states. It has a structure that splits neurons of a regular Neural Network into two parts, which are responsible for positive time direction and negative time direction respectively (Mike, 2673). However, we did not see a significant improvement on the MAPE score and the same problem for normal LSTM still exists in bidirectional LSTM. We quickly gave up on this model and moved on to other models.

5 SEQ2SEQ

5.1 model description

seq2seq is a general-purpose encoder-decoder framework for Tensorflow that can be used for Machine Translation, Text Summarization, Conversational Modeling, Image Captioning, and more.[3] The encoder encodes the input sequence, while the decoder produces the target sequence.[4] In our seq2seq model, we run LSTM as an encoder cell over the input and store the last hidden state outputed by the LSTM. In the decoder step, we feed the output vector to another LSTM cell that computes the next hidden state. Seq2seq is usually in translation of different languages, as it allows different

[2]https://en.wikipedia.org/wiki/Long_short-term_memory
[3]https://google.github.io/seq2seq/
[4]https://guillaumegenthial.github.io/sequence-to-sequence.html

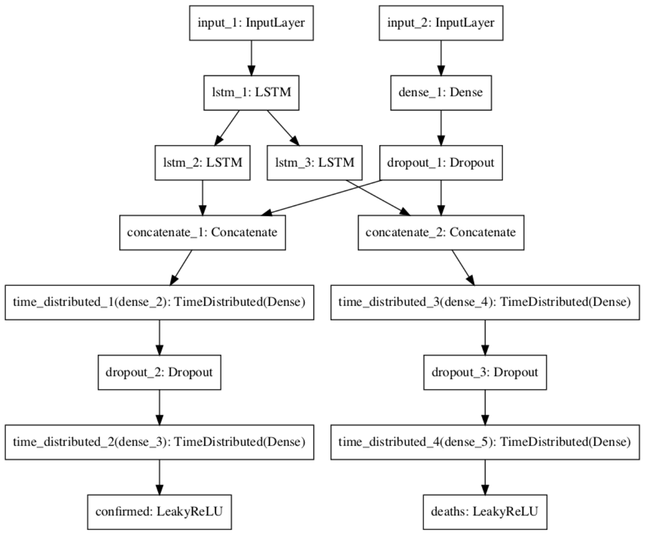
input_1: InputLayer
input_2: InputLayer
lstm_1: LSTM
dense_1: Dense
lstm_2: LSTM
lstm_3: LSTM
dropout_1: Dropout
concatenate_1: Concatenate
concatenate_2: Concatenate
time_distributed_1(dense_2): TimeDistributed(Dense)
time_distributed_3(dense_4): TimeDistributed(Dense)
dropout_2: Dropout
dropout_3: Dropout
time_distributed_2(dense_3): TimeDistributed(Dense)
time_distributed_4(dense_5): TimeDistributed(Dense)
confirmed: LeakyReLU
deaths: LeakyReLU

Figure 2: seq2seq structure

input output length, which solves the problem in LSTM. However, it will lose information for longer sequence. The detailed structure of Seq2seq in our model is in figure 2.

As shown in figure 2, $Input_1$ is the time series data for deaths and confirmed; $Input_2$ is the one hot matrix for state information; we repeat the one hot matrix for 7 times to match the shape and concatenate it to the 3d time series data.

However, seq2seq will lose information for longer sequence. So we add attention to the original seq2seq structure. An attention mechanism is a part of a neural network. At each decoder step, it decides which source parts are more important. At different steps attention lets a model "focus" on different parts of the input.[5] Therefore, Seq2seq with attention improves the original seq2seq by solving the problem that longer sequence will result in information lose. So we finally decide to fine-tuning on seq2seq with attention. The detailed structure of our seq2seq with attention is in figure 3.

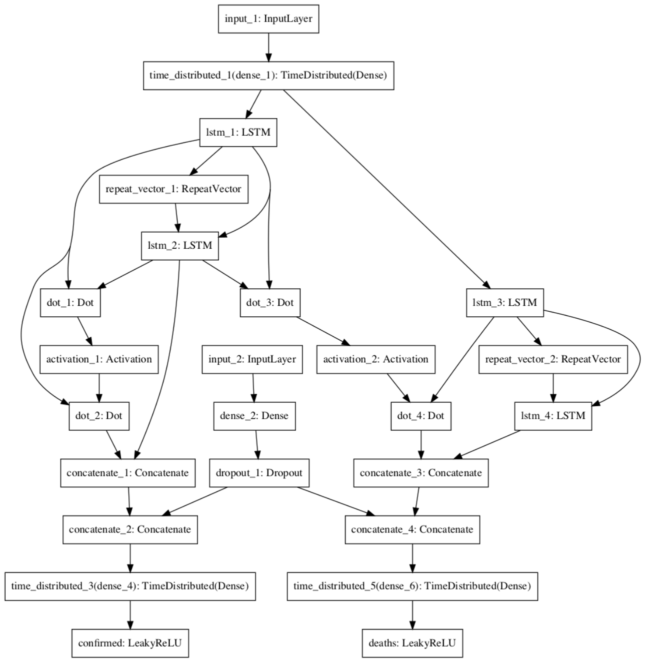
input_1: InputLayer
time_distributed_1(dense_1): TimeDistributed(Dense)
lstm_1: LSTM
repeat_vector_1: RepeatVector
lstm_2: LSTM
dot_1: Dot
dot_3: Dot
lstm_3: LSTM
activation_1: Activation
input_2: InputLayer
activation_2: Activation
repeat_vector_2: RepeatVector
dot_2: Dot
dense_2: Dense
dot_4: Dot
lstm_4: LSTM
concatenate_1: Concatenate
dropout_1: Dropout
concatenate_3: Concatenate
concatenate_2: Concatenate
concatenate_4: Concatenate
time_distributed_3(dense_4): TimeDistributed(Dense)
time_distributed_5(dense_6): TimeDistributed(Dense)
confirmed: LeakyReLU
deaths: LeakyReLU

Figure 3: seq2seq with attention structure

5.2 Implementation

To enhance the performance of seq2seq with attention, We have done following methods:

- We share the context vector for two output pipelines; we concatenate the context vector from the pipelines of confirmed to that of death.
- We perform Hyper-parameter tuning both by grid search and manual eyeballing.
- We use Orthogonal initialization to combat exploding and vanishing gradients.
- We use leaky relu as activation functions. This is determined from repeated experiment, as in our setting leaky relu has better performance than tanh, sigmoid and other activation functions.
- We use ADAM as optimizer to update network weights.
- Since we find there is no decrease in validation loss after 10 epochs, We perform early stopping to prevent overfit.
- We dropout after every dense layer; use combination of many layers and high dropout rates instead of less layers and low dropout rates.
- We also Use train loss vs validation loss to determine whether there exists an overfit or underfit and decide whether we should increase or decrease the layers of neural network.

5.3 Output

The final train loss vs validation loss of seq2seq with attention is in figure4. We could see there is a sharp drop in both train and validation loss in first several epochs. After 10 epochs, the lines of both train and validation lass become very flat and there is little improvement.

[5]https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

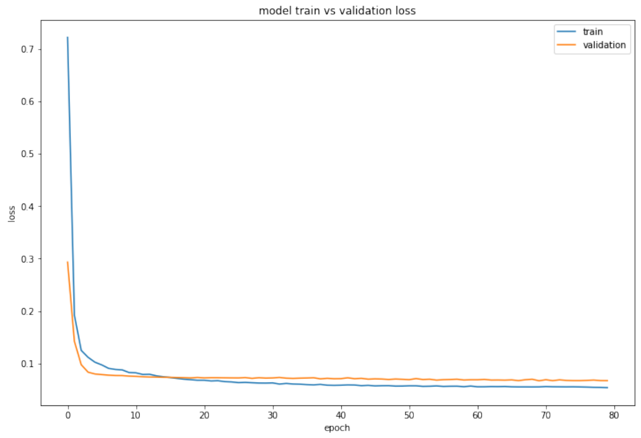

Figure 4: train loss vs validation loss

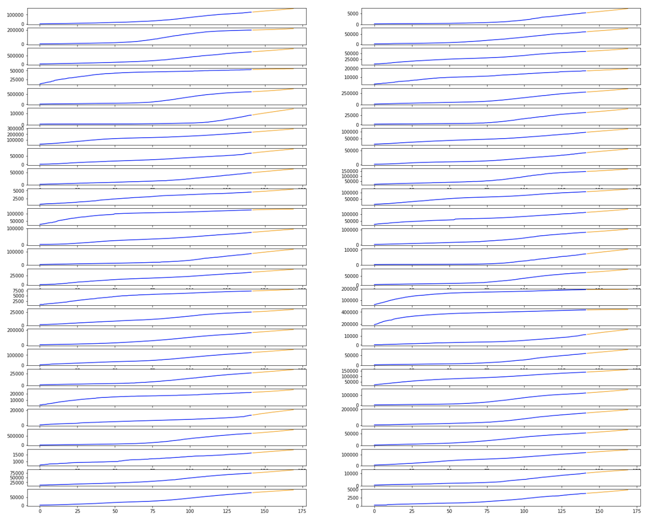

Figure 5: Sample output

We have the sample output in figure5. The figure shows how the confirmed cases change by time in each of the 50 states. The blue line ranges from 4/12/2020 to 8/31/2020. The yellow line shows the prediction of confirmed cases that range from 9/1/2020 to 9/28/2020. The MAPE score on the round 1 testing data-set is 2.03, which is a significant improvement compare to our other models.

6 ARIMA

Since COVID-19 data is obviously related with time because if its monotonicity and the fact that people are more likely to be infected when the number of cases gets high. We learned in class that Auto-regression is a model that works well with time series data. ARIMA stands for Auto Regression Integrated Moving Average. It is an algorithms specifically designed for time series data pattern capturing that is a combination of the auto regression algorithm and the moving average algorithm. In this part of the report we will first breakdown the details of ARIMA and how we used this model to produce relatively good results.

6.1 Auto Regression

Auto regression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step. The math behind auto regression is actually fairly simple. Given a hyper-paramer p, we predict data at time t by looking at its first p: $Y_t = \alpha+\beta_1 Y_{t-1}+\beta_2 Y_{t-2}+\beta_3 Y_{t-3}+...+\beta_p Y_{t-p}$. In the above equation both α and $\beta_i (1 \leqslant i \leqslant p)$ are parameters that the model is going to estimate.

6.2 Moving Average

This is an extension of the auto regression algorithm where prediction of data at time t actually depends on previous error of the auto regression model, namely:$Y_t = \alpha + \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + \phi_3\epsilon_{t-3} + ... + \phi_q\epsilon_{t-q}$. In the above equation both α and $\phi_i (1 \leqslant i \leqslant q)$ are parameters that the model is going to estimate.

After we have the above two algorithms, combining them together gives us the complete ARIMA model, namely:

$Y_t = \alpha+\beta_1 Y_{t-1}+\beta_2 Y_{t-2}+\beta_3 Y_{t-3}+...+\beta_p Y_{t-p}+\phi_1\epsilon_{t-1}+\phi_2\epsilon_{t-2}+\phi_3\epsilon_{t-3} + ... + \phi_q\epsilon_{t-q}$.

In human words, this means that: Predicted Yt = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags)

In practice, there is another hyper-parameter d which determines the degrees of differencing on the data in order to make it stationary. Differencing here means substracting every data point with the previous one. For simplicity, in this project we did not look into differencing on the data thus fixing the degree of differencing to one.

6.3 Implementation

For the implementation of ARIMA we utilized the statsmodel package which has a ready-to-use ARIMA model. During our implementation, a big challenge is how to search for the best hyper-parameter and what standard to use for testing. Because of the submission cap on Kaggle, we cannot run test on every set of hyper-parameters. We explored two ways of measuring performance. 1. AIC. AIC stands for Akaike's Information Criterion which is an estimator of out-of-sample prediction error and thereby relative quality of statistical models for a given set of data. 2. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Thus, AIC provides a means for model selection. Its mathematical representation is the following: $AIC = -2log(L) + 2(p + q + k + 1)$.

Generally a smaller AIC means a better model. With this information we first implemented a grid search mechanism within a pre-determined hyper-parameter candidates pool. For every set of candidate hyper-parameters we use that to construct an ARIMA model to fit the data and compute the respective AIC to find the best (p, d, q) for our dataset. This gave us a performance(MAPE) of about 3. We immediately realized several drawbacks to our design. First of all, since every state has a different situation in terms of policy, effectiveness, population density or medical infrastructure, it would be unfair to choose one set of parameters for the whole nation. Secondly, we found that using the set of parameters that gives the lowest AIC does not necessarily give the best performance, since our goal is minimizing MAPE, it would be more reasonable to use MAPE to find the best parameters.

6.4 Model tuning

Thus we designed a different grid search mechanism. We first split the data by states. Then we split the data into training sets and validation sets. Since data is increasing with time, we decided to use the last several days as validation for model selection and the previous data as training. This greatly improve our performance and we are able to get our lowest MAPE on the test set of around 2.3. However, there are still disadvantages to our design that still need to be fixed. First of all, from the trend of COVID-19 cases, the rate of increase of both confirmed and death numbers are always getting steeper and steeper. This trend make the model easily underestimate in the prediction. With our use of the last several days as validation set, we miss out on the trend that most closely resemble the real trend. This tend to make prediction even more inaccurate. Secondly, due to our limitation in computation power and time, the condidate hyper-parameter pool is very small $1 \leqslant p \leqslant 4, d = 1, 0 \leqslant q \leqslant 7$.If we have better computation power and more time, we will be able to improve our ARIMA estimation even more.

7 FB PROPHET

As the number of cases is significantly affected by the seasonal effects, especially the holiday seasons, we tried to find a model that is good at predicting time series data while taking the holidays into account. After reviewing some papers, we found out that fbprophet actually has a model that is good at training this type of data. It takes into account the changes caused by seasonal effects and holidays, which traditional time series methods such as ARIMA cannot take into account. It is also easy to interpret as the model is designed to help explain these time series, such as the website visits and the cases confirmed in a pandemic. According to some of the past papers on covid prediction, fbprophet is good at predicting the number of cases, although it is not as robust and stable as the LSTM and ARIMA.[6]

7.1 Implementation

This method is a typical model-fitting method. The model, which in essence is an additive model that adds up all kinds of effects, combines functions that three sub-models. Two of these sub-models record the periodic change caused by the seasonal changes and the holiday changes effects. The rest one is the growth model that learns the trend. The main function is a specific type of logistic regression with a piecewise linear growth rate. This function is responsible for learning the general trend without the effect of the holidays and seasons. When predicting, the model will also use the maximum likelihood method to learn the future piecewise growth rate and then predict using the logistic model. This function has several hyperparameters that are tunable, including the frequency that the growth rate changes and the coefficient of the function (the maximum number of people available for infection). In addition, the model also contains a function that mimics the effect of the seasonal changes, for instance, the difference of data between workday and weekends, using a standard Fourier series and with a normal distribution as its prior distribution. The last function in this model is a function that represents the holiday change's effect. It is just a matrix of the 1 functions (which equals 1 when true and 0 when false), with a normal distribution as its prior distribution.

[6]https://facebook.github.io/prophet/

7.2 Model tuning

We first trained the entire dataset, with each state having its own model. The MAPE on the training set was about 0.002 while the MAPE on the round 1 testing set was 2.26. As the model was overfitting, we tried different methods to tune the hyperparameter and did cross-validation. We tried to tune the two hyperparameters mentioned above. First, the number of people allowed to be infected was adjusted. By using the total population of the United States, the result was very inaccurate. Then, when only one state's data was used to train the model, the MAPE was also large. Thus, the original default value was used. Then, the frequency of growth rate change was tuned using cross-validation. The first cross-validation was run based on Alabama's data, and then the same hyperparameter was applied to other states' model. The MAPE was boosted significantly by doing so, probably because different states have a different response to the seasonal changes and holidays. Thus, cross-validation was run again on each states' data independently, and the hyperparameters were only used in their corresponding models. However, the MAPE did not change significantly and only increase at some other values. Therefore, the default value was also used in the final submission.

8 CONCLUSION

Table 1 shows the performance of all our models on round 1 testing data based on MAPE. We find Seq2seq with attention performs the best and we dicide to use it as our final model for round2. We are confident that the seq2seq with attention will outperform most of other models.

Table 1: MAPE for all modelsconsidered

Method	MAPE
Neural Network with LSTM	2.38
Seq2seq with attention	2.03
ARIMA	2.36
PROPHET	2.26

9 REFERENCE

Akaike, H. (1973), "Information theory and an extension of the maximum likelihood principle", in Petrov, B. N.; Csáki, F. (eds.), 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, USSR, September 2-8, 1971, Budapest: Akadémiai Kiadó, pp. 267–281. Republished in Kotz, S.; Johnson, N. L., eds. (1992), Breakthroughs in Statistics, I, Springer-Verlag, pp. 610–624

https://en.wikipedia.org/wiki/Time_series

https://en.wikipedia.org/wiki/Long_short-term_memory https://google.github.io/seq2seq/

https://guillaumegenthial.github.io/sequence-to-sequence.html

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.htm https://facebook.github.io/prophet/

https://smerity.com/articles/2016/orthogonal_init.html

https://www.kaggle.com/frlemarchand/covid-19-forecasting-with-an-rnn/output

https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/

https://levelup.gitconnected.com/building-seq2seq-lstm-with-luong-attention-in-keras-for-time-series-forecasting-1ee00958decb

https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/#:~:text=ARIMA%2C%20short%20for%20'Auto%20Regressive,used%20to%20forecast%20future%20values.
https://otexts.com/fpp2/arima-estimation.html