6501 Capstone Final Report – Group 1
Stock price prediction using GAN

Team member: Chen Chen, HungChun Lin

Working Schedule

Check Point Date	**Milestone**
09/14/2020 - 09/20/2020	Project Proposal, Search Data
09/21/2020 - 09/26/2020	Data Preprocessing
09/27/2020 - 10/03/2020	Data Preprocessing
10/04//2020 - 10/10/2020	LSTM model
10/11/2020 - 10/17/2020	LSTM model
10/18/2020 - 10/25/2020	GAN model
10/26/2020 - 10/31/2020	GAN model
11/1/2020 - 11/7/2020	GAN model
11/8/2020 - 11/14/2020	Final prediction
11/15/2020 - 11/21/2020	Final prediction
11/22/2020 - 11/29/2020	Final report
11/30/2020	Mock Presentation
12/07/2020	Final Presentation

Week2 – 09/21

This week, we did some research to write down our proposal, and found the proper dataset.

1. Dataset:

Data category	Description
Stock price (Technology stocks)	10 years of Apple, Amazon, Microsoft, Google Yahoo finance: https://finance.yahoo.com
Stock index	NASDAQ, NYSE, FTSE100, Nikkei225, BSE SENSEX, HENG SENG, SSE
Economic index	Crude Oil, Gold, VIX, USD index FRED: https://fred.stlouisfed.org
Daily news	Daily news of Apple company Seeking alpha: http://seekingalpha.com/
Technical indicator	7 and 21 days moving average, Exponential moving average, Momentum, Bollinger bands, MACD
Fourier transform	3, 6 and 9 components

2. Literary Research
    - Github: Using the latest advancements in AI to predict stock market movements:
      This Github is also using the GAN to predict stock price, and it included whole process of the project that we will build up our own model based on this Github.
      Reference: https://github.com/borisbanushev/stockpredictionai#overview

    - Paper: Stock Market Prediction Based on Generative Adversarial Network:
      This paper is about predicting the stock price through GAN, MLP and LSTM methods, and it only used 7 features to do the prediction.

    - Stanford Course Report: Generative Adversarial Network for Stock Market price Prediction:
      This is a report from Stanford course, this team used three methods to predict the stock price, which are ARIMA, LSTM and GAN, for them, they set ARIMA model as their base line.

    - Paper: Computational Intelligence in Data-Driven Modelling and Its Engineering Applications:
      This paper includes ARIMA, LSTM and GAN methods, they have tried GAN-F/ GAN-D/ GAN-FD/ LSTM-FD model.

3. Benchmark
   In our project, we will use LSTM and GAN to predict the stock price, and the benchmark of our project is **Basic LSTM**.
   Base on the basic model, we will try different activation function on each model, and for the GAN, we will try to use different loss function to improve the result.

4. The goal of this project
   The goal of this project is to predict stock price more accurate.

5. The contribution of this project

In this project, we will improve the performance of predicting the stock price by using GAN and adjusting the loss function.

Week3 – 09/28

In the week, we set up our model structure, and will show the detail of our structure below.

a. The input data -> (n-sample, feature, input-steps):

```
Stock price data: Apple real stock price
Features: Stock Index
          Economic Index
          Technical Indicator
          Fourier Transform
          Daily News
```

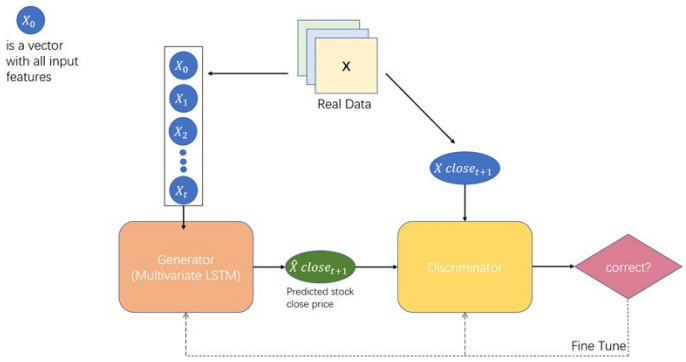
b. Output data -> (n-samples, output-steps):

```
Stock price data: Apple real stock price
```

c. LSTM (stacked LSTM):

```
Use Multivariate LSTM Model

gan_num_features = dataset_total_df.shape[1]
 sequence_length = 17

 class RNNModel(gluon.Block):

  def __init__(self, num_embed, num_hidden, num_layers, bidirectional=False,
               sequence_length=sequence_length, **kwargs):
      super(RNNModel, self).__init__(**kwargs)
         self.num_hidden = num_hidden
         with self.name_scope():
            self.rnn = rnn.LSTM(num_hidden, num_layers, input_size=num_embed,
                        bidirectional=bidirectional, layout='TNC')

            self.decoder = nn.Dense(1, in_units=num_hidden)      ## One dimension output

  def forward(self, inputs, hidden):
       output, hidden = self.rnn(inputs, hidden)
       decoded = self.decoder(output.reshape((-1, self.num_hidden)))
      return decoded, hidden

  def begin_state(self, *args, **kwargs):
      return self.rnn.begin_state(*args, **kwargs)

  lstm_model = RNNModel(num_embed=gan_num_features, num_hidden=500, num_layers=1)
```

d. GAN model:



Suggestion from Professor Jafari:

- The detail of the structure is still not clear.
- We need to choose a baseline for the benchmark.
- Create some neat files in Github.

Week4 – 10/5

In this project, the dataset includes 2518 rows and 26 columns (for now).

```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'NASDAQ ', 'NYSE ', 'S&P 500',
       'FTSE100', 'NIKKI225', 'BSE SENSEX', 'RUSSELL2000', 'HENG SENG', 'SSE',
       'Crude Oil', 'Gold', 'VIX', 'USD index', 'MA7', 'MA21', 'MACD', '20SD',
       'upper_band', 'lower_band', 'EMA', 'logmomentum'],
```

The data preprocessing in for this project:

1. Filling the N/A value with the average of previous and next value
2. Normalized the data

```
X_scaler = MinMaxScaler(feature_range = (-1,1))
y_scaler = MinMaxScaler(feature_range = (-1,1))
```

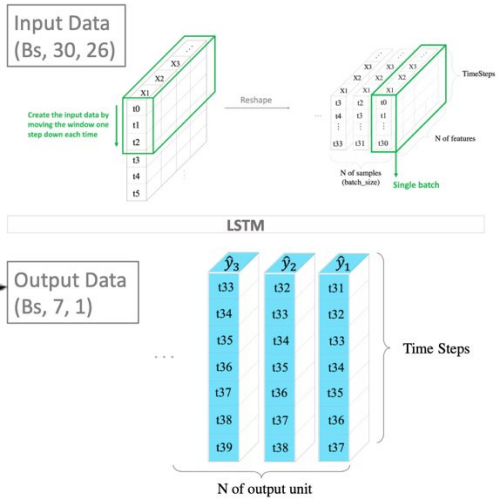
3. Split the data into train and test

Structure:

(In this project, we will input 30 days to predict 7 days)

- LSTM

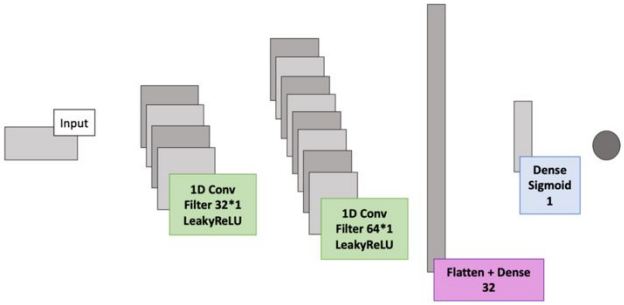
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            24576
_________________________________________________________________
lstm_1 (LSTM)                (None, 16)                5184
_________________________________________________________________
dense (Dense)                (None, 7)                 119
=================================================================
Total params: 29,879
Trainable params: 29,879
Non-trainable params: 0
_________________________________________________________________
```



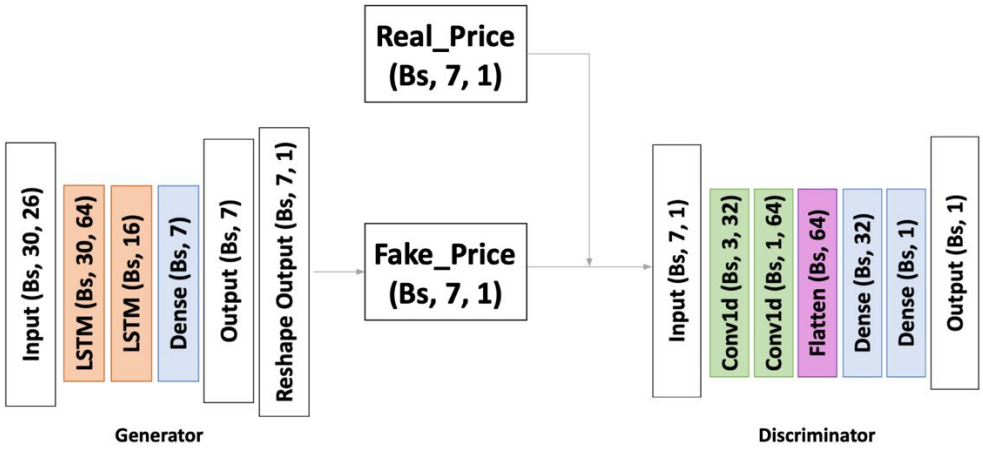
- CNN

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 3, 32)	128
conv1d_1 (Conv1D)	(None, 1, 64)	6208
flatten (Flatten)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2048
dense_2 (Dense)	(None, 1)	33

Total params: 8,417
Trainable params: 8,417
Non-trainable params: 0



- GAN



Future work:

- Feature Engineering:
  - Add one more feature: Add “News” feature through NLP
  - Do the feature selection: XGBoost
- Improvement of GAN model

Week5 – 10/12

This week, we did some feature engineering to extract the features we would like to use in our prediction.

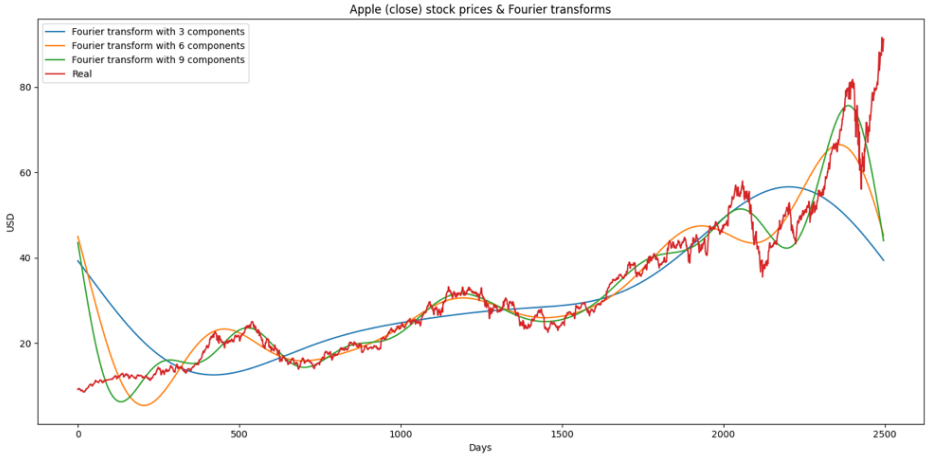
1. Technical Indicator
   After did some technical calculation, we got some new features, **“MA7, MA21, MACD, 20SD, upper_band, lower_band, EMA, logmomentum”**, and after calculation these features, we will drop the first 21 rows which don’t contain the feature MA21.

MA7	MA21	MACD	20SD	upper_band	lower_band	EMA	logmomentum
		-0.20785699999999900				8.874286	2.0636025139345600
		-0.2154812115384620				8.833035250000000	2.056593118151980
		-0.18838493384388000				8.865301461538460	2.064282471365670
		-0.06664999810664090				9.117017425000000	2.108783572855420
		-0.0607472098938846				9.184282619834710	2.106266065562400
		-0.024828310352711800				9.242935997252750	2.1128934839630300
9.069999714285710		-0.025036426858408200				9.206914861848130	2.1027831200831700
9.086938428571430		-0.04675237484041790				9.064187866768300	2.07854826782952
9.116479428571430		-0.0208648504099731				9.038775997866070	2.0826952426492400
9.130867285714290		0.0014540732724004400				8.999829339723620	2.07698314730316
9.086122428571430		-0.013595373951007500				8.949942831630410	2.0700223197633400
9.022296000000000		-0.026781204343379300				8.830457385710520	2.0503621807379400
8.982857428571430		0.02713910983188940 0				8.94086719782202	2.0789504210998600
8.96729614285714		-0.03973327917259260				9.03362241866682	2.089391872533000
9.004132857142860		-0.037062551855838100				9.17835014964193	2.1102997420562900
9.040918428571430		-0.027707747785598300				9.248497384843540	2.1142741556158300
9.080867428571430		-0.03759449330374310				9.256165795007230	2.1114245875328900
9.153214428571430		-0.033243220226003300				9.373007931970660	2.1319662712897100
9.231683714285710		-0.054549338344452600				9.337669310626480	2.1186622548331200
9.263418285714280		-0.06518359999256870	0.18980204783030300			9.258032436852650	2.1063529105213000
9.278775428571430	9.105357095238090	-0.05022519097677150	0.18288518340701800	9.471127462052130	8.739586728424060	9.211010812279720	2.1026086009613700
9.293214285714290	9.128095190476190	-0.047483531899699100	0.17590373453385900	9.479902659543910	8.776287721408470	9.304860937429570	2.1224754082066500
9.303367428571430	9.153588428571430	-0.047158293940562100	0.17037813399824600	9.494344696567920	8.812832160574940	9.33804897914354	2.122817432229600
9.322245000000000	9.177993238095240	-0.05279270100900750	0.1770950862237290	9.532183410542700	8.823803065647780	9.37411165971464	2.127295910986950
9.310102142857140	9.183146333333330	-0.05445277109814660	0.18102716127822000	9.54520065588977	8.821092010776890	9.35565655323819	2.121833782717490 0
9.30566342857143	9.186547714285710	-0.04937607849698900 0	0.18150817764700700	9.549564069579730	8.823531358991700	9.311171517746050	2.1149207690038800

2. Fourier Transform

We calculated these features to generalize several long- and short-term trends. Using these transforms we will eliminate a lot of noise (random walks) and create approximations of the real stock movement.

absolute of 3 comp	angle of 3 comp	absolute of 6 comp	angle of 6 comp	absolute of 9 comp	angle of 9 comp
39.39496894499040	-0.057978620854961700	45.06110110989400	-0.046991430007473100	43.584984434410000	-0.02483714466587250
39.29506496517760	-0.05809923513918570	44.78361321431910	-0.04751562868426270	43.11460480662600	-0.025025251115793500
39.19506807061120	-0.058217280298072500	44.50528889092010	-0.04803637408467220	42.64409519435040	-0.02520460657596860
39.094980604234200	-0.058332735635508700	44.226154656325500	-0.048553549822598800	42.17355069630410	-0.025374912644689900
38.99480491127750	-0.058445580366563500	43.94623713452310	-0.0490670373075501	41.70306621355240	-0.02553586271982320
38.894543339187900	-0.058555793617434100	43.66556305320790	-0.04957671568403900	41.23273642068580	-0.025687141736631500
38.794198237555800	-0.05866335442540880	43.38415924011920	-0.050082461769343600	40.762655737130600	-0.025828425898402000
38.693771958042700	-0.058768241738838000	43.10205261936530	-0.05058414998957070	40.292918298602600	-0.025959382399791100



3. News (Sentiment analysis)

For collecting the news, we extracted the news from the website: Seeking Alpha, and used Finbert pretrained model (particular for finance data)

Date	Article Title
2020/6/30	Apple Arcade cancels games in strategy shift - Bloomberg
2020/6/30	Shipment estimates cut for Apple's 5G iPhones - Digitimes
2020/6/29	NYT pulls out of Apple News partnership
2020/6/29	Apple leaving adapters out of iPhone 12 box - analyst
2020/6/27	Apple seen benefiting from chips play
2020/6/26	DOJ's Apple probe focusing on App Store payment rules - Bloomberg
2020/6/25	Apple re-closing 14 stores in Florida
2020/6/25	Apple closes more stores amid spike in COVID-19 cases
2020/6/24	UBS reviews names to watch in 'consumerization of healthcare'
2020/6/24	Apple roundup: Acquires Fleetsmith, iPhone sales drop in China
2020/6/24	Apple antitrust probe possible from DOJ, state AGs
2020/6/23	Big 5 tech stocks rallying further to new highs
2020/6/23	Apple targets raised on WWDC, iPhone optimism
2020/6/22	Apple WWDC: Macs leave Intel for custom silicon
2020/6/22	Apple's chip will outperform Intel's Macs - analyst
2020/6/22	Sonos to get acquired? Apple may be lurking, Citron says
2020/6/22	What's happening at Apple WWDC 2020?
2020/6/19	Reopening-trade stocks stung as new COVID worries mount
2020/6/19	Apple may announce Intel breakup Monday - NYT
2020/6/19	Apple closing some U.S. stores as coronavirus spikes
2020/6/19	Microsoft calls for App Store antitrust investigations
2020/6/19	Stocks jump early; oil tops $40/barrel
2020/6/19	Apple gets another price target boost from 5G iPhone
2020/6/18	Apple assembling new iPhone SE in India - The Information
2020/6/18	France calls U.S. withdrawal from digital tax talks a 'provocation'

Then did the Sentiment analysis and giving score to the news (positive, neutral, negative).

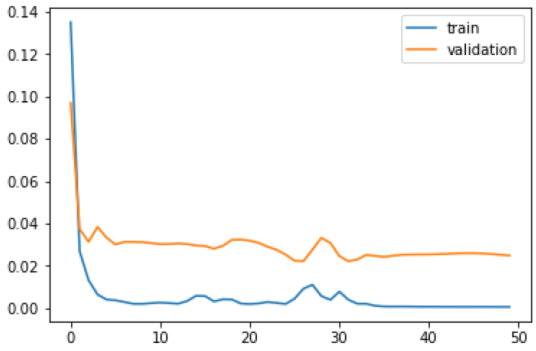
Date	Sentimen Score	Article Title
2020/6/30	-0.918081701	Apple Arcade cancels games in strategy shift - Bloomberg
2020/6/30	-0.888130665	Shipment estimates cut for Apple's 5G iPhones - Digitimes
2020/6/29	-0.885745525	NYT pulls out of Apple News partnership
2020/6/29	-0.825985014	Apple leaving adapters out of iPhone 12 box - analyst
2020/6/27	0.847903252	Apple seen benefiting from chips play
2020/6/26	-0.084378615	DOJ's Apple probe focusing on App Store payment rules - Bloomberg
2020/6/25	-0.912210941	Apple re-closing 14 stores in Florida
2020/6/25	-0.709997535	Apple closes more stores amid spike in COVID-19 cases
2020/6/24	0.006058903	UBS reviews names to watch in 'consumerization of healthcare'
2020/6/24	-0.966615498	Apple roundup: Acquires Fleetsmith, iPhone sales drop in China
2020/6/24	-0.144765303	Apple antitrust probe possible from DOJ, state AGs

Results:

- **Basic LSTM**

(LR = 0.001, BATCH_SIZE = 64, N_EPOCH = 50) -> **RMSE = 6.55**

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (None, 64)                24832
_________________________________________________________________
dense_15 (Dense)             (None, 7)                 455
=================================================================
Total params: 25,287
Trainable params: 25,287
Non-trainable params: 0
_________________________________________________________________
```

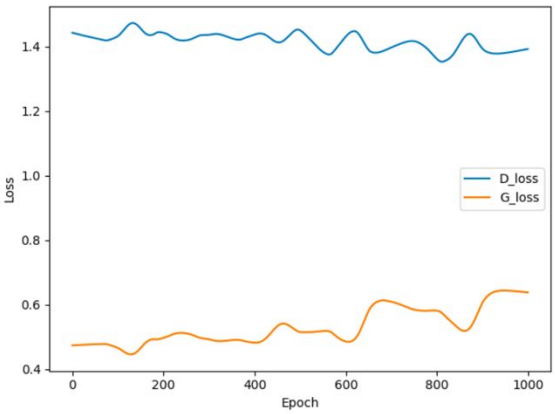


- **Basic GAN**

(We are still tuning the parameters)

Generator: LSTM

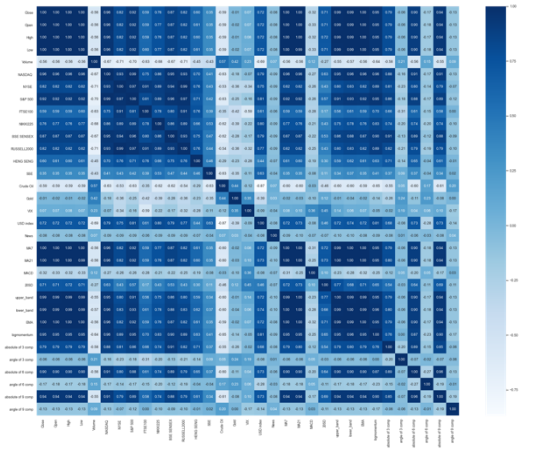
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	25088
dense (Dense)	(None, 7)	455

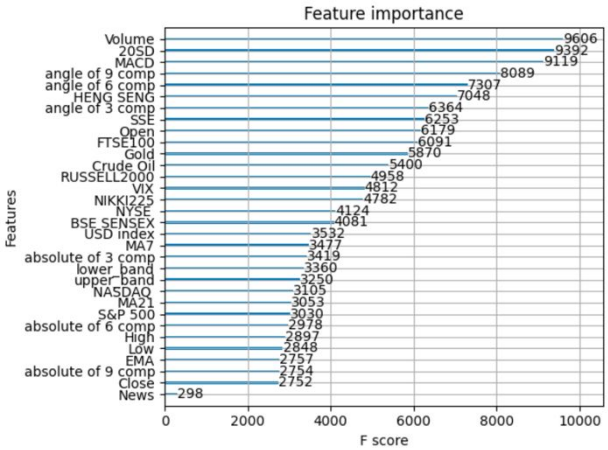


Discriminator: CNN

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 3, 32)	128
conv1d_1 (Conv1D)	(None, 1, 64)	6208
flatten (Flatten)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2048
dense_2 (Dense)	(None, 1)	33

The result of features selection:





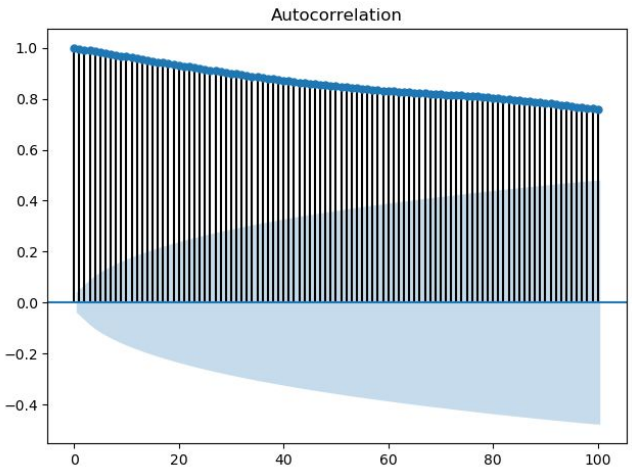
Future work:

- Improving GAN structure

- Working on plot predicted result and the RMSE
- Working on WGAN

Suggestions from Professor Jafari:
- About the feature selection, we should do the check of autocorrelation.

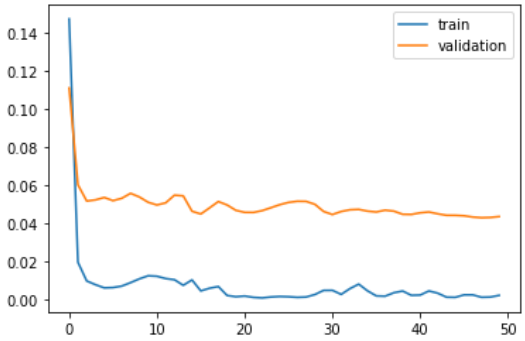
Week6 – 10/19 (Mid-presentation)
From the last week, about the suggestion of doing the autocorrelation check.
We did the statistical check:



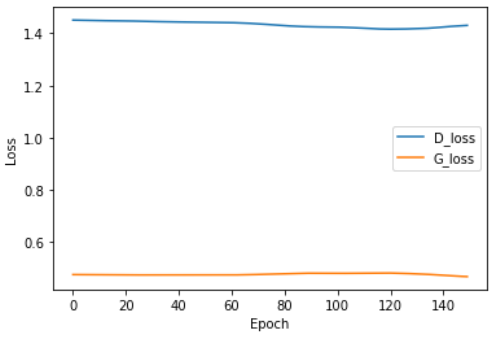
This is the plot of Autocorrelation for target value, and we found out that previous stock price has influence when making predict, so we decided to use previous close price as a feature in the model.

In this week, we tried the LSTM with bidirectional layer, and the result looks better than basic LSTM layer.
- **RMSE = 2.75**

Layer (type)	Output Shape	Param #
bidirectional_2 (Bidirection	(None, 128)	51712
dense_4 (Dense)	(None, 7)	903



We have got some result of GAN:



The result of GAN for now looks not good as LSTM, from the loss plot, we can say the generator and discriminator both seem not learning, and from the training result plot, we can see there are two spikes at the beginning and the end.
We should find out where are the spikes from and keep improving the model.

In this week, we also tried to do the hyper parameters tuning, we did the tuning through Bayesian optimization.

```
pbounds={'lr': (0.0001, 0.0008),
         'epoch': (100, 300),
         'bs': (64, 512)},
```

This time, we tuned the learning rate, epoch and batch size, in the future, we will try to tune the strides and kernel size as well.

Here is the result of our benchmark:

	RMSE
Basic LSTM	2.75
GAN	4.91
WGAN	In progress
WGAN-GP	In progress

We do encounter some challenges in our project, the first one is that, the loss in GAN training process looks not good, because as normal, we though that the G_loss should be larger than D_loss, and it may indicate that our Discriminator is too weak; Second, the hyperparameter tuning through Bayesian optimization seems not improving our result; the last one is that the RMSE of GAN is still larger than basic LSTM.

Future work:

- Keep working on WGAN and WGAN-GP
- Adjust CNN model structure to improve the Discriminator
- Work on the Bayesian Optimization

Suggestion from Professor Jafari:

- The loss of G is not necessarily greater than loss of D, since we are not using the normal GAN, we put the RNN into our model, it might be a different situation as usual.
- We should find out the reason of the spikes, if remove the spikes, the RMSE of GAN model may be good.

Week7 – 10/26

This week, we tried to solve some problems we have encountered.
There are about the spikes appear at the beginning and the end, our GAN model is unstable and the generator and discriminator seem not learning.

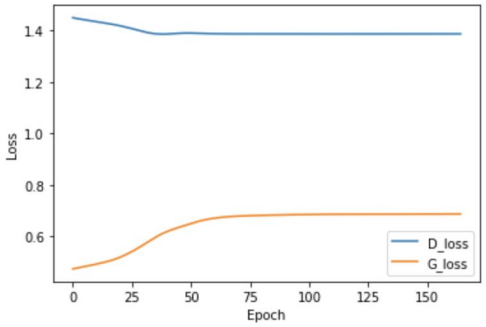
To solve these problems, we tried to

1. Reduce output steps first, we would like to check if the problem comes from the too many output steps.
2. Set MLP as our discriminator
3. Add batch normalization, dropout, initializer in CNN
4. Adjust the generator

From the first try, change the output steps, we thought the strikes might come from too many output steps, so we tried to change the output steps from 7 to 3

- **RMSE = 5.56**

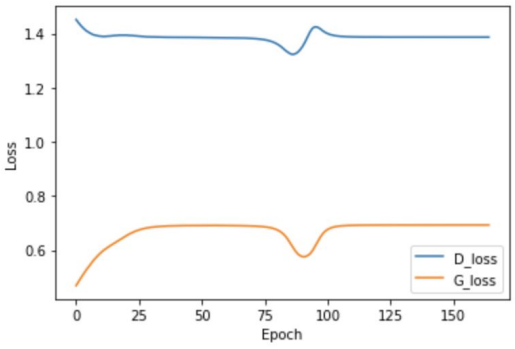
```
Model: "Generator-LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_5 (Bidirection (None, 128)               51712
_________________________________________________________________
dense_20 (Dense)             (None, 3)                 387
=================================================================
```



The spikes phenomenon didn't be solved, and also the RMSE didn't reduce, so the problem may not come from the output steps.

Second, we tried to put the MLP as discriminator according to a paper we found, but the result was even worse.

- **RMSE = 27.62**



But from this paper, we found out that it looks normal that D loss is larger than G loss in this case, but D loss is better to decrease through the process.

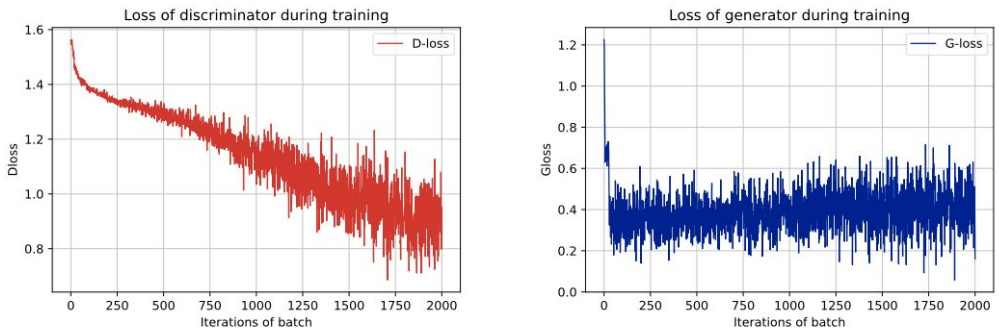


Fig. 4. Losses of the discriminator and generator during training.

*Loss plots from the paper
Reference:
https://reader.elsevier.com/reader/sd/pii/S1877050919302789?token=B294C2B4282B0B6792946527DDCE625AEE6A7B868F7C96A5BE9C7BD65239880760F8E36BA0BD3EDE90A274911BA2D83D

Then we tried to set Batchnorm(), Dropout() and Initializer() into our discriminator, and it turns out not working.

Last thing we did in this week, which did improve our result a lot is that we adding one more layer in our generator, and also increased the neuron numbers.
Previous, we through that all the problems come from the discriminator is too weak, but while we tried to adjust the generator, the result did get improved.

(*All the model we ran this time, the output step was 3.)

- On the **Basic GAN**:

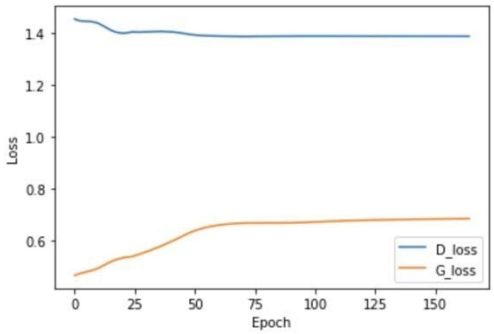
Previous generator:

```
Model: "Generator-LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_5 (Bidirection (None, 128)               51712
_________________________________________________________________
dense_20 (Dense)             (None, 3)                 387
=================================================================
```

New generator:

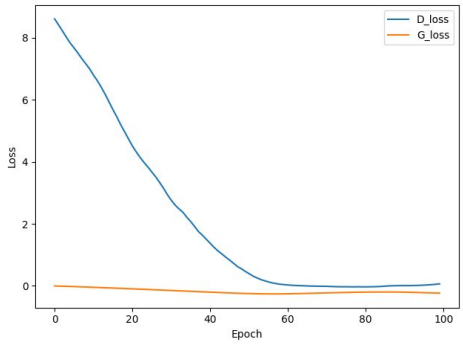
```
Model: "Generator - new"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_10 (Bidirectio (None, 512)               600064
_________________________________________________________________
dense_32 (Dense)             (None, 128)               65664
_________________________________________________________________
dense_33 (Dense)             (None, 3)                 387
=================================================================
```

From this model, we can see the spikes disappeared.



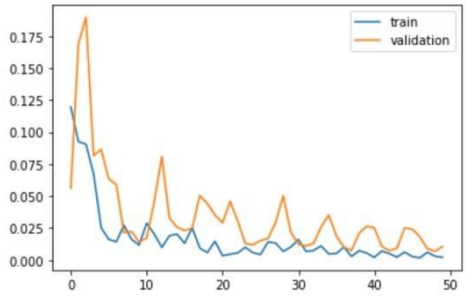
- **And the RMSE = 2.57**

- On the **WGAN-GP**



- **RMSE = 1.80**

- **Basic LSTM**



- **RMSE = 3.51**

Conclusion for this week:

- The strikes disappear
- The result is much more stable than previous version, the RMSE range is about 1.5 to 6

- more complicated LSTM in GAN help the model perform better.
- WGAN model can resolve the problem of D is not learning, and the result is better than basic LSTM and basic GAN

Future work:
- Continue to improve the GAN model
- Find a way to make the model much more stable
- Try to predict different output days

Suggestion from Professor Jafari:
- We should try to use GRU but not LSTM, since LSTM is more complicated than GRU, so it might be unstable.
- We should try to use basic GRU, and then build up a larger model.
- Using one directional, not use bidirectional.
- Try to predict only the last one test dataset.

Week8 – 11/02

We tried to set the GRU in our generator, and the result did improve.
(*For the models we ran this time, input = 30 days, output = 3 days.)

- **Basic GAN**

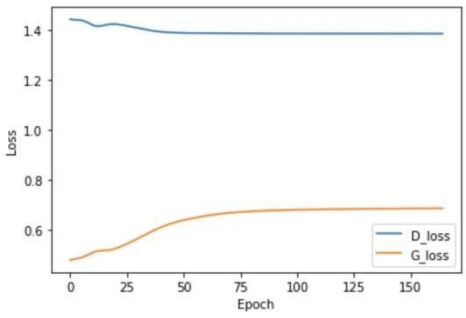
The structure:

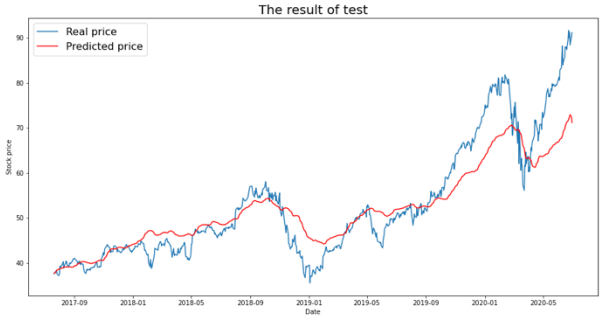
Layer (type)	Output Shape	Param #
gru_44 (GRU)	(None, 30, 1024)	3262464
gru_45 (GRU)	(None, 30, 512)	2362368
gru_46 (GRU)	(None, 256)	591360
dense_78 (Dense)	(None, 128)	32896
dense_79 (Dense)	(None, 64)	8256
dense_80 (Dense)	(None, 3)	195

Layer (type)	Output Shape	Param #
conv1d_36 (Conv1D)	(None, 17, 32)	128
conv1d_37 (Conv1D)	(None, 9, 64)	10304
conv1d_38 (Conv1D)	(None, 5, 128)	41088
flatten_12 (Flatten)	(None, 640)	0
dense_81 (Dense)	(None, 220)	140800
leaky_re_lu_51 (LeakyReLU)	(None, 220)	0
dense_82 (Dense)	(None, 220)	48400
dense_83 (Dense)	(None, 1)	221

The result of GRU improve a lot from LSTM

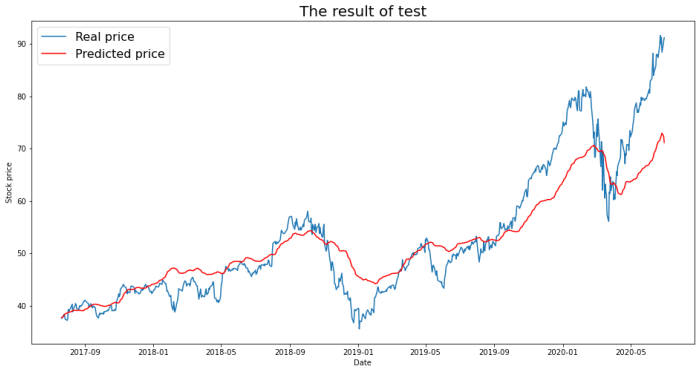
The result of training dataset:





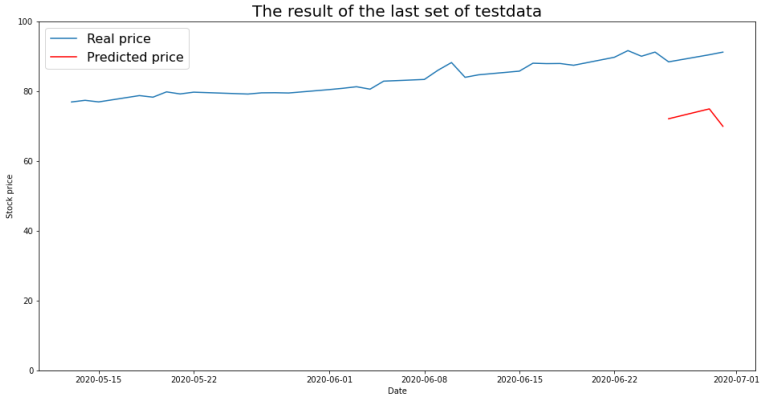
- **RMSE = 2.07**

The result of test dataset:



- **RMSE = 5.54**

The result of predicting last three days:

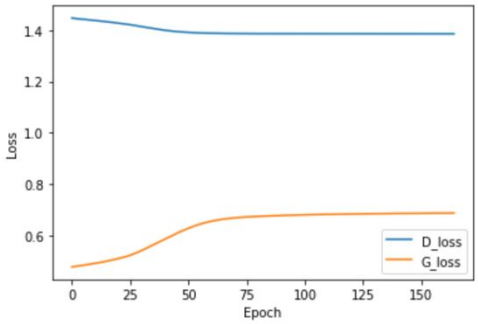


Date	Predicted_price	Real_price
2020-06-26	72.10	88.41
2020-06-29	74.94	90.45
2020-06-30	69.94	91.20

- **RMSE = 17.88**

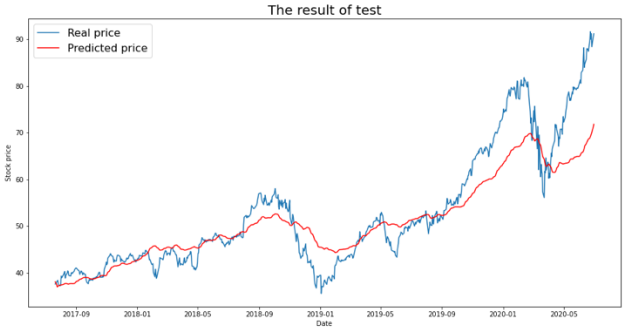
- **Basic GAN – Train G 2 times**

The result of training dataset:



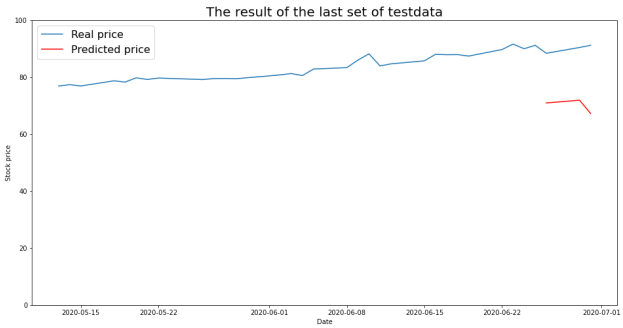
- RMSE = 1.64

The result of test dataset:



- **RMSE = 5.88**

The result of predicting last three days:



Date	Predicted_price	Real_price
2020-06-26	70.96	88.41
2020-06-29	71.93	90.45
2020-06-30	67.26	91.20

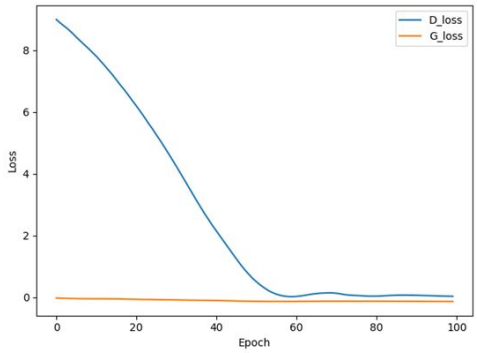
- RMSE = 20.17

For Basic GAN, the training times of G seems has no obvious effect on the result.

- **WGAN-GP**

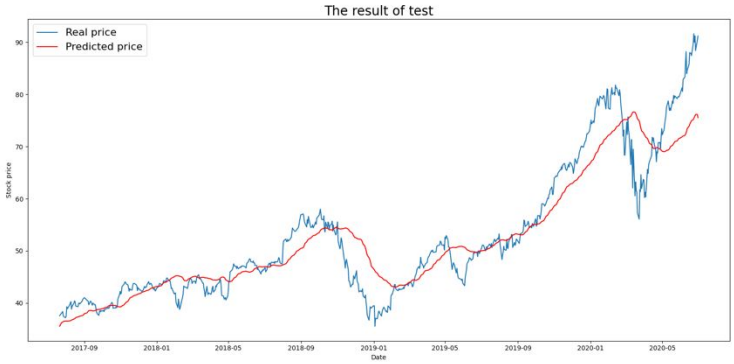
For WGAN-GP, we trained the generator 3 times and discriminator 1 time.

The result of training dataset:



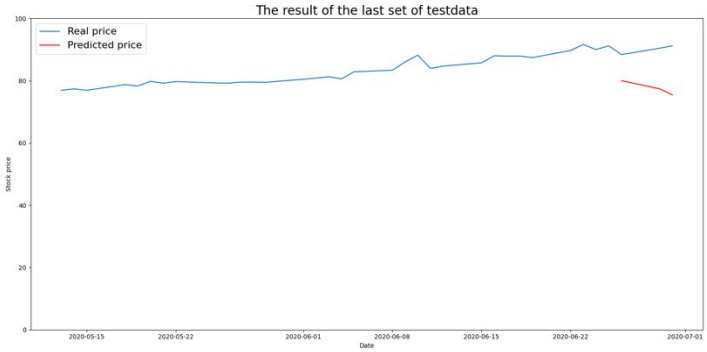
- RMSE = 1.89

The result of test dataset:



- RMSE = 4.77

The result of predicting last three days:

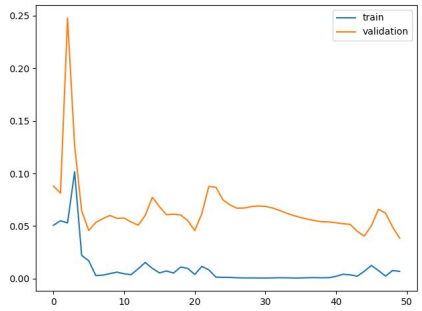


Date	Predicted_price	Real_price
2020-06-26	80.02	88.41
2020-06-29	77.40	90.45
2020-06-30	75.45	91.20

- RMSE = 12.75

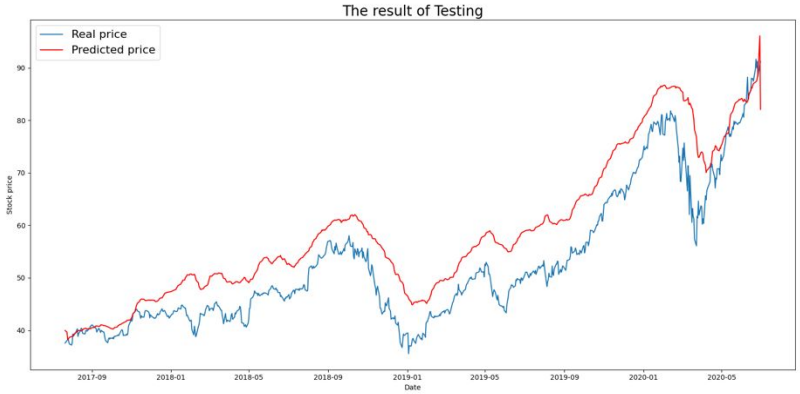
- **Baseline model – LSTM**

The result of training dataset:



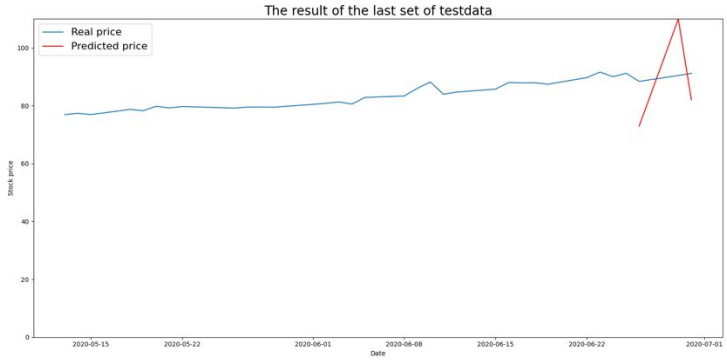
- RMSE = 1.52

The result of test dataset:



- RMSE = 6.60

The result of predicting last three days:



Date	Predicted_price	Real_price
2020-06-26	103.33	88.41
2020-06-29	94.21	90.45
2020-06-30	105.05	91.20

- RMSE = 11.95

Model comparison:

Train dataset

	Basic LSTM	Basic GAN	Basic GAN - 2G	WGAN-GP
RMSE	1.52	2.07	1.64	1.89

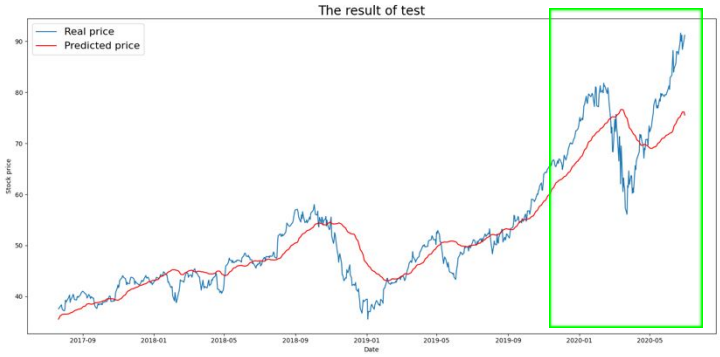
Test dataset

	Basic LSTM	Basic GAN	Basic GAN - 2G	WGAN-GP
RMSE	6.60	5.54	5.88	4.77

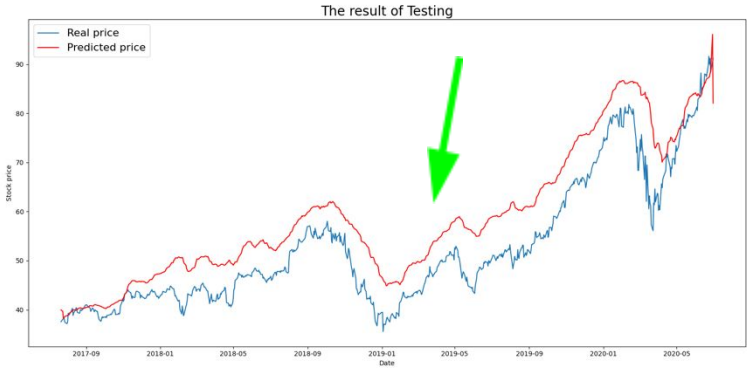
The last three days

	Basic LSTM	Basic GAN	Basic GAN - 2G	WGAN-GP
RMSE	11.95	17.88	20.17	12.75

There is a special phenomenon of the last three days prediction we found out from our result: The prediction of the last three days through **GAN** is very inaccurate, this might cause from unpredictable sharp drop and growth which due to the COVID-19.



On the other hand, the prediction of the last three days through **Basic LSTM** is more accurate, but as it can be seen from the plot, the prediction of LSTM is always much higher than the real price, which makes the prediction of the recent period more accurate.



The problem for this week:

- GAN model did not perform better than Baseline model and model still need to be improved

Question for the professor:

- WGAN-GP best model (train RMSE 1.89, test RMSE 4.77) when train RMSE smaller than 1.89 test RMSE will increase, overfitting?

Suggestion from Professor Jafari:

- For our question, it's hard to say if it is overfitting or not.
- In our test dataset, we can remove the data of 2020, and see if the GAN model performs better than the base line model on predicting the last three days.
- How does the LSTM know about our dataset has time series between each sample?
- We can try to build up a model for input = 3 days and output = 3 days.

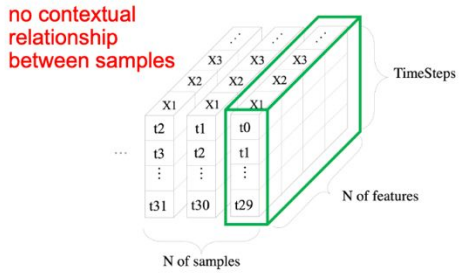
Week9 – 11/09

This week, we focused on understanding how the model knows our dataset does have a time-series relation with each sample.

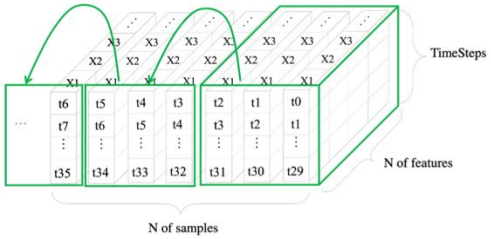
And for doing the time-series prediction with LSTM, there is a parameter called stateful, we can build up a stateful model or a stateless model.
The concept is that, **stateful** connections (state) between sequences(samples) would be retained. It can enable the model to learn the timing characteristics between samples. Which sample is in the first place and which sample is in the second place will have an impact on the model; **Stateless** during training, the state is reset after each sequence, state isn't retained between the sequences(samples), Which can be said to be independent of each sample, with no contextual relationship between them.

For splitting the original data to stateful or stateless data.

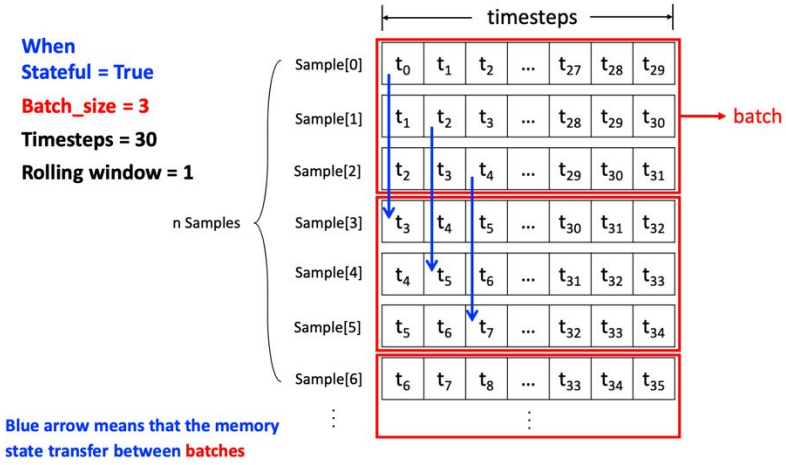
Stateless:



Stateful:



The concept of stateful data:



For building up the stateful model, we need to adjust the dimension of the input of the generator, and also need to adjust the input of discriminator.

Generator:

```
def Generator(input_dim, output_dim, feature_size, batch_size) -> tf.keras.
    model = Sequential()
    model.add(GRU(units=512,
                  return_sequences=True,
                  batch_input_shape=(batch_size, input_dim, feature_size),
                  stateful=True,
                  recurrent_dropout=0.01,
                  recurrent_regularizer=regularizers.l2(1e-3)))
    model.add(GRU(units=126,
                  #return_sequences=True,
                  stateful=True,
                  recurrent_dropout=0.01,
                  recurrent_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(64, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(units=output_dim))
    return model
```

Discriminator:

```
def Discriminator(batch_size, input_dim) -> tf.keras.models.Model:
    model = tf.keras.Sequential()
    model.add(Conv1D(32, batch_input_shape=(batch_size, input_dim, 1), kernel_size=
    model.add(Conv1D(64, kernel_size=3, strides=2, activation=LeakyReLU(alpha=0.01)
    model.add(BatchNormalization())
    model.add(Conv1D(128, kernel_size=3, strides=2,  activation=LeakyReLU(alpha=0.0
    model.add(BatchNormalization())
    model.add(Flatten())
    model.add(Dense(220, use_bias=True))
    model.add(LeakyReLU())
    model.add(BatchNormalization())
    model.add(Dense(64, use_bias=True))
    model.add(ReLU())
    model.add(Dense(1))
    return model
```

Future work:

- Working on the code with stateful model
- Pretrain the model for Generator and Discriminator

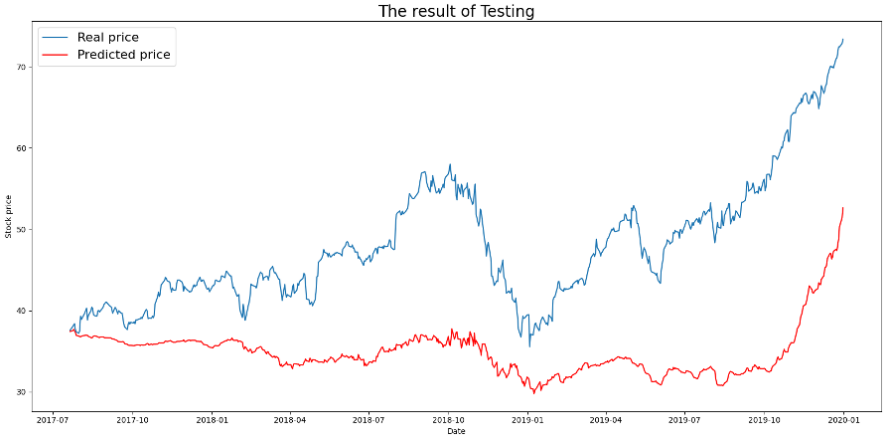
Suggestions from Professor Jafari:

- From our structure, it looks like we are doing the input = 3 steps, we can try to do input = 3.

Week 10 - 11/16

This week we tried to set stateful = true in LSTM and pretrained the model as a generator. Both of them have not improved our model.

Baseline model - LSTM



RMSE: 15.93 (include 2020) RMSE: 14.74 (remove 2020 data)
The result was worse, so we decided not to use stateful in the GAN model.

Pre-trained model

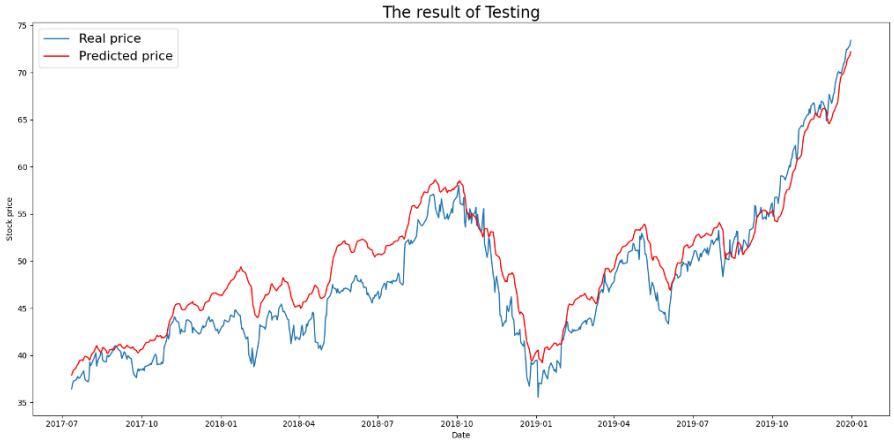
```
def build_encoder(input_dim, output_dim, feature_size) -> tf.keras.models.Model:
    model = Sequential()
    model.add(GRU(units=512, return_sequences=True, input_shape=(input_dim, feature_
    model.add(GRU(units=256, return_sequences=True, recurrent_dropout=0.01, dropout_
    model.add(GRU(128, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(64, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(32, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(16, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(units=output_dim))
    model.compile(optimizer=Adam(lr=LR), loss='mse')
    return model

en = build_encoder(input_dim, output_dim, feature_size)
en.fit(X_train, y_train, epochs=N_EPOCH, batch_size=BATCH_SIZE,
       validation_data=(X_test, y_test), verbose=2, shuffle=False)
en.save("gru_encoder.h5")
```

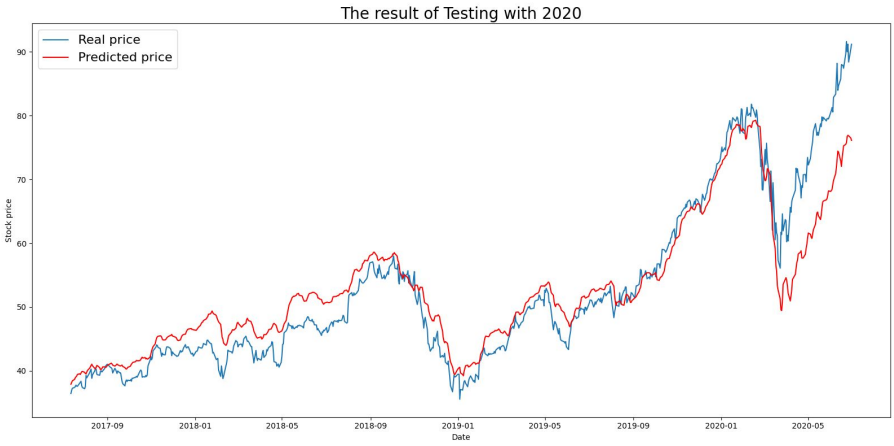
```python
def Generator(en, output_dim):
    inter_output_model = Model(inputs=en.input, outputs=en.layers[-4].output)
    inter_output_model.trainable = False

    model = Sequential()
    model.add(inter_output_model)
    model.add(Dense(64, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(32, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(units=output_dim))
    return model
```

So we will use our previous model as our Final model, We set GRU as a generator and CNN as a discriminator. And we made two types of predictions. Many to one prediction: time step 3, dim(features) 36, output 1 and many to many prediction: time step 30, dim(features) 36, output 3.

- Final model (time step 3, output 1)

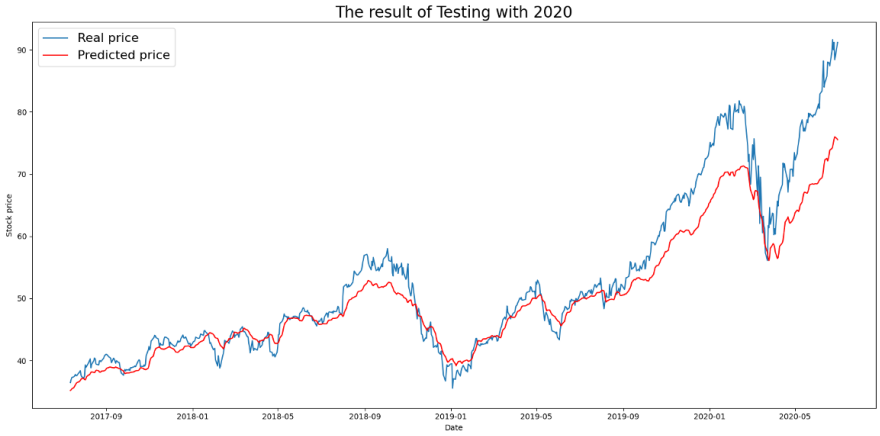
  Baseline model - LSTM

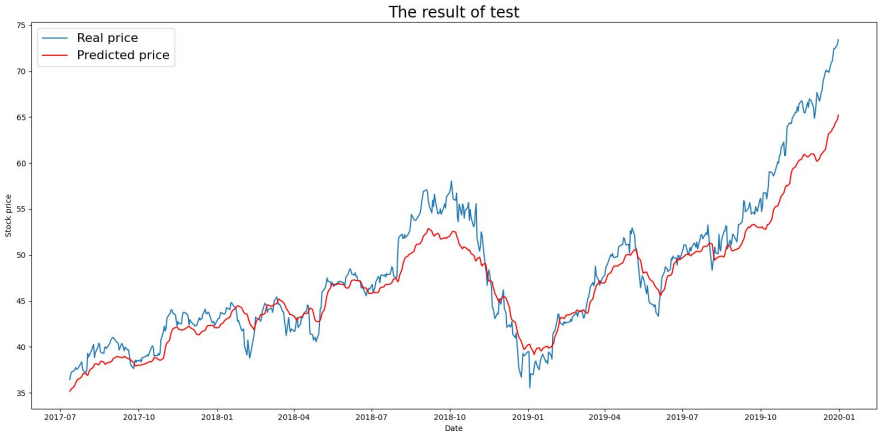




RMSE: 4.72(include 2020)
RMSE: 2.79(remove 2020 data)

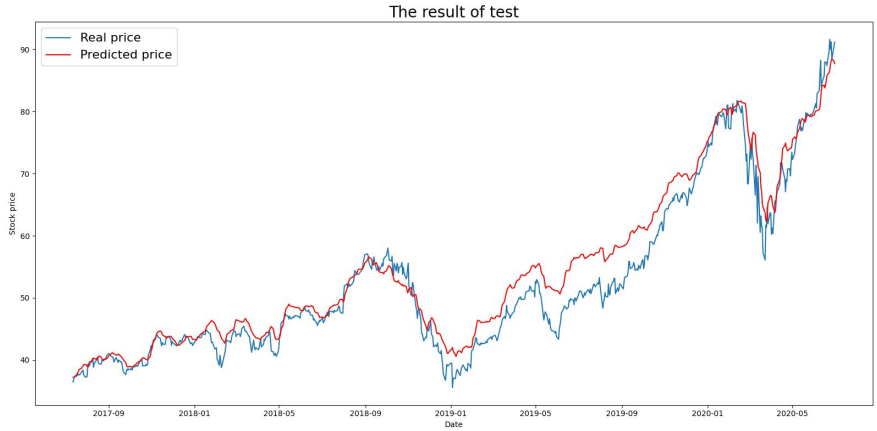
Basic GAN

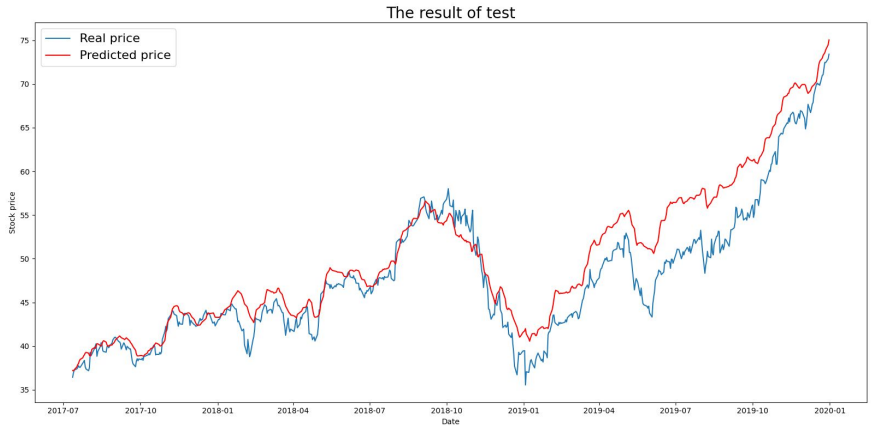




RMSE: 4.67(include 2020)
RMSE: 2.56(remove 2020 data)

WGAN-GP





RMSE: 3.2(include 2020)
RMSE: 3.17 (remove 2020 data)

Model Comparison(time step3, output 1)

Train dataset

	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	3.40	4.03	1.47

Test dataset

	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	4.72	4.67	3.2

Test dataset（ remove year 2020)

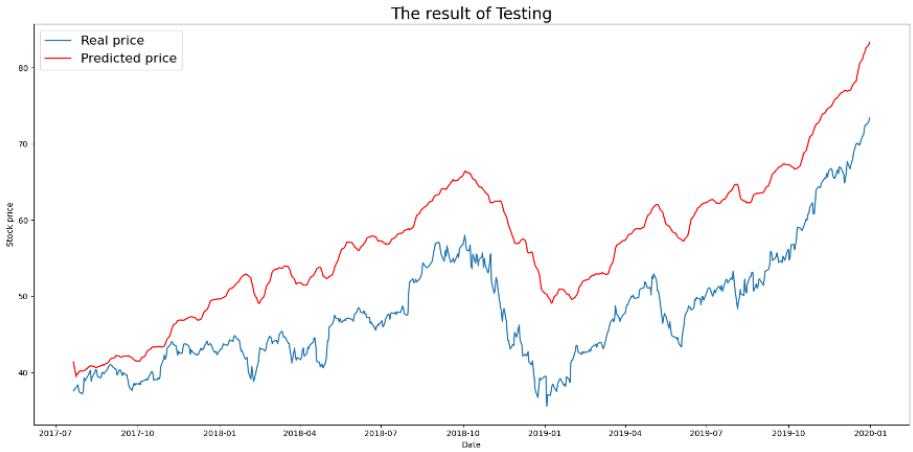
	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	2.79	2.56	3.18

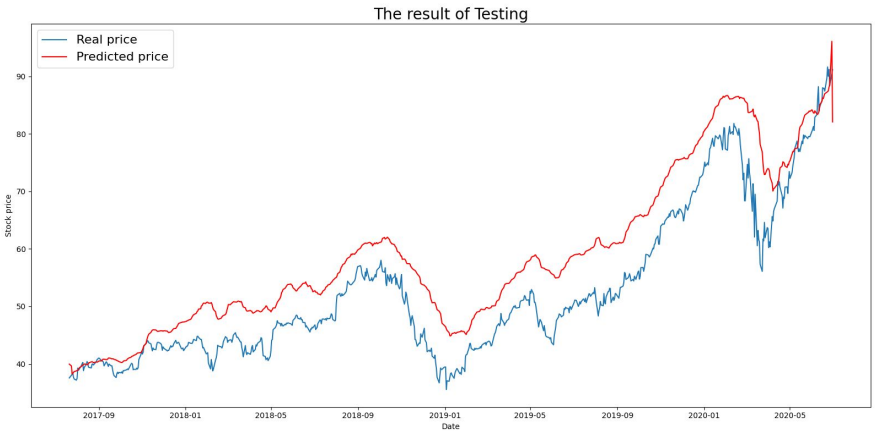
When there are unexpected events like Covid-19, WGAN-GP performs much better.

When the stock market operates normally, Basic GAN performs better.

- Final model (time step 30, output 3)

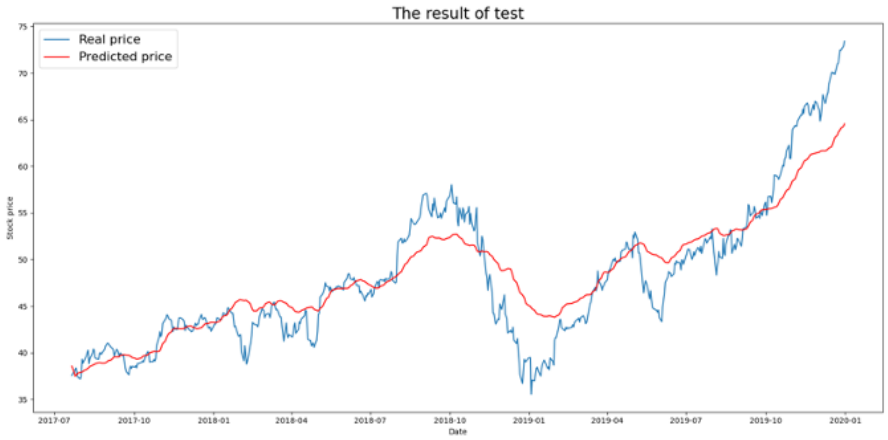
  Baseline model - LSTM

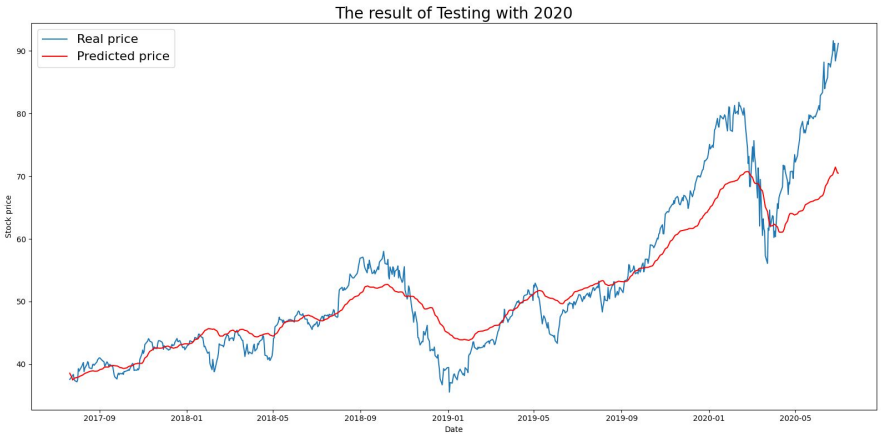




RMSE: 6.60(include 2020)
RMSE: 9.47(remove 2020 data)

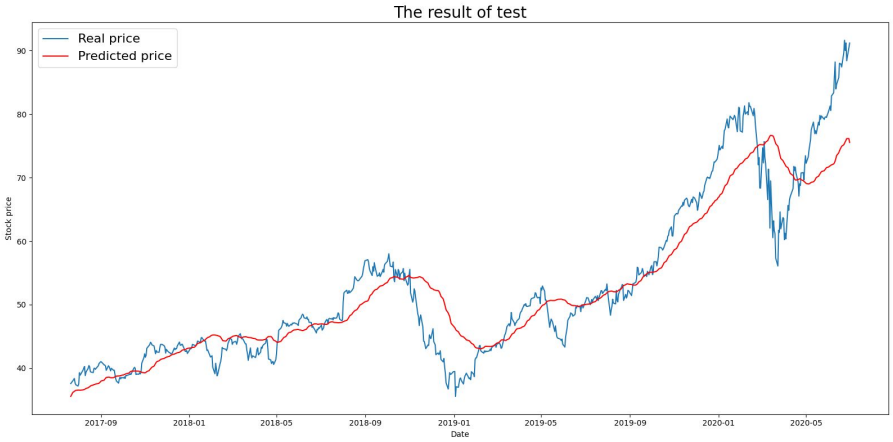
Basic GAN

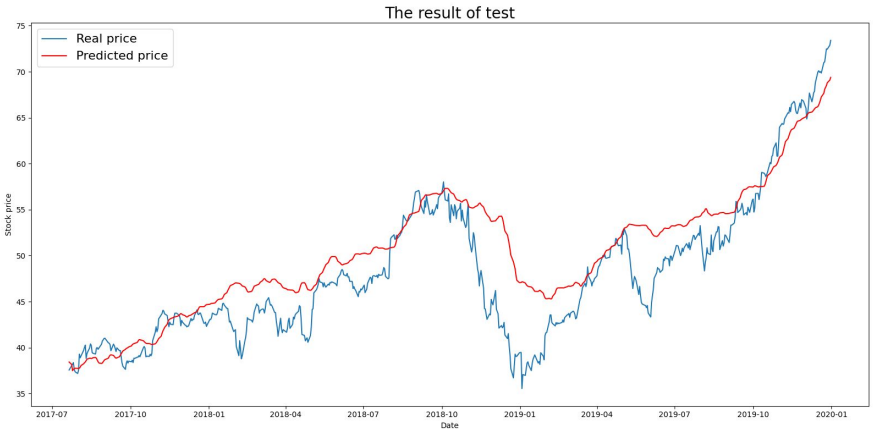




RMSE: 5.36(include 2020)
RMSE: 3.09(remove 2020 data)

WGAN-GP





RMSE: 4.77(include 2020)
RMSE: 3.88 (remove 2020 data)
Model Comparison(time step 30, output3)

Train dataset

	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	1.52	1.64	1.74

Test dataset

	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	6.60	5.36	4.77

Test dataset（remove year 2020)

	Baseline LSTM	Basic GAN	WGAN-GP
RMSE	9.45	3.09	3.88

Many to many predictions, GAN performance on test data is much better than LSTM. When the stock market operates normally,without unexpected extreme events like Covid-19, GAN performance is much better than Baseline model.