AWS Machine Learning Engineer Nanodegree

Capstone Project

Philipp Richter
June 23th, 2022

PROJECT DEFINITION

Project Overview

Information science and math have long been an integral part of investment management, but recently artificial intelligence in the form of deep learning also became a mainstream part of the investment management toolkit for predicting future asset prices and/or returns. These predictions can be based on several kinds of data, like historical prices and volatility, and news articles using NLP ("Natural Language Processing").

Most neural networks work quite well for image data, but for time series data which is essential in finance, most neural network types are not very good. However, there is one type of neural network which is said to perform exceptionally well at predicting time series data and this is LSTM ("Long Short-Term Memory"), and I am courious to find out if that is true, because I have been working in the finance industry for several years and a stock return prediction tool would certainly be quite useful. The main questions is: Can AI beat the market?

Problem Statement

In order to make an investment profit, I would like to predict tomorrows returns on a selection of S&P500 stocks, based on historical returns of the same stock up until today. In order to take into account short, medium and long-term effects, the prediction will be a weighed average of the predictions based on daily, weekly and monthly returns. In order to make a profit, the predicted return and the real return need to have the same sign. However, I will have to define an "inconclusive zone" around 0 which will be treated as the investment recommendation "do not invest today", i.e. neither assume a long nor a short position in the stock. This is because very small predictions like +0.00451% or -0.00235% are very close to each other, but they would result in opposite investment positions (long vs. short). The ideal range of this zone will probably depend on the volatility of the stock. The success of the model will be mainly measured by the percentage of correct investment recommendations (either "take a long position" or "take a short position" or "stay out of the market"). Therefore, this is a classification problem and not a regression

problem, that the ML algorithm has to solve. Furthermore, I will also calculate how successful a strategy based on these recommendations would be in terms of profit.

The solution of the problem is to build 3 parallel LSTM neural networks for each stock and train them on the daily, weekly and monthly returns of the stocks chosen by the user. Short, medium and long-term effects are quite different from each other, so I believe that having one common network for all 3 together would not work well. To take into account the fact that all companies' stocks behave differently, the training will be performed separately for each given ticker. Each network will predict a return for tomorrow and depending on the size and sign, the prediction would be one of the following (as mentioned above):

- 1. Buy stock (long position)
- 2. Keep cash, i.e. do not invest (neutral position)
- 3. Sell short (short position)

Metrics

There are two ways of evaluating the success of the model: 1) The percentage of correct investment recommendations, and 2) the net profit made by applying these recommendations. It might be possible, that the prediction is mostly correct for small returns and incorrect for large returns, which might still look good in terms of the percentage of correct recommendations, but would make the user loose money.

Therefore, the percentage of correct recommendations will be calculated, and then the profit/loss of an investor following these recommendatios will be compared with the profit/loss of investors follow the recommendations from an 18 days simple moving average, which will serve as benchmark.

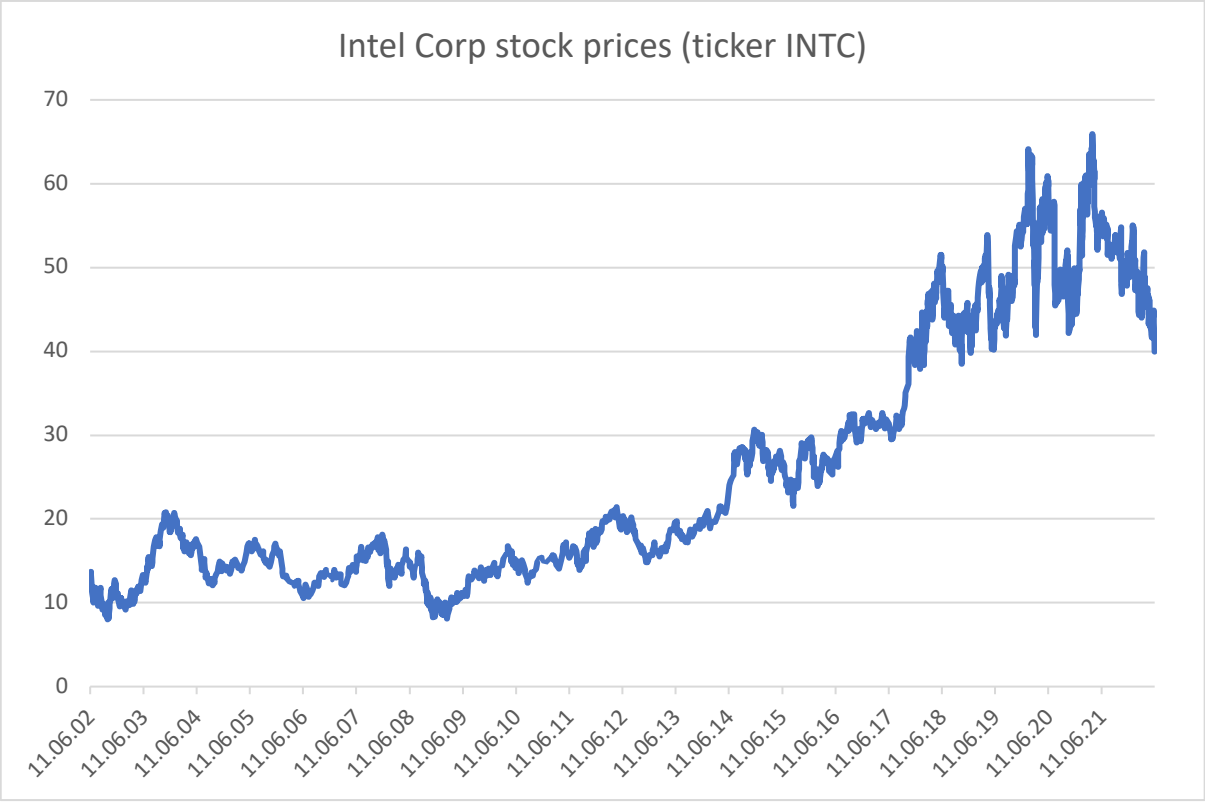
ANALYSIS

Data Exploration and Visualization

All the input data will be obtained from “Yahoo Finance” (https://finance.yahoo.com), where historical stock market quotes can be downloaded free of charge. All available tickers from the S&P500 will be downloaded as CSV file and stored locally. This is how the raw data from Yahoo Finance looks like for Intel Corp. stock (ticker “INTC”, screenshot taken from https://finance.yahoo.com/quote/INTC/history?p=INTC):

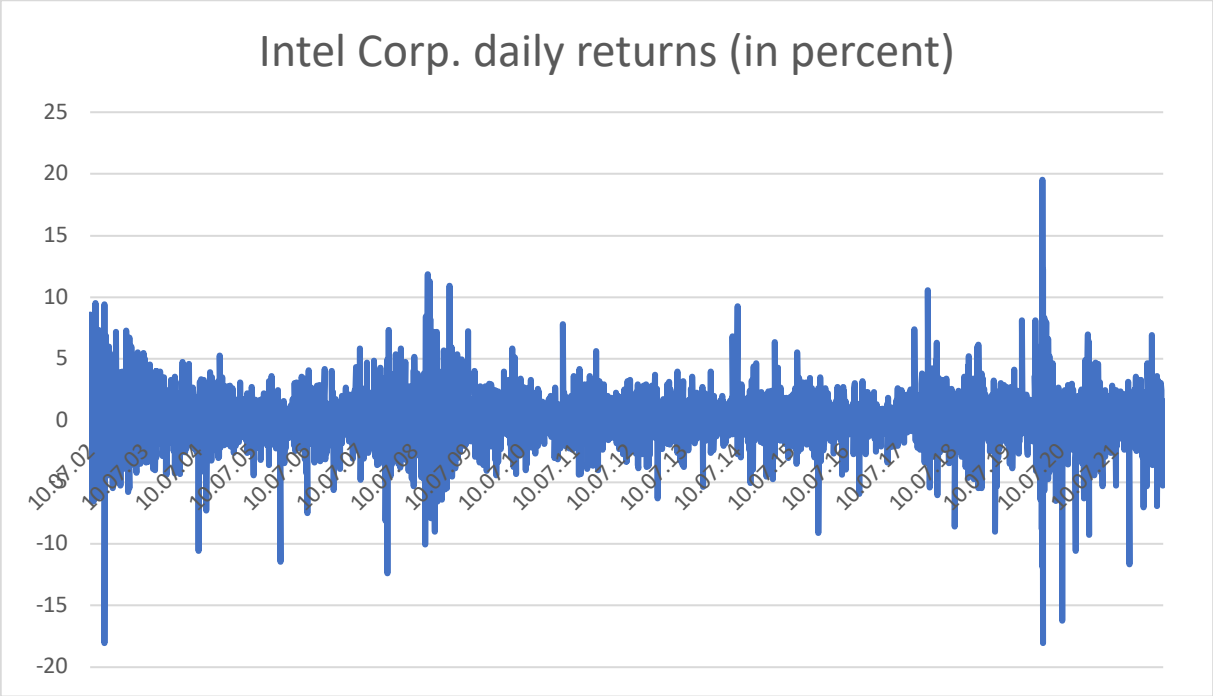
Date	Open	High	Low	Close*	Adj Close**	Volume
Jun 03, 2022	44.11	44.25	43.34	43.39	43.39	33,162,300
Jun 02, 2022	44.19	44.88	43.94	44.84	44.84	28,727,600
Jun 01, 2022	44.77	44.93	43.53	44.11	44.11	29,885,500
May 31, 2022	44.25	44.75	43.65	44.42	44.42	41,111,500
May 27, 2022	43.59	44.55	43.55	44.55	44.55	30,553,300
May 26, 2022	42.17	43.68	42.08	43.48	43.48	28,826,700
May 25, 2022	41.44	42.52	41.39	42.20	42.20	26,283,100
May 24, 2022	41.70	41.89	41.10	41.67	41.67	29,837,500
May 23, 2022	41.69	42.25	41.33	42.00	42.00	27,199,500
May 20, 2022	42.25	42.29	40.31	41.65	41.65	44,780,900

The next graph displays the stock prices for Intel Corp. with ticker INTC, quoted in USD (created in Excel after having downloaded the stock prices of the last 20 years):

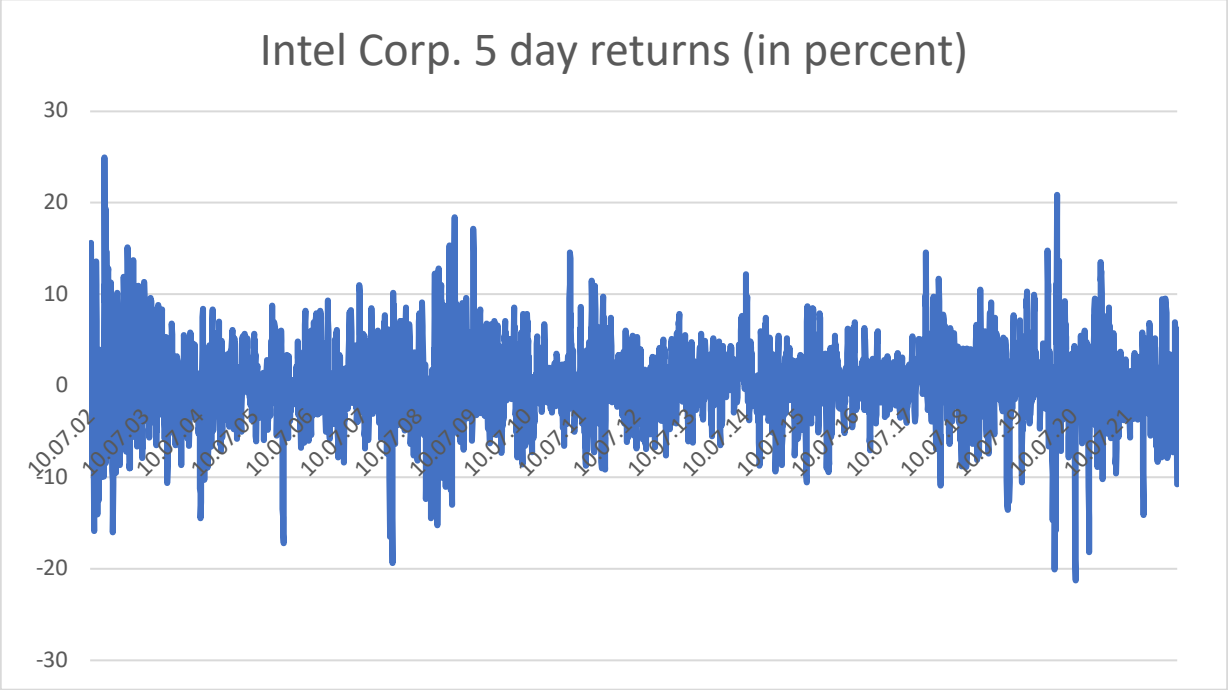


LSTM neural networks are said to have problems with data which contains trends, so I will avoid this problem by focussing on the returns, i.e. daily, 5-day and 20-day returns (trading days, not calendar days) which corresponds about to daily, weekly and monthly returns.

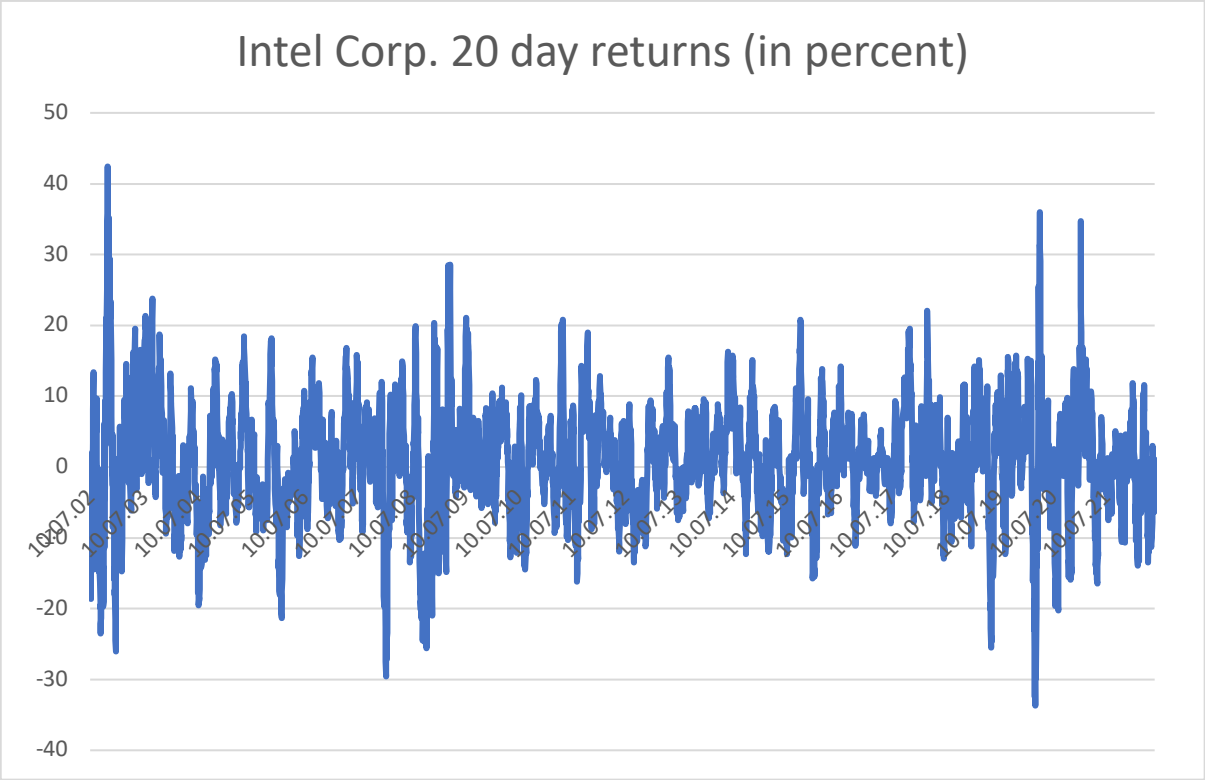
These are the daily returns for Intel Corp. with ticker INTC (created in Excel after having downloaded the stock prices of the last 20 years):



These are the weekly (5-day) returns for Intel Corp. with ticker INTC (created in Excel after having downloaded the stock prices of the last 20 years):



These are the monthly (20-day) returns for Intel Corp. with ticker INTC (created in Excel after having downloaded the stock prices of the last 20 years):



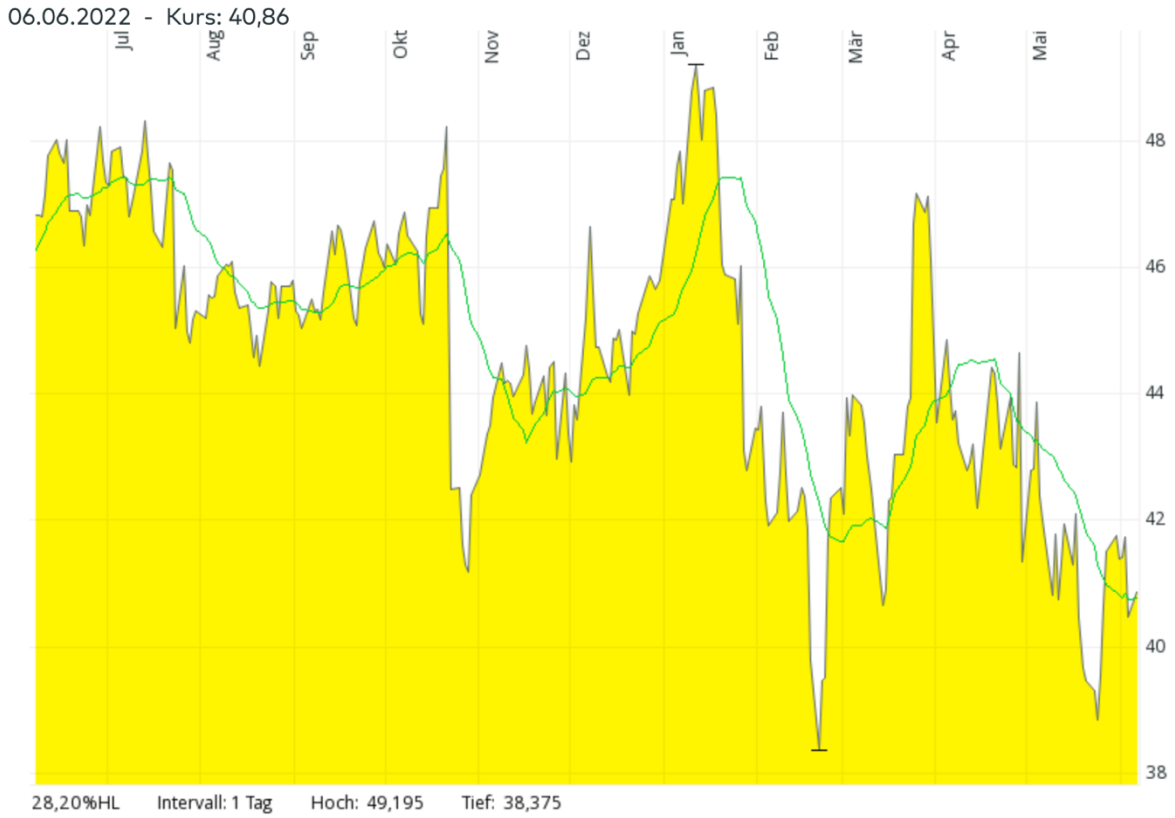
All three graphs look resonable, and the stock market crisises in 2001/2002, 2007 and 2020 can be easily identified.

Benchmark

As a benchmark for comparing the model, an 18 days simple moving average of the daily returns will be used (and also random investing, but this cannot really be explored/visualized). The same 3 types of investment recommendation will be generated and compared to the LSTM model.

To better visualize the concept of the 18 day simple moving average, the INTC ticker with its 18 days simple moving average of the stock price (i.e. not the returns) are displayed in the screenshot below (taken from Comdirect Bank, https://www.comdirect.de/inf/aktien/US4581401001, the green line is the 18 days simple moving average). Note: The graph is based on the historical prices, while the benchmark and the LSTM model will be based on the historical returns. I chose to display the stock price, because using the return, the visual representation would not be helpful and pretty messy. The time interval is only one year because with 20 years, the 18 say simple moving average would be pretty much indentical with the stock price itself and therefore the visualization would not be very helpful.

It is also important to realize, that the 18 day simple moving average lags behind the stock for a couple of days, because changes in stock price will only slowly change the simple moving average.



METHODOLOGY

Data Preprocessing

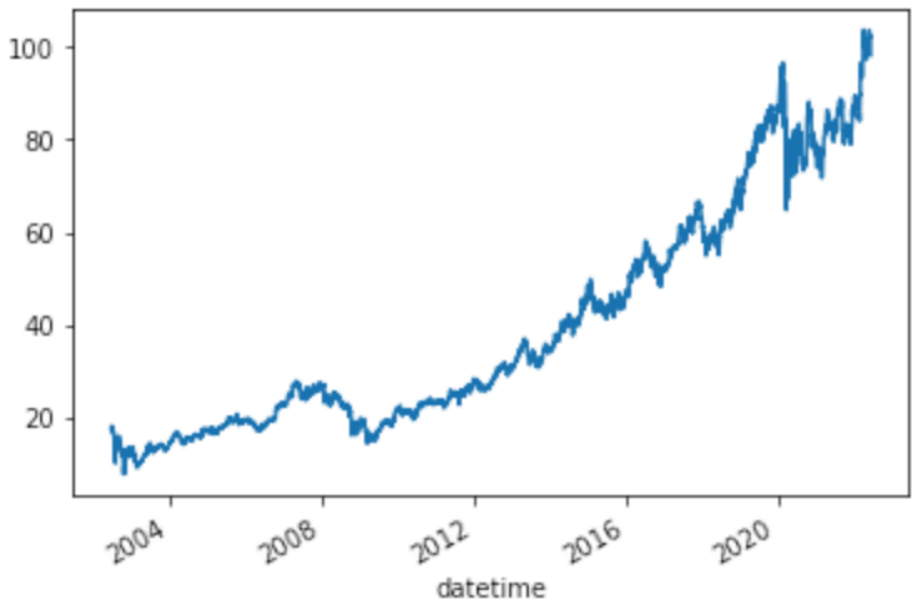
The stock price data from Yahoo Finance has a very high quality, at least as far as the S&P500 stocks are concerned. I could not find any gaps in the data I downloaded, so there are no gaps to be filled. If there were gaps, I would fill them with the average value of the values before and after the gap.

However, not all available tickers can be used, because some companies may be removed from the S&P500 index, either because their capitalization became too small, they were merged into another company, or they just went bankrupt. Also, some tickers were not in the S&P500 during all the last 20 years, because they came in later, either because their capital was not always high enough to be part of the index or because they were founded after 2002. Right after joining an index or right before leaving an index, the markets treat the stocks very differently (because many investors like investment funds are forced to increase/decrease their exposure to the stock). Therefore, I will exclude all tickers from this project, which were not in the S&P500 for the entire last 20 years.

The only tasks related to preprocessing, which have to be performed, are the following:

- Download current list of S&P500 stocks from Wikipedia (https://en.wikipedia.org/wiki/List_of_S%26P_500_companies), by copy&pasting HTML table into Excel-Sheet and saving it under SP500-tickers.xlsx. Remove all columns except the one with the ticker symbol.
- Export the data into CSV file SP500-tickers.csv
- Install the yahoo financials library using "pip install yahoofinancials" on the command line.
- Create a python script price-loader.py which reads the above ticker list and downloads all tickers for the last 20 years, which represents “modern investment behaviour”. Using this tool, create a new folder “prices” with one csv file per ticker, containing the historical price from the last 20 years.
- Validate data for randomly picked ticker AEP: The screenshot from Yahoo Finance from 11.06.2002-9.6.2022 taken from (https://finance.yahoo.com/quote/AEP/history?p=AEP) looks just like the graph drawn in the Jupyter notebook first-validation.ipynb, so the data acquisition method seems to be fine.
- Calculate the daily, 5-day and 20-day returns and save them separately in a new folder “returns”, using the tool preprocess.py.
- Upload the returns to a new bucket in S3 (see screen-shot files).

Out[52]: <AxesSubplot:xlabel='datetime'>



Implementation

The whole application basically has the following parts:

- Data acquisition (price-loader.py)
- Data pre-processing (preprocess.py)
- Model (model.py)
- Training (trainer.py)
- Evaluation (client.py)

The code in loader.py loads all the required time-series and returns them as Pytorch Dataloaders for easy batch processing. The data is divided up into 80% training data, 10% validation data and 10% testing data. Therefore, 3 dataloaders are created (along with the scaler used, so the predictions can be unscaled later).

The code in module.py creates the LSTM model. Only the hidden dimension can be changed, while the rest is fixed: There is only one time-series as input, so the input size is 1, and there is only one output value (i.e. the predicted return), so the output size is also 1. I chose to use only one LSTM layer for simplicity (see chapter "Refinement"), and only one fully-connected layer afterwards. I don't really see a point in using more than one fully-connected layer here, because interpreting the output from the LSTM should already be simple enough (i.e. there should be no higher-order patterns the LSTM itself hasn't figured out itself yet).

The code in trainer.py is responsible for the training process and it returns the best model found (using validation), along with the training/validation losses and the predictions based on the test data.

The main client code is contained in client.py. This code uses the other Python scripts to load the pre-processed data, create the LSTM model, train the model and evaluate the model performance. The implementation details are all documented in the comments in the Python code. I will just mention the most important aspects here.

For the 5-day and the 20-day returns, the returns are broken down into daily returns, and then all 3 predicted returns are averaged (equally weighted), to get the predicted 1d return, which the investment recommendation is based on. If the absolute value of the predicted return is smaller than the "neutral range", the investment recommendation is neutral. Otherwise it is a buy or a sell.

In order to better evaluate the predictive performance, not only the accuracy is evaluated, but also the compound returns based on all the investment recommendations inside the test period is calculated. The aim is, that this compound

return is higher for LSTM predictions than for predictions based on 18 days simple moving average.

The model uses an Adam optimizer, which always performed quite well for me, so I will also use it here. The loss criterion is the mean of squared errors. This is actually a categorization problem, but the prediction of a stock return used for the investment recommendation is rather a regression problem, so MSE is fine.

The application can be told to plot the losses and errors for training (by selecting a ticker using "—ticker"), however, this does not work on EC2, but just locally, even though it is actually a command line tool (it opens a new window for each graph). This is absolutely necessary for manual hyperparameter tuning (learning rate, number of epochs, batch size, LSTM sequence size and hidden dimension. Luckily, my local machine is more performant than the EC2 instances available for my account, so this is fine.

In order to create an overview over a random choice of 50 tickers, the application can be called without setting the "—ticker" parameter. The results will then be printed on the command line in CSV format so they can be easily copy&pasted into an Excel sheet for further analysis.

In order to allow for further optimization, the application does not only track the correct investment recommendations, but there is also a detailed analysis of the incorrect ones. For example, it shows how often which kind of errors were made, so it can be analysed if the model has some systematic bias which could be dealt with (example: if too many "neutral" recommendations were made, which should be actually buy or sell recommendations, the threshold for the "neutral range" should be increased).

Refinement

Most of the refinements have already been part of the project proposal, because I had already developed a part of the code when I wrote it. These were basically the major refinements compared to my original idea:

- Instead of just using the daily returns, which seemed like the model was not learning at all, I indroduced also the 5d and 20d returns, which made the model really learn something, as can be seen in the results. Before that, I was desperately trying to find the bug in my logic, because I could not see any progress whatsoever.
- Originally, I just wanted to check the accuracy, but then I realized, that this is not a good metric on its own, because it could be that one model has higher accuracy than another but is actually mostly accurate only on predicting small returns. The other model could have a lower accuracy, but better predict the

large returns and therefore make more money, which is obviously the ultimate goal.

- In my original design, I had not introduced batching for simplicity, because I had planned to just use hundreds of iterations to get the same effect. However, the increase in code complexity was totally worth it, because the learning is actually faster and the code is also more interesting and closer to real-life applications.

RESULTS

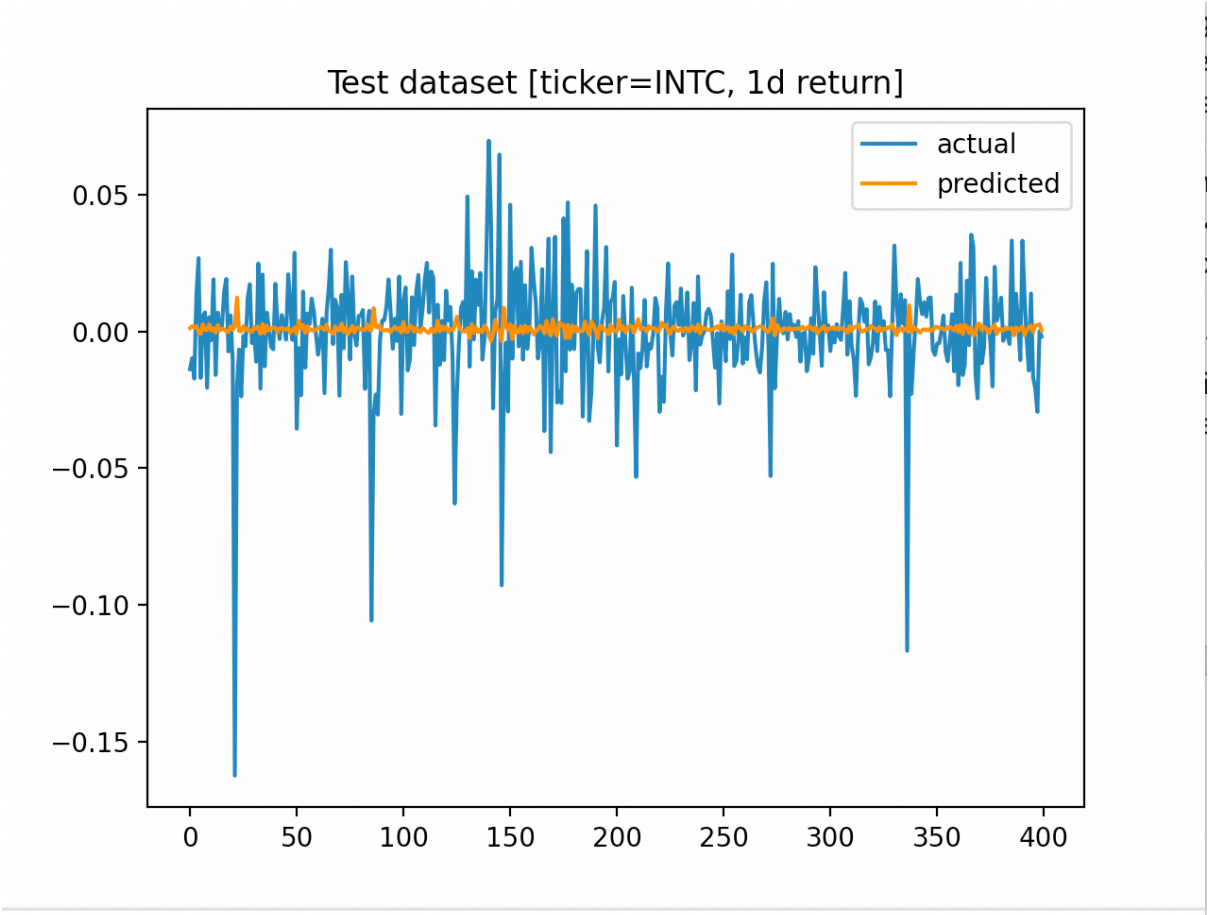
Model Evaluation and Validation

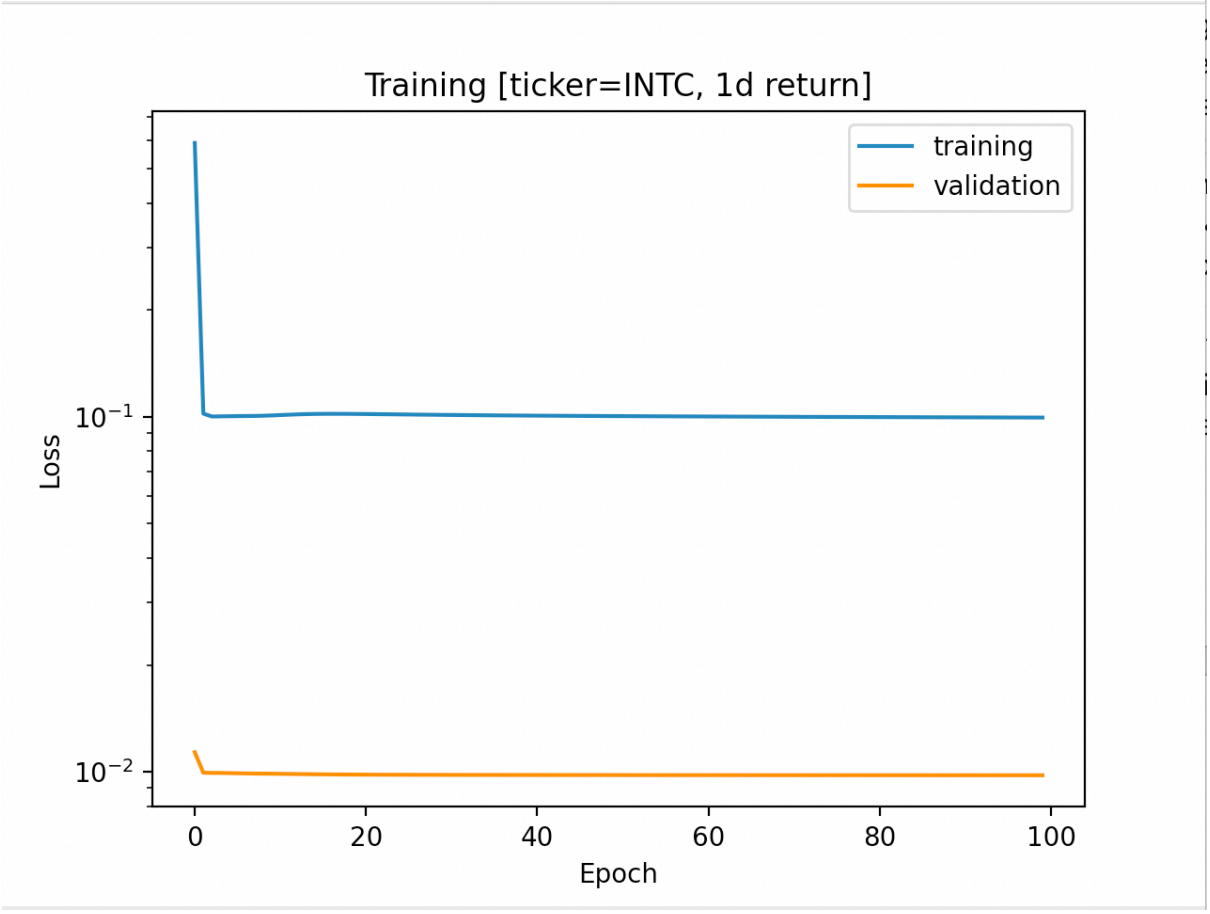
After trying numerous parameters, the following choice turned out to be optimal:

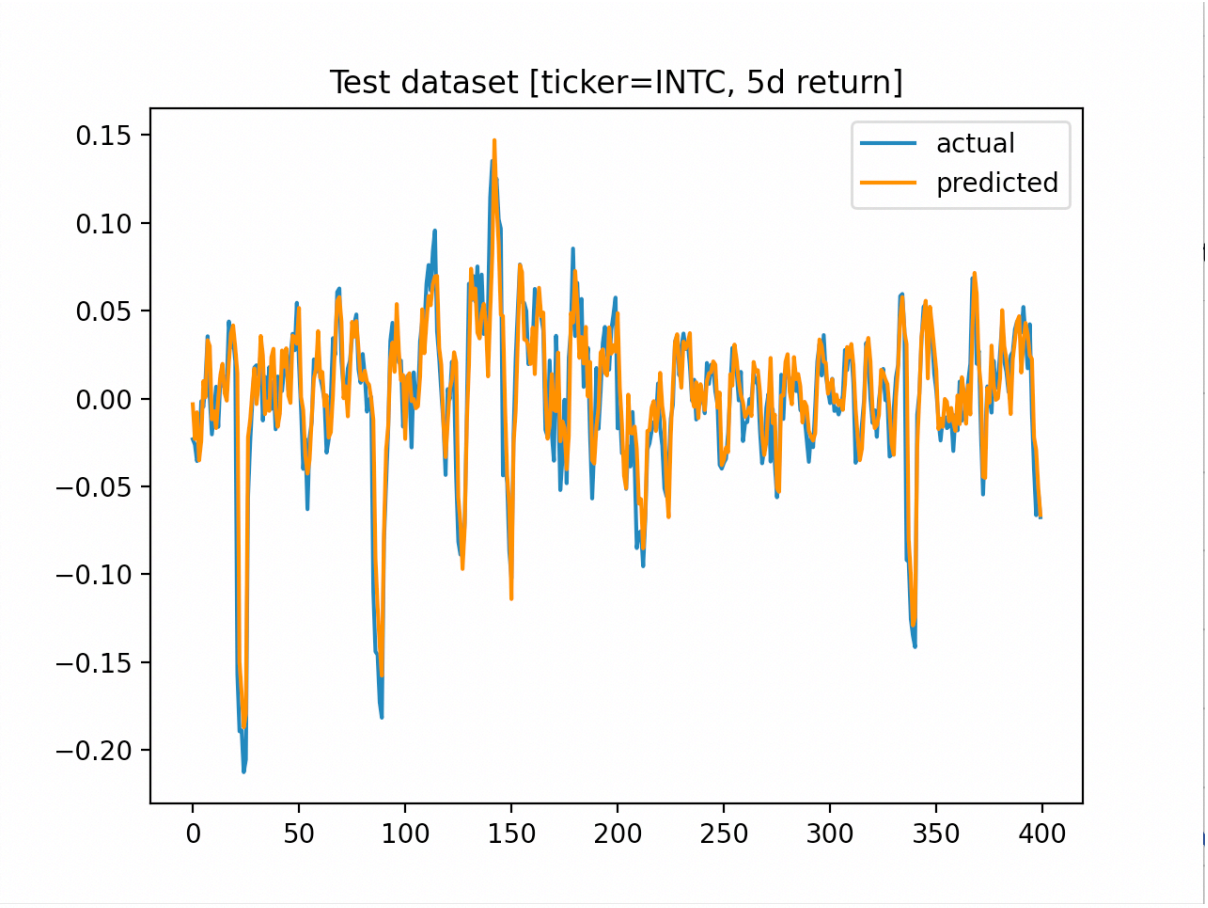
- Learning rate = 0.01 (smaller rates are too slow and there was also no overshooting so this relatively large learning rate is not too large)
- LSTM sequence length = 50 (larger lengths were only slower, but did not predict better)
- LSTM hidden dimension = 30 (also here, larger lengths were only slower, but did not predict better)
- Training batch size = 100 (no significant impact on predictive power detected, however, I have also tried it without batching and this took several hundred epochs to get the same level of accuracy)
- Number of training epochs = 100 (after that, there was no more improvement)
- Neutral range = 0.0005 (minimum absolut value of a prediction neccassary to be able to make a buy or sell recommendation. Otherwise stay neutral.). A larger rate causes the logic too risk too little and a smaller rate causes the logic to get the sign of the position wrong too often)

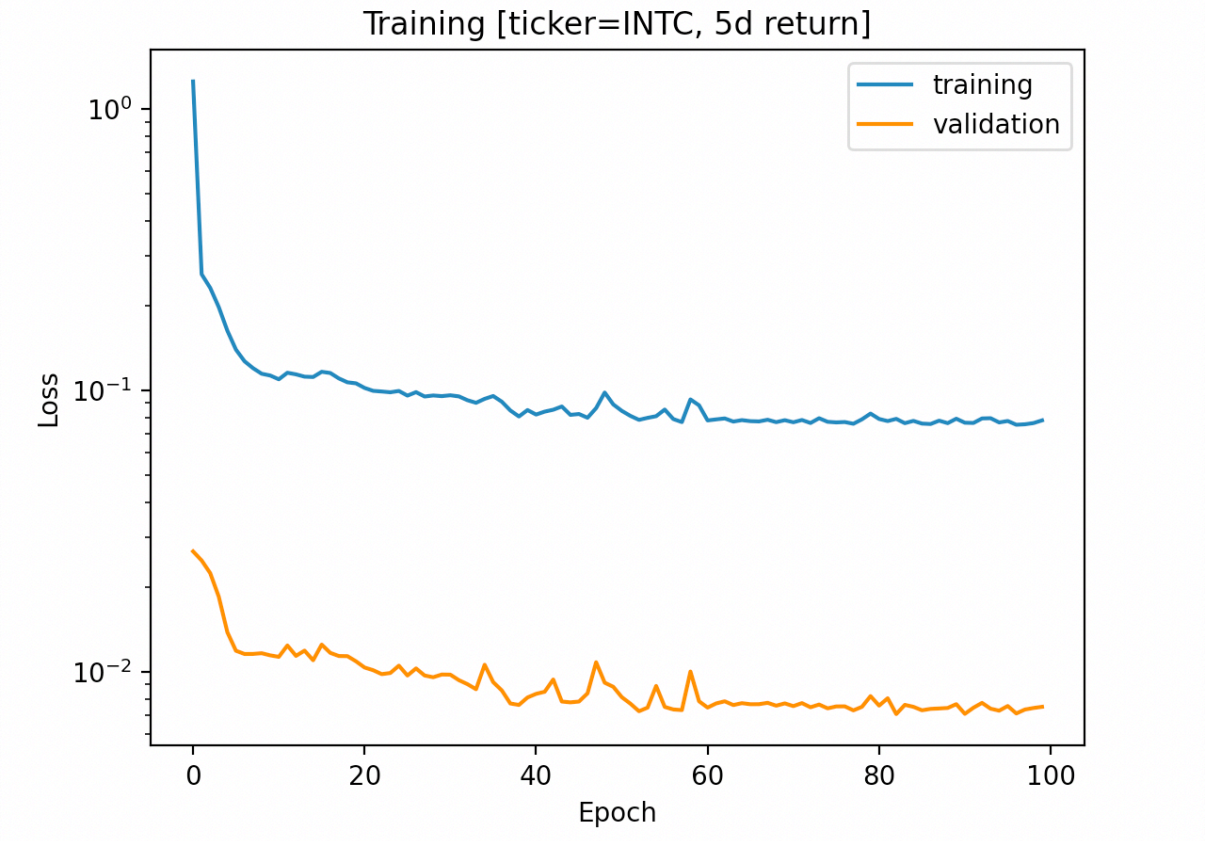
The results and training progress is displayed for the growth stock INTC (Intel Corp.) and value stock WMT (Walmart Inc.), for 1d, 5d and 20d returns:

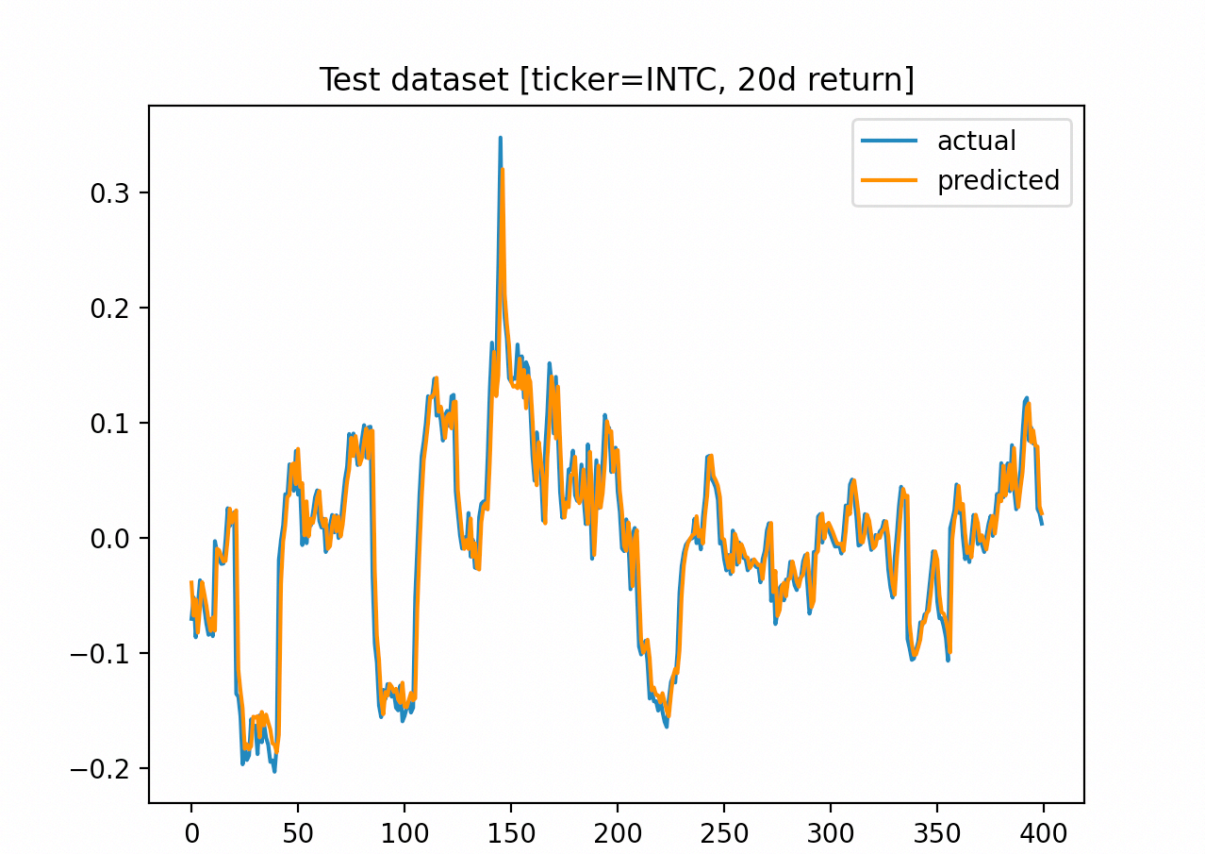
```
EVALUATION OF LSTM PREDICTIONS:
Number of predictions: 400
Number of correct predictions: 177
Recommendation accuracy: 0.4425
Number of BUY recommendations: 210
Number of NEUTRAL recommendations: 61
Number of SELL recommendations: 129
Number of correct BUY recommendations: 109
Number of correct SELL recommendations: 66
Number of correct NEUTRAL recommendations: 2
Number of incorrect BUY recommendations (where NEUTRAL would be correct): 3
Number of incorrect BUY recommendations (where SELL would be correct): 98
Number of incorrect SELL recommendations (where NEUTRAL would be correct): 8
Number of incorrect SELL recommendations (where BUY would be correct): 55
Number of incorrect NEUTRAL recommendations (where BUY would be correct): 34
Number of incorrect NEUTRAL recommendations (where SELL would be correct): 25
profit/loss=-18.77724582189282%

EVALUATION OF 18 DAY SIMPLE MOVING AVERAGE PREDICTIONS (BENCHMARK):
Number of predictions: 382
Number of correct predictions: 178
Recommendation accuracy: 0.46596858638743455
Number of BUY recommendations: 189
Number of NEUTRAL recommendations: 55
Number of SELL recommendations: 138
Number of correct BUY recommendations: 102
Number of correct SELL recommendations: 73
Number of correct NEUTRAL recommendations: 3
Number of incorrect BUY recommendations (where NEUTRAL would be correct): 6
Number of incorrect BUY recommendations (where SELL would be correct): 81
Number of incorrect SELL recommendations (where NEUTRAL would be correct): 3
Number of incorrect SELL recommendations (where BUY would be correct): 62
Number of incorrect NEUTRAL recommendations (where BUY would be correct): 27
Number of incorrect NEUTRAL recommendations (where SELL would be correct): 25
profit/loss=278.5092352763576%

CONCLUSION:
SMA18 beats LSTM for ticker INTC.
```

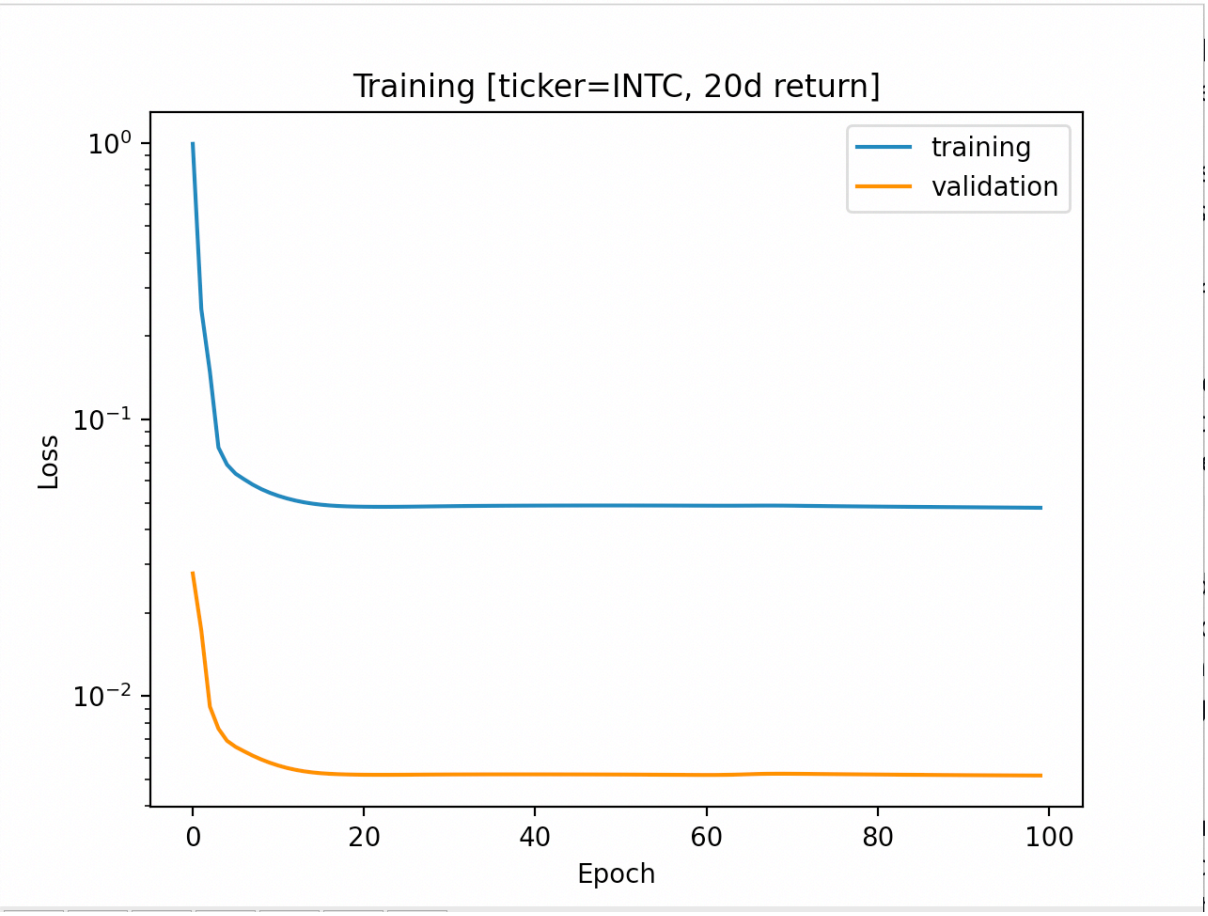

Test dataset [ticker=INTC, 1d return]
actual
predicted
0.05
0.00
−0.05
−0.10
−0.15
0
50
100
150
200
250
300
350
400


Training [ticker=INTC, 1d return]
training
validation
Loss
10^{-1}
10^{-2}
0
20
40
60
80
100
Epoch

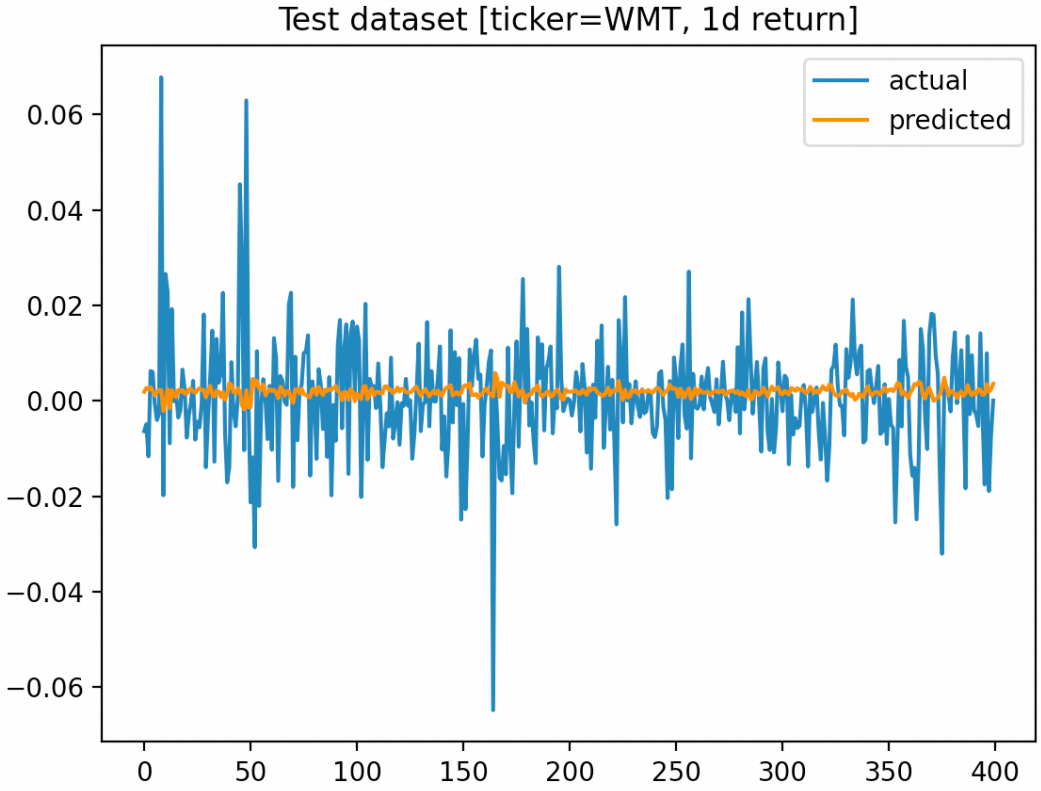

Test dataset [ticker=INTC, 5d return]
actual
predicted
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0
50
100
150
200
250
300
350
400

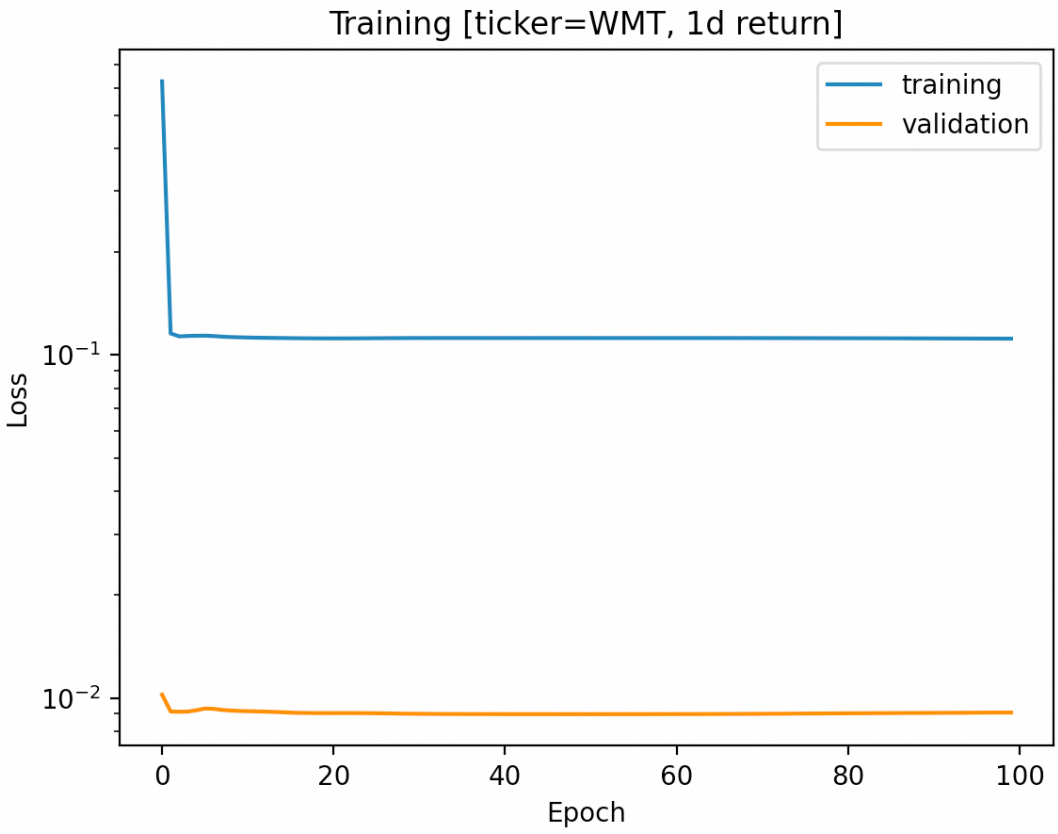

Training [ticker=INTC, 5d return]
training
validation
Loss
10^{0}
10^{-1}
10^{-2}
0
20
40
60
80
100
Epoch

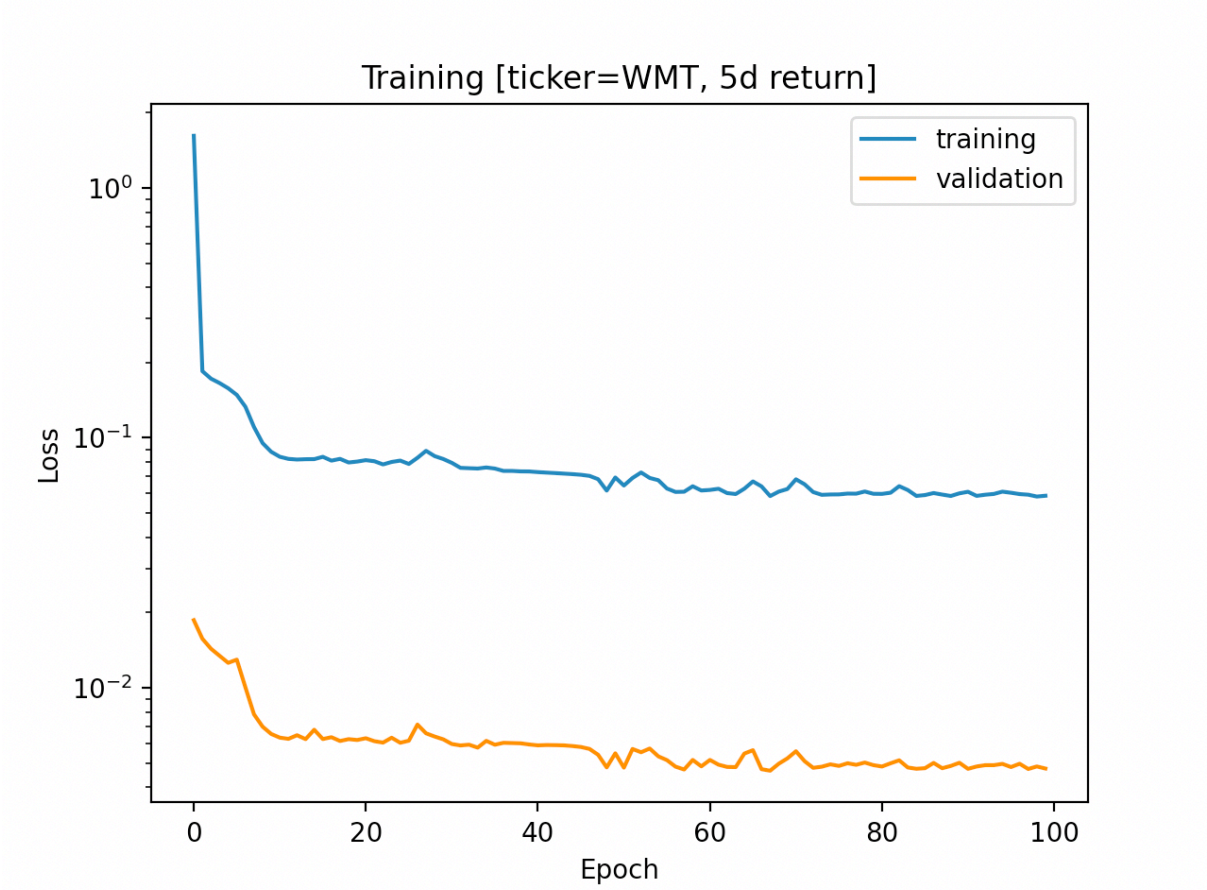

Test dataset [ticker=INTC, 20d return]
actual
predicted
0.3
0.2
0.1
0.0
−0.1
−0.2
0
50
100
150
200
250
300
350
400

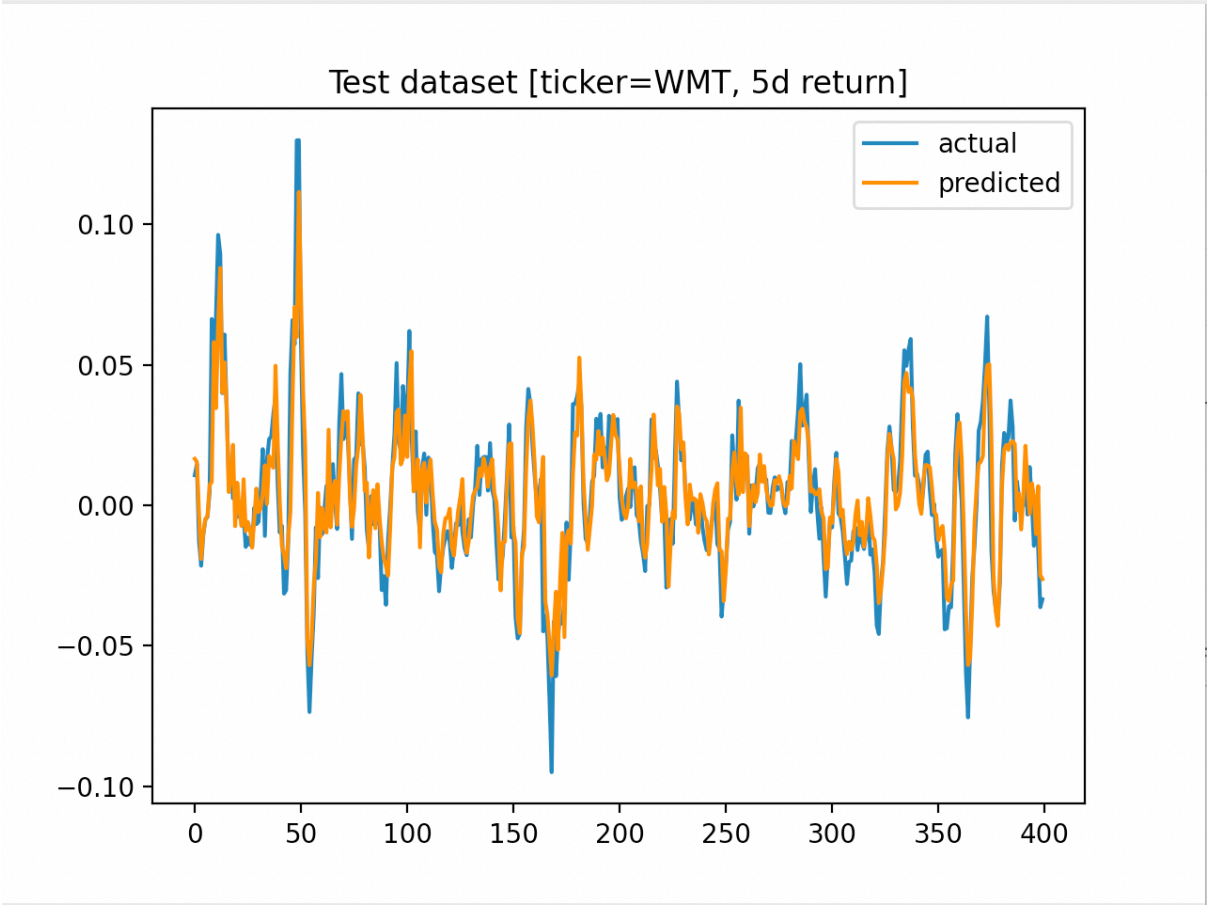

Training [ticker=INTC, 20d return]
training
validation
10^{0}
10^{-1}
10^{-2}
Loss
0
20
40
60
80
100
Epoch

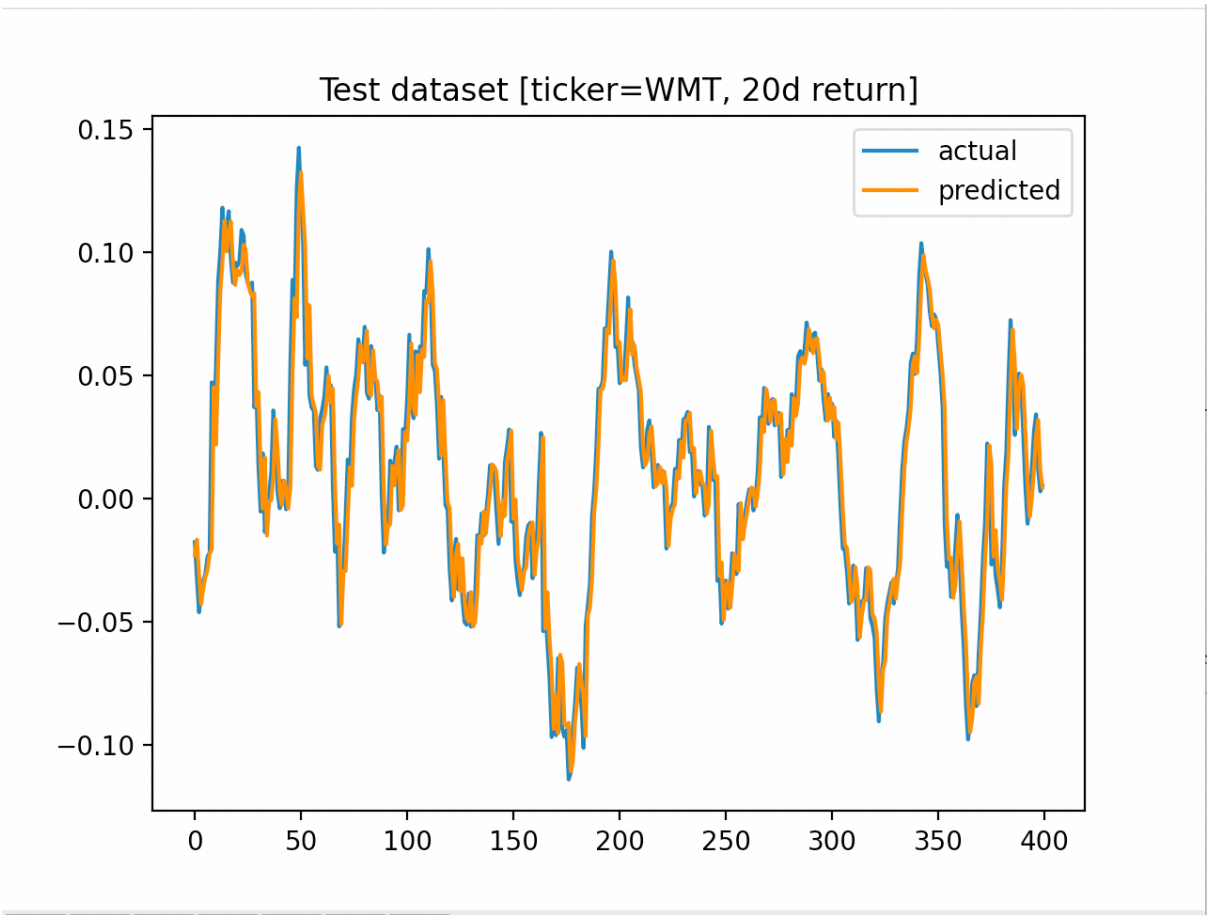
```
EVALUATION OF LSTM PREDICTIONS:
Number of predictions: 400
Number of correct predictions: 156
Recommendation accuracy: 0.39
Number of BUY recommendations: 260
Number of NEUTRAL recommendations: 68
Number of SELL recommendations: 72
Number of correct BUY recommendations: 119
Number of correct SELL recommendations: 33
Number of correct NEUTRAL recommendations: 4
Number of incorrect BUY recommendations (where NEUTRAL would be correct): 18
Number of incorrect BUY recommendations (where SELL would be correct): 123
Number of incorrect SELL recommendations (where NEUTRAL would be correct): 4
Number of incorrect SELL recommendations (where BUY would be correct): 35
Number of incorrect NEUTRAL recommendations (where BUY would be correct): 37
Number of incorrect NEUTRAL recommendations (where SELL would be correct): 27
profit/loss=10.83806914157217%

EVALUATION OF 18 DAY SIMPLE MOVING AVERAGE PREDICTIONS (BENCHMARK):
Number of predictions: 382
Number of correct predictions: 176
Recommendation accuracy: 0.4607329842931937
Number of BUY recommendations: 198
Number of NEUTRAL recommendations: 60
Number of SELL recommendations: 124
Number of correct BUY recommendations: 106
Number of correct SELL recommendations: 66
Number of correct NEUTRAL recommendations: 4
Number of incorrect BUY recommendations (where NEUTRAL would be correct): 10
Number of incorrect BUY recommendations (where SELL would be correct): 82
Number of incorrect SELL recommendations (where NEUTRAL would be correct): 9
Number of incorrect SELL recommendations (where BUY would be correct): 49
Number of incorrect NEUTRAL recommendations (where BUY would be correct): 30
Number of incorrect NEUTRAL recommendations (where SELL would be correct): 26
profit/loss=127.88585443371888%

CONCLUSION:
SMA18 beats LSTM for ticker WMT.
```

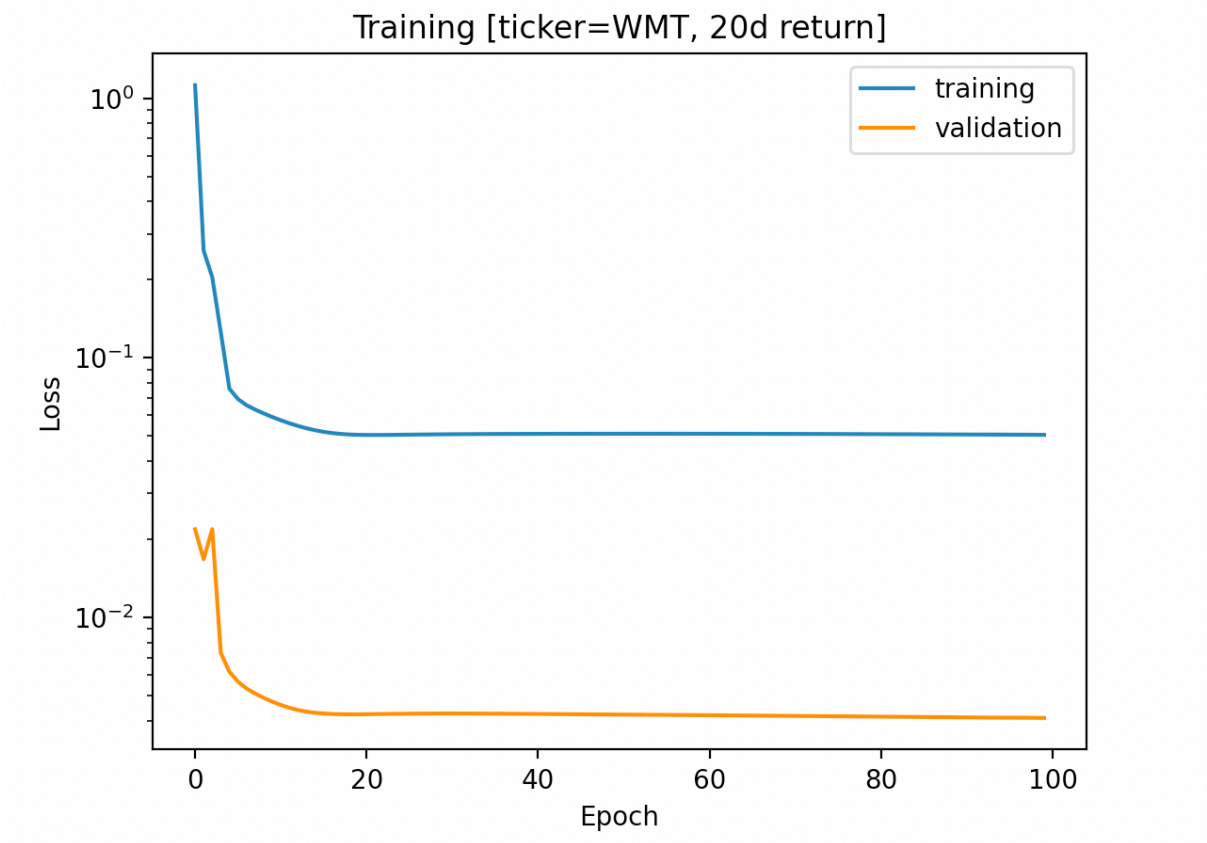

Test dataset [ticker=WMT, 1d return]
actual
predicted
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
50
100
150
200
250
300
350
400


Training [ticker=WMT, 1d return]
training
validation
Loss
10^{-1}
10^{-2}
0
20
40
60
80
100
Epoch


Training [ticker=WMT, 5d return]
training
validation
Loss
10^0
10^{-1}
10^{-2}
0
20
40
60
80
100
Epoch


Test dataset [ticker=WMT, 5d return]
actual
predicted
0.10
0.05
0.00
−0.05
−0.10
0
50
100
150
200
250
300
350
400


Test dataset [ticker=WMT, 20d return]
actual
predicted
0.15
0.10
0.05
0.00
−0.05
−0.10
0
50
100
150
200
250
300
350
400


Training [ticker=WMT, 20d return]
training
validation
10^{0}
10^{-1}
10^{-2}
Loss
0
20
40
60
80
100
Epoch

This is a list of the accuracies and profits/losses for 50 randomly picked stocks, using the parameters recommended at the beginning of the chapter:

ticker	accuracy_lstm	accuracy_sma18	profit_loss_lstm	profit_loss_sma18
COP	45%	52%	50%	546%
ALK	45%	54%	-15%	737%
CCI	41%	47%	3%	80%
JPM	40%	46%	-7%	174%
CCL	48%	54%	-41%	1270%
AMAT	45%	50%	-1%	300%
HST	40%	48%	-70%	391%
MHK	45%	56%	6%	801%
TXN	36%	46%	-54%	115%
FCX	46%	56%	138%	1405%
UDR	40%	50%	-15%	139%
EMN	43%	52%	3%	286%
DLTR	44%	49%	28%	353%
AMD	45%	54%	43%	684%
MU	47%	49%	24%	533%
FMC	44%	46%	-30%	140%
GIS	34%	49%	-31%	147%
IEX	43%	48%	-12%	128%
DRE	40%	50%	-8%	113%
GE	42%	51%	-39%	511%
BEN	42%	52%	-52%	259%
ZION	38%	48%	-59%	676%
WAB	35%	45%	-29%	169%
SYY	40%	52%	-24%	249%
EMR	41%	50%	-14%	191%
LIN	43%	50%	9%	118%
GE	44%	51%	-12%	511%
CSX	42%	52%	-26%	173%
MSFT	44%	48%	-9%	173%
CCL	48%	54%	-31%	1270%
MCD	38%	46%	-5%	97%
NEM	42%	53%	-33%	233%
EMR	39%	50%	-28%	191%
AON	43%	48%	15%	163%
TXT	42%	54%	-11%	315%
FE	43%	48%	-5%	155%
UDR	35%	50%	-41%	139%

SPG	42%	56%	-49%	1032%
AVB	44%	48%	17%	100%
PFE	45%	52%	102%	288%
FDS	45%	50%	1%	158%
COO	46%	53%	-1%	206%
BMY	41%	51%	-15%	132%
T	37%	43%	-4%	94%
FMC	44%	46%	-31%	140%
VTR	37%	48%	-29%	205%
ATO	38%	53%	-20%	191%
LIN	45%	50%	13%	118%
WTW	38%	47%	-4%	216%
CTRA	42%	47%	-24%	264%

Justification

Despite the predictions for 5d and 20d looking really good, it is important to realize, that unlike the predictions for the 1d returns, the predictions for the 5d and 20d returns contain some information which is already known. 80% of the information for the 5d return prediction comes from the 4 days before (i.e. the past), and 95% of the information for the 20d return prediction comes from the 19 days before. The only thing the models really predict, are the return components of the last day.

As far as the graphs for the training losses are concerned, it has to be noted, that the training period is 8 times longer than the validation period, so the losses will be also be bigger by a factor of 8. As only the development over time is relevant, this is not a problem.

CONCLUSION

Reflection

Apparently, the LSTMs can not successfully predict stock returns for S&P 500 stocks, in the way it has been used in this project.

I had been warned about this by several people, saying that the stocks in the S&P500 are so liquid and transparent, that all possible information is immediately priced in. Also there are countless quant funds (algorithmic trading, etc.) which try to make exactly this kind of predictions, and as soon as there is any slight opportunity for making a profit, it is "scooped up" in a matter of milliseconds, returning the prices to their fair value immediately.

All that a model could reasonably be expected to pick up on would be momentum trading, which is why the SMA18 works quite well. However, LSTMs are meant to look for patterns, and are not good for analysing trends.

Improvement

It could be tried to increase the number of LSTM layers (stacked LSTMs), to see if the model can pick up several kinds of patterns at the same time, which, when combined, could yield a better accuracy.

Given the fact, that stocks generally move faster when they go down than when they go up (crashes are generally much faster than recoveries), different thresholds for the neutral behaviour could be introduced on the long and on the short side, with the short side having a larger threshold.

Also interesting would be to incorporate other types of models in parallel (like sentiment models). Also some purchase manager indices or income statement data could be incorporated, to get "hints" from as many different sources as possible. Also some macroeconomic time series like the "Baltic Dry Index" would be very interesting to give the model a macroeconomic influence, which would certainly help with long-term tendencies.

REFERENCES

- Historical stock prices for Intel Corp. (ticker INTC) from Yahoo Finance:
  https://finance.yahoo.com/quote/INTC/history?p=INTC
- INTC ticker from Comdirect Bank:
  https://www.comdirect.de/inf/aktien/US4581401001
- LSTM model basics:
  Chapter 15.2.7.2 “Long short term memory (LSTM)” (p. 507)
  (Murphy, Kevin P.. Probabilistic Machine Learning (Adaptive Computation and Machine Learning series) MIT Press.)
- LSTM model details:
  https://colah.github.io/posts/2015-08-Understanding-LSTMs
- Recurrent layers in PyTorch:
  https://pytorch.org/docs/stable/nn.html#recurrent-layers
- List of S&P 500 tickers with industry and addition date:
  https://en.wikipedia.org/wiki/List_of_S%26P_500_companies
- Implementation and use of LSTM models:
  “Recurrent Neural Networks” from “Deep Learning Nanodegree” program, Udacity