IEOR E4720 Deep Learning Final Report

EFFICIENCY OF ATTENTION MECHANISM IN LSTM BASED STOCK PRICE MOVEMENT PREDICTION

January 2, 2019

Ethan Finkiel - ef2553
Yuan Ma - ym2635
Chuqiao Wang - cw3038
Deepti Kalikivaya - dk2985

1 Introduction

Predicting stock price movement has always been an attractive and challenging topic in the financial world. Recently, quite a few deep learning techniques including: LSTM, GRU and CNN-LSTM have been used to study the mechanism of the stock market. The attention mechanism has recently been acknowledged as an effective method in sequential learning applications. Based on exploring and furthering the paper *Exploring Attention Mechanism in LSTM based Hong Kong Stock Price Movement Prediction* (to appear in Quantitative Finance), this project is trying to evaluate the efficiency of Deep Learning on sequence classification in the context of stock prices.

In our attempt to replicate the paper, we used Hong Kong stock data along with various technical factors which were computed beforehand as input variables. We then tuned various parameters. For our extension, we did parameters tuning and randomly chose 100 stocks among the constituents of the S&P 500. For further information, we also made some investment analysis.

The remainder of this paper is organized as follows. In part 2, we introduce the methodology, replication strategy and then present the results of our replication part. In part 3, we present the extension of the paper. First, we evaluate different parameters settings tuned on the Hong Kong data. Afterwards, we apply the optimal Hong Kong settings to the S&P stock data and evaluate the results. Finally, we provide the analysis of our research results.

2 Methodology and Implementation

2.1 Attention-LSTM and Feature Selection

The LSTM network has been proven to be an effective Deep Learning model for sequential tasks. A two layered LSTM network programmed using Keras is used as in the original paper. After each layer, a dropout is introduced to help prevent over-fitting. After the two LSTM layers, a dense layer is introduced followed by Batch Normalization. Then a softmax layer is added to provide one of two outputs: up or down. The Attention-LSTM model introduces an Attention layer before the first LSTM layer. For detailed information on each layer in the model, please refer to the appendix.

The model utilizes daily stock data including: open, close, high, low, and volume. From there, we used TALib and calculated moving averages and their percent changes of each factor, the moving average convergence/divergence, relative strength index and its percentage change, and Bollinger Bands. Further information on the technical indicators is provided in the appendix.

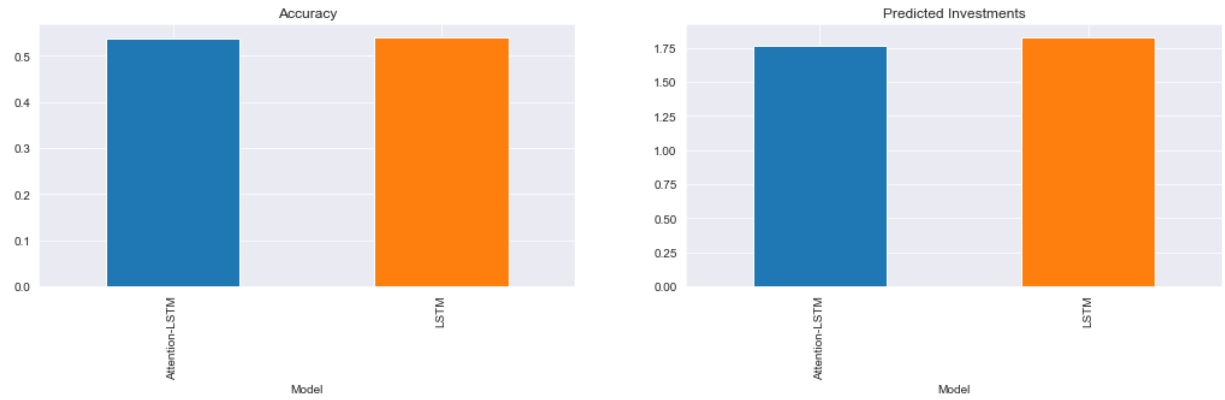
2.2 Replicating Result from HK Market

We used all available data upto and including October 12th, 2018 for stocks with tickers between 00801.HK and 00899.HK . The original paper did not specify the end date for the data used. We built the specified model in Keras and then recreated it in TensorFlow. Batch size, learning rate and dropout are set to 512, 0.001, and 0.2 respectively. We split the data into test and training sets. The test set consisted of the latest 700 days and the training set contained all other days. The training set used 30% of the data for validation. We required each stock to have 1060 days of data. This gave a minimum of one trading year or 252 days for training, not including days used for testing.

Below is a table outlining the performance of the models. Our results do not align with the results of the paper. The original paper had an AttLSTM win rate of 82.05% compared to our 50% win rate. We believe this is due to a number of factors, including the methodology we used to calculate the accuracy rates of each model. The results of our model are shown in table 1.

In addition, we deemed accuracy of prediction to be an inaccurate measure of success. One does not measure the success of a hedge fund in how many days they made the correct investments, but rather in how much they returned. As such, we also calculated a predicted investment for each stock using a simplistic investment model. Each day, we used the prediction to either long or short that days rate of change from open to close and reinvested all prior profits. We compared this to the total performance of that stock over the same 700 day period. The data of this analysis is shown in table 2.

Figure 1



(a) Prediction accuracy

(b) Predicted Investments

After analyzing the data, one can see a number of large outliers. The model is predicting days with larger returns correctly. With the fact that the investment is compounding daily, this leads to larger returns and losses. The LSTM and Attention-LSTM models outperform the market nearly 70% of the time.

3 Parameter Tuning

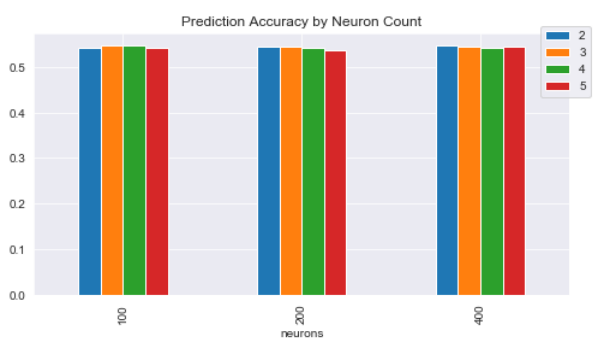
3.1 Neuron Count Analysis

One main problem to adding more neurons is that this does not necessarily improve the prediction results of the model, rather it leads to over-fitting. As one can see in figure 4, the accuracy is highest in models with one hundred neurons on each layer. One can note a downward trend in the 200 and 400 neuron count layers, suggesting that the model does quickly over-fit.

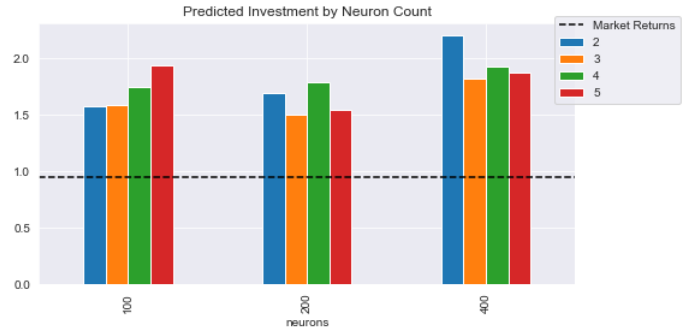
As mentioned before, we believe investment return is a better performance measurement than simple prediction accuracy. After analyzing the data, there were several outliers observed in the data. Hence, we excluded any stock whose returns were greater than 1000% over 700 days. This was to remove statistical anomalies where over-performance was merely due to data issue. This leads to

some very interesting results. As seen in figure 5, all of the models outperformed the market quite well.

Figure 2



(a) Prediction accuracy by neuron count

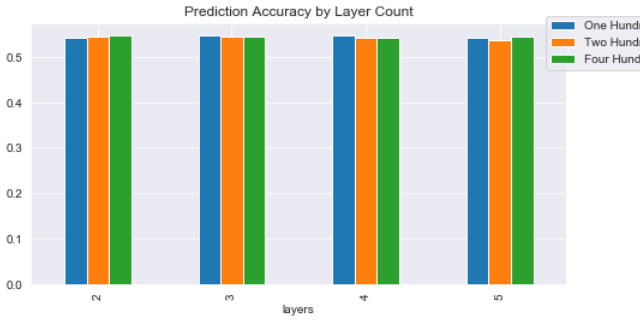


(b) Prediction investment performance by neuron count (outliers removed)

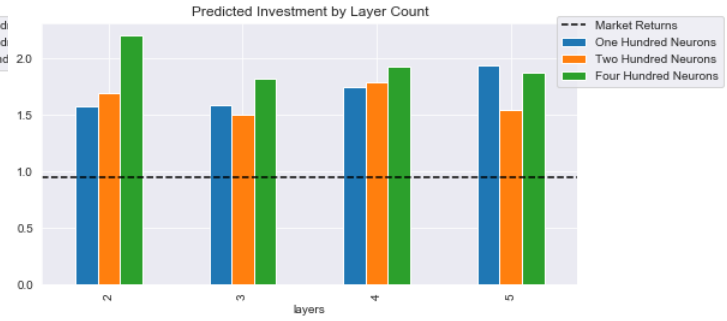
3.2 Layer Count Analysis

Similar to neuron count, additional layers led to over-fitting. As can be seen from figure 6, for three, four and five layers,there is no clear improvement in performance by adding neurons. However, with two layers, there is definitely some improvement to the accuracy. In contrary to the prediction accuracy, adding additional layers does lead to better investment performance as seen in figure 7.

Figure 3



(a) Prediction accuracy by layer count



(b) Prediction investment performance by layer count

3.3 Analysis of Tuning

It is crucial to define a model that captures as much prediction power as possible without over-fitting. Our empirical analysis shows that using two layers with 400 neurons with the Attention-LSTM model is going to give the optimal performance on both accuracy and return on investment.

4 Extension: Implementation on S&P 500

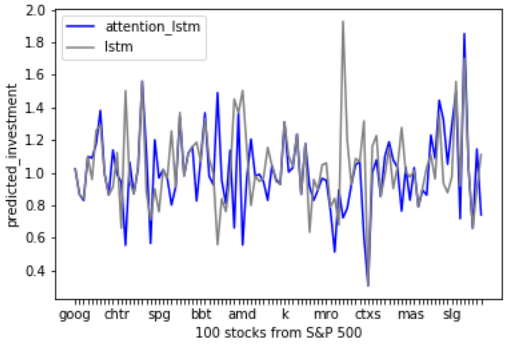
4.1 Experiment setup

The S&P 500 data is downloaded from Kaggle dataset, 100 of the stocks are randomly picked to run the model. As the time duration for which S&P stock data is available is longer than that of the Hong Kong stocks, we consider increasing the sequence length to 20. The train/valid/test set is changed to 70%/20%/10%. The optimal number of neurons each layer is tested to be 200. Other parameters are the same the model implemented to Hong Kong data.

4.2 Result

The experimental results are listed in Table 1 - 'accuracy comparison'. The AttLSTM outperforms LSTM 75.51% of the time. But both models' performance are just above a fair guess. For the investment performance, the result is shown in figure 10. As one can see, AttLSTM gives a more narrow range of returns for these 98 stocks, while LSTM may perform extremely in both directions.

Figure 4: Investment Result



Name	LSTM	AttLSTM	Name	LSTM	AttLSTM
GOOG	**0.4118**	**0.4118**	XEL	**0.3837**	**0.3837**
JNJ	**0.5583**	**0.5583**	K	**0.5091**	**0.5091**
PEP	0.4732	**0.4927**	ED	0.5000	**0.5042**
ABBV	**0.2609**	**0.2609**	SYF	0.4000	**0.4667**
IBM	0.4750	**0.5036**	KR	**0.4829**	**0.4829**
LMT	**0.5854**	0.5756	WY	0.4683	**0.4878**
SLB	0.4444	**0.4722**	ROK	**0.5611**	**0.5611**
GS	**0.4783**	**0.4783**	NTAP	**0.4771**	**0.4771**
CVS	**0.5273**	**0.5273**	KEY	0.4600	**0.6000**
EOG	**0.5071**	0.4929	HRL	**0.5079**	**0.5079**
CHTR	**0.5789**	**0.5789**	CTAS	0.4710	**0.5870**
TJX	**0.5638**	0.5101	MRO	0.5333	**0.5375**
AGN	**0.5447**	0.4878	DTE	0.4875	**0.4958**
PNC	0.3973	**0.6096**	AZO	**0.5414**	0.5188
ATVI	**0.2698**	**0.2698**	FITB	**0.4493**	**0.4493**
MDLZ	**0.5273**	**0.5273**	CA	**0.5120**	0.4398
INTU	**0.6260**	**0.6260**	FE	**0.5100**	0.4800
RTN	0.4944	**0.5556**	MKC	**0.6304**	**0.6304**

CL	0.5415	**0.5512**	FAST	**0.5217**	**0.5217**
PSX	**0.7037**	0.4815	ESS	**0.5079**	**0.5079**
SPG	0.4790	**0.5042**	DHI	**0.5079**	**0.5079**
BSX	**0.4488**	**0.4488**	CTXS	**0.4771**	**0.4771**
VLO	**0.4603**	**0.4603**	TPR	**0.5556**	0.4286
NSC	**0.5000**	0.4607	FTI	0.4762	**0.4921**
HCA	**0.4375**	**0.4375**	CMS	**0.5183**	**0.5183**
TGT	0.5318	**0.5434**	STX	**0.6032**	0.3810
ZTS	**0.6957**	**0.6957**	CF	**0.5902**	0.5082
SO	**0.5444**	**0.5444**	MOS	0.3492	**0.3651**
STZ	**0.5238**	**0.5238**	EXPD	**0.4928**	**0.4928**
WM	0.3810	**0.6190**	CINF	**0.5217**	0.4855
BBT	0.4928	**0.5000**	ALB	**0.5873**	0.5238
APD	0.4913	**0.5087**	MAS	0.4767	**0.5291**
ROST	0.4459	**0.4522**	CBOE	**0.7778**	0.7500
AEP	0.5167	**0.5208**	M	**0.4286**	**0.4286**
ADSK	**0.5432**	0.4444	VAR	0.5397	**0.5556**
STT	0.4713	**0.4777**	TMK	0.3960	**0.5906**
CXO	**0.5294**	0.5098	IRM	**0.6349**	0.5873
PXD	0.4286	**0.5079**	QRVO	**0.5385**	0.2308
YUM	**0.6500**	0.3400	UAA	0.6271	**0.6441**
ADM	**0.5202**	**0.5202**	WU	**0.6727**	0.4909
AMD	**0.5115**	0.4655	FLIR	0.5238	**0.6190**
RCL	**0.4762**	**0.4762**	SLG	0.4762	**0.5556**
DFS	**0.5098**	0.4902	XRAY	**0.5606**	**0.5606**
ZBH	**0.6049**	**0.6049**	ALK	0.4762	**0.5397**
GIS	0.5058	**0.6047**	XRX	**0.5366**	0.5317
CMI	**0.5305**	**0.5305**	EVHC	**0.5500**	**0.5500**
PH	**0.5000**	**0.5000**	FL	**0.5079**	**0.5079**
BF.B	**0.5429**	**0.5429**	HRB	0.4452	**0.5742**
EQR	0.6033	**0.6198**	SRCL	0.4286	**0.5238**
AttLSTM wins	**74**	**Total Num**	**98**	**AttLSTM wins**	**75.51%**

5 Conclusion

We evaluated the performance of Attention LSTM vs. LSTM on Hong Kong stock market using TensorFlow and Keras. Our results do not align with the results of the original paper. We believe there are several factors contributing to the discrepancy of the results. One of which is accuracy calculation. Investment performance seems to be a better measure than the number of times the prediction was right. Both the models outperform market in terms of investment performance. We then tune the parameters and obtain the optimal set of parameters avoiding overfitting: 2 layers with 400 neurons in each layer. As extension to our study, we implement the model on 100 randomly selected stocks from S&P 500. Attention LSTM outperformed LSTM 75.5% of the times, which is better than that from Hong Kong market.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[4] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Appendices

A Results from Replication on Hong Kong Stock Data

A.1 Prediction Accuracy

Name	LSTM	AttLSTM	Name	LSTM	AttLSTM
00800.HK	**0.5729**	0.4014	00801.HK	0.6257	**0.6486**
00802.HK	**0.5671**	0.5343	00803.HK	**0.5714**	0.5557
00805.HK	**0.4857**	0.4614	00806.HK	**0.5**	0.4786
00808.HK	**0.6214**	0.6086	00809.HK	0.4843	**0.5486**
00810.HK	**0.5557**	0.5229	00811.HK	**0.5157**	0.5
00812.HK	**0.5729**	0.5229	00813.HK	0.4643	**0.5014**
00814.HK	0.5529	**0.5814**	00815.HK	**0.5471**	0.5329
00816.HK	0.4829	**0.5257**	00817.HK	0.5257	**0.5271**
00818.HK	**0.5457**	0.4829	00819.HK	0.4929	**0.5129**
00820.HK	**0.6029**	0.5743	00821.HK	0.5257	**0.5529**
00822.HK	0.5643	**0.5714**	00823.HK	0.4486	**0.49**
00825.HK	**0.57**	0.5529	00826.HK	**0.5571**	0.5371
00827.HK	**0.5214**	0.5014	00829.HK	0.5643	**0.6043**
00830.HK	**0.5714**	0.5514	00832.HK	0.54	**0.5514**
00833.HK	0.5571	**0.5629**	00836.HK	0.48	**0.5157**
00837.HK	**0.5714**	0.56	00838.HK	0.53	**0.5614**
00840.HK	0.6286	**0.6443**	00841.HK	0.55	**0.5714**
00842.HK	**0.5786**	0.5543	00844.HK	0.5657	**0.5771**
00845.HK	0.6043	**0.6186**	00848.HK	**0.6**	0.5757
00850.HK	0.5186	**0.52**	00851.HK	**0.6243**	0.6114
00852.HK	**0.5771**	**0.5771**	00853.HK	**0.5214**	0.4957
00855.HK	0.4686	**0.5086**	00856.HK	0.5243	**0.55**
00857.HK	**0.4986**	0.4857	00858.HK	0.5686	**0.5943**
00859.HK	**0.5329**	0.5314	00860.HK	**0.5457**	0.5029
00861.HK	0.51	**0.5186**	00862.HK	0.54	**0.5429**
00864.HK	**0.5914**	0.5571	00866.HK	**0.5357**	0.44
00867.HK	**0.5371**	0.5314	00868.HK	**0.5114**	0.4929
00869.HK	**0.5529**	0.53	00871.HK	0.5029	**0.5157**
00872.HK	**0.5857**	0.5743	00874.HK	**0.5429**	0.5243
00875.HK	0.4471	**0.5357**	00876.HK	0.5671	**0.5829**
00877.HK	0.4929	**0.5057**	00878.HK	**0.4986**	0.4614
00880.HK	0.5	**0.5114**	00881.HK	**0.52**	0.5114
00882.HK	0.49	**0.5314**	00883.HK	**0.4929**	0.4686
00884.HK	0.5329	**0.5614**	00885.HK	**0.6743**	0.6443
00886.HK	**0.6086**	0.5729	00887.HK	0.6271	**0.63**
00888.HK	0.6371	**0.6614**	00891.HK	0.5529	**0.5643**
00893.HK	**0.5514**	0.5257	00894.HK	**0.5886**	0.58
00895.HK	0.4886	**0.5**	00896.HK	**0.6429**	0.62
00897.HK	0.5429	**0.5643**	00899.HK	0.5514	**0.5614**
AttLSTM Win	39	Total Num	78	AttLSTM Win	50.00%

A.2 Predicted Investment Performance

Name	Stock Performance	LSTM	AttLSTM	Name	Stock Performance	LSTM	AttLSTM
00800.HK	0.5268	**1.3125**	0.5446	00801.HK	0.6891	**7.9405**	**7.9405**
00802.HK	0.7593	**4.43**	**4.43**	00803.HK	0.2442	**4.2618**	**4.2618**
00805.HK	**1.3542**	0.5555	0.9151	00806.HK	0.6574	2.2847	**2.579**
00808.HK	1.025	**1.0715**	**1.0715**	00809.HK	0.2731	0.1934	**0.6609**
00810.HK	0.5089	**63.876**	5.9173	00811.HK	**0.6898**	0.5356	0.4936
00812.HK	0.2213	**6.4876**	0.9002	00813.HK	1.1638	0.3404	**2.0842**
00814.HK	0.8041	0.7347	**1.3772**	00815.HK	**0.3417**	0.2262	0.085
00816.HK	0.725	**1.6948**	1.4285	00817.HK	**1.2571**	0.232	0.1887
00818.HK	0.6159	1.1735	**1.8642**	00819.HK	**0.8972**	0.8865	0.5105
00820.HK	0.7965	**6.4313**	2.7123	00821.HK	0.5714	**0.7452**	0.1628
00822.HK	0.614	0.7703	**2.4727**	00823.HK	1.5991	0.4539	**1.6033**
00825.HK	1.3739	**3.8419**	0.6136	00826.HK	**2.1912**	0.6883	1.5711
00827.HK	0.3241	**1.4368**	0.1237	00829.HK	0.3942	0.5519	**0.7749**
00830.HK	0.4715	0.5841	**0.8282**	00832.HK	**1.9221**	1.7611	1.1911
00833.HK	1.3857	**1.4994**	0.4872	00836.HK	**0.9337**	0.5079	0.4664
00837.HK	**0.9908**	0.9723	0.9479	00838.HK	**0.4532**	0.3884	0.2742
00840.HK	0.3523	**186.9746**	**186.9746**	00841.HK	0.4802	**17.494**	7.5665
00842.HK	0.9429	1.5612	**2.1471**	00844.HK	0.4182	**1.4796**	0.6301
00845.HK	0.4096	22.4856	**26.1062**	00848.HK	**0.6154**	0.5371	0.5371
00850.HK	0.9008	0.2358	**2.3579**	00851.HK	0.8	**1567.5624**	**1567.5624**
00852.HK	**1.5455**	1.1402	0.7838	00853.HK	**2.8625**	1.8384	1.7227
00855.HK	**1.8468**	0.6134	1.1957	00856.HK	**1.658**	0.7563	1.3982
00857.HK	**1.1873**	0.6625	0.6835	00858.HK	0.3333	**36344.8645**	8469.3245
00859.HK	**0.7685**	0.1233	0.7475	00860.HK	2.3	**24.3111**	0.3744
00861.HK	0.4747	**4.8289**	1.1747	00862.HK	0.6921	**22.2528**	15.1996
00864.HK	0.6222	65.0636	**150.4373**	00866.HK	0.9825	**3.7062**	0.1576
00867.HK	0.8774	**0.8956**	0.1975	00868.HK	**2.1988**	1.3587	1.451
00869.HK	0.3636	0.7652	**1.4107**	00871.HK	0.1	**0.4855**	0.187
00872.HK	0.5593	7.7051	**20.161**	00874.HK	1.1778	**2.5758**	0.4799
00875.HK	0.0255	**1.1654**	0.0882	00876.HK	0.53	7.5831	**13.5191**
00877.HK	**1.5**	0.6716	0.5402	00878.HK	1.0899	0.9038	**1.1185**
00880.HK	**1.3119**	1.0447	0.6578	00881.HK	**3.515**	0.2016	1.3774
00882.HK	0.5476	0.7777	**1.331**	00883.HK	**1.8696**	0.667	1.303
00884.HK	**1.7758**	0.9765	0.5647	00885.HK	**0.2586**	0.2054	0.2054
00886.HK	**0.6258**	0.1328	0.2951	00887.HK	1.575	**12.4248**	10.3661
00888.HK	1.9032	2.5627	**7.7713**	00891.HK	0.3073	**1.8221**	1.6576
00893.HK	1.4807	**10.2408**	9.6007	00894.HK	0.8375	2.6886	**6.0122**
00895.HK	0.6599	**0.9028**	0.889	00896.HK	1.1739	1.6854	**2.0676**
00897.HK	0.1306	**9.0129**	8.3132	00899.HK	0.1919	**3.3153**	1.591

B Deep Learning for Sequence Classification

B.1 LSTM

Long Short-Term Memory, is a special type of Recurrent Neural Network (RNN). An RNN is a network with a loop inside of it which allows learned information to be passed along. However, once an RNN learns a lot of information, it can have trouble using older information. LSTMs solve this problem. Introduced in 1997 by Hochreiter and Schmidhuber, LSTM networks are extremely common today. Instead of containing a single layer like a standard RNN, an LSTM will contain a number of interacting layers. The LSTM has the addition of sigmoid layers, which output numbers

between zero and one and tell us how much of the information to let through each part of the cell.

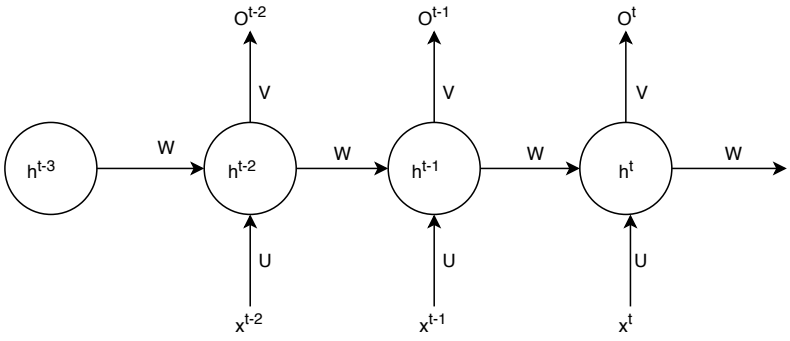


Figure 5: RNN

The first layer in a standard LSTM cell is a sigmoid function, which decides which data we are going to keep. This layer is known as the forget gate. After this, we need to decide what will be stored within this layer. A sigmoid layer known as the input gate is used to decide which values need to be update. This is passed along to a tanh layer which creates a vector of values which may be added to the state. We refer to this vector as the candidate vector. By multiplying the candidate vector by the vector output of the input gate, we identify how much to update each state by. We pass through one more layer which decides the output of the cell. This layer operates similarly to the previous layer. It mixes a sigmoid gate with a tanh function and only outputs the relevant information.

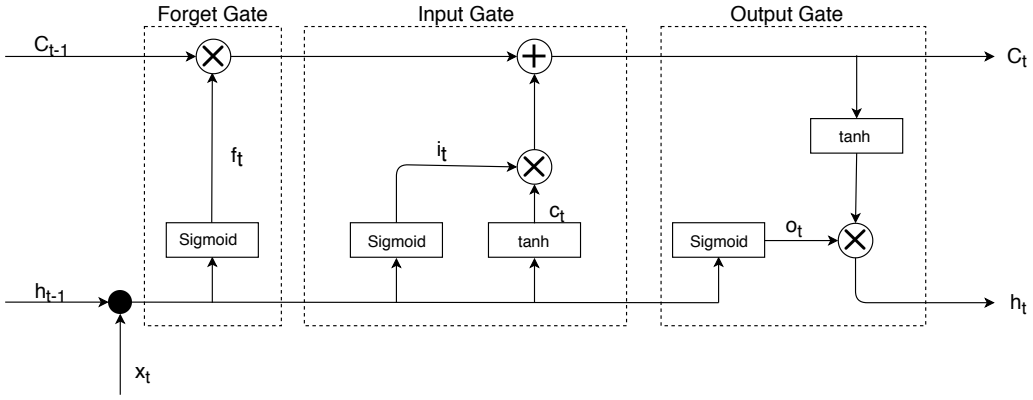


Figure 6: LSTM

B.2 Attention

While LSTM provides an improvement to the long-term memory problem of RNNs, they are not a perfect solution. A limitation of the encoder-decoder architecture is that it encodes the input sequence to a fixed length internal representation, resulting in worse performance for long sequences. As a potential solution, we can use an Attention Mechanism, reducing the burden of encoding all the information. Instead of trying to encode the full set of data, we let the decoder "attend" to different parts of the data at each point in the output generation. The model trains itself to on

what to attend to based on the input and its output. What this does is create an output y_i which depends on a weighted combination of all of the previous input states.

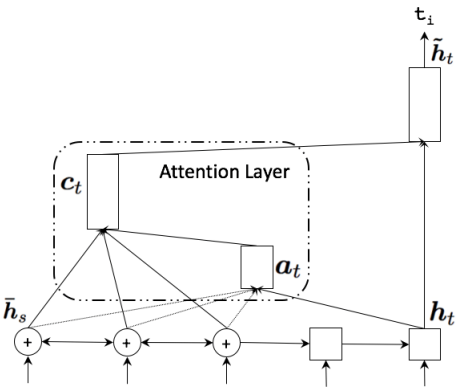


Figure 7: Attention Mechanism

B.3 Preventing Overfitting

Dropout and early stopping are both used to prevent over-fitting. Dropout is a form of regularization where random neurons are ignored during each pass. This adds a penalty to the loss function so that the model does not learn an interdependent set of feature weights.

In addition to dropout, we use early stopping to avoid over-fitting. At the end of each iteration, we check to see if the cross-entropy loss function has improved beyond a specified parameter. If after a certain amount of iterations the loss function has failed to improve during validation, early stopping is employed.

B.4 Batch Normalization

Batch Normalization is a methodology used to help neural networks converge in less time, utilize higher learning rates, and prevent non-linear activation functions from saturation early in the training. To do this, Batch Normalization normalizes the layer using the statistics of the last batch trained. This solves the problem of reducing the internal covariate shift.

The first step of Batch Normalization is to calculate the mean (1) and variance (2) of the last batch.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{2}$$

From there, we proceed to normalize the inputs of the layer using these statistics.

$$\bar{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3}$$

And finally, we scale and shift. Observe that γ and β are learned.

$$y_i = \gamma \hat{x_i} + \beta \tag{4}$$

After performing the Batch Normalization, one can expect to see faster convergence. This is because the distribution of each layer's inputs are changed while training. This is the internal covariate shift which Batch Normalization is helping to reduce.

C Technical indicators used

- **ROCP** Rate of Change Percentage is the percentage change between the price today and the price of last day. We used ROCP of open, close, high and low price.
- **MA** Moving Average is the average value of sequential data in a specified time period. We used MAs of close price and their percentage changes for 5, 10, 20, 30 days.
- **MACD** Moving Average Convergence/Divergence is an indicator mainly to reveal the change in the momentum of stock price trend. A momentum oscillator is constructed by subtracting the longer EMA from the shorter EMA. We used 12 days as fast period and 26 days as the slow period.
- **RSI** Relative Strength Index is an indicator measures the relative strength between the momentum of positive change in price and the momentum of negative change during a specified time period. We used RSIs and their percentage changes for 6, 16 and 24 days.
- **VROCP** Volume Rate of Change Percentage measures a volume trend of a stock. A positive VROCP means the market support the price to continue moving in the current trend while a negative VROCP means a lack support on the moving in the current trend, and the current price trend maybe disappear or reverse. We used VROCPs for 5, 10, 20, 30 days.
- **BOLL** Bollinger Bands include three indicators which are middle band, upper band and lower band. Middle band could be calculated by moving average. Upper/lower band is obtained by middle band plus/minus K times an N-period standard deviation. These three bands are supposed to reveal high or low stock price.
- **VMA** VMA measures a volume trend of a stock with a positive/negative value revealing that the market support/do not support the price to continue moving in current trend. We used VMAs for 5, 10, 20, 30 days.