Application of Deep Learning to Algorithmic Trading

Guanting Chen [guanting][1], Yatong Chen [yatong][2], and Takahiro Fushimi [tfushimi][3]

[1]Institute of Computational and Mathematical Engineering, Stanford University
[2]Department of Civil and Environmental Engineering, Stanford University
[3]Department of Management Science and Engineering, Stanford University

I. Introduction

Deep Learning has been proven to be a powerful machine learning tool in recent years, and it has a wide variety of applications. However, applications of deep learning in the field of computational finance are still limited (Arévalo, Niño, Hernández & Sandoval, 2016). In this project, we implement Long Short-Term Memory (LSTM) network, a time series version of Deep Neural Networks, to forecast the stock price of Intel Corporation (NASDAQ: INTC). LSTM was first developed by Hochreiter & Schmidhuber (1997). Over the years, it has been applied to various problems that involve sequential data, and research has demonstrated successful applications in such fields as natural language processing, speech recognition, and DNA sequence.

The input features we use are categorized into three classes: (1) the historical trading data of INTC (OHLC variables), (2) commonly used technical indicators derived from OHLC variables, and (3) the index of the financial market and the semiconductor sector are fed into the network. All of these features reflect daily values of these variables, and the network predicts the next day's adjusted closing price of INTC based on information available up to the current day.

Our experiment is composed of three steps. First, we choose the best model by training the network and evaluating its performance on a dev set. Second, we make a prediction on a test set with the selected model. Third, given the trained network, we examine the profitability of an algorithmic trading strategy based on the prediction made by the model. For the sake of comparison, Locally Weighted Regression (LWR) is also performed as a baseline model.

The rest of this paper is organized as follows: Section II discusses existing papers and the strengths and weaknesses of their models. Section III describes the dataset used in the experiment. Section IV explains the models. Section V defines the trading strategy. Section VI illustrates the experiment and the results. Section VII is the conclusion, and Section VIII discusses future work.

II. Related Work

Existing studies that apply classical neural networks with a few numbers of hidden layers to stock market prediction problems have had rather unsatisfactory performance. For instance, Sheta, Ahmed & Faris (2015) examines Artificial Neural Network (ANN) with 1 hidden layer in addition to multiple linear regression and Support Vector Machine (SVM) with a comprehensive set of financial and economic factors to predict S&P500 index. Even though these methods can forecast the market trend if they are correctly trained, it is concluded that SVM with RBF kernel outperforms ANN and the regression model. Guresen, Kayakutlu & Daim (2011) analyzes some extensions of ANN such as dynamic artificial neural network and the hybrid neural networks which use generalized autoregressive conditional heteroscedasticity. Their experiment, however, demonstrates that these sophisticated models are unable to forecast NASDAQ index with a high degree of accuracy.

On the other hand, Deep Learning models with multiple layers have been shown as a promising architecture that can be more suitable for predicting financial time series data. Arevalo et al., (2016) trains 5-layer Deep Learning Network on high-frequency data of Apple's stock price, and their trading strategy based on the Deep Learning produces 81% successful trade and a 66% of directional accuracy on a test set. Bao, Yue & Rao (2017) proposes a prediction framework for financial time series data that integrates Wavelet transformation, Stacked Autoencoders, and LSTM. Their network with 10 hidden layers outperforms the canonical RNN and LSTM in terms of predictive accuracy. Takeuchi & Lee (2013) also uses a 5-layer Autoencoder of stacked Boltzmann machine to enhance Momentum trading strategies that generates 45.93% annualized return.

III. Dataset and Features

A. Dataset

Our raw dataset is the historical daily price data of INTC from 01/04/2010 to 06/30/2017, sourced from Yahoo! Finance. In order to examine the robustness of the models in different time periods, the dataset is devided into three periods.

Fig. 1: Stock price of Intel from 01/04/2010 to 06/30/2017. The black arrows indicate the training sets, the red arrows indicate the dev sets, and the blue arrows indicate the of test sets.

Period I ranges from 01/04/2010 to 06/29/2012. Period II ranges from 07/02/2012 to 12/31/2014. Period III ranges from 01/02/2015 to 06/30/2017. Furthermore, we divide each sub-dataset into the training set, the dev set and the test set, and the length of their periods is 2 years, 3 months, and 3 months, respectively. The historical stock price of INTC is shown in Figure 1, which also illustrates how we split the data into the three different sets.

B. Input Features

The input features consist of three sets of variables. The first set is the historical daily trading data of INTC including previous 5 day's adjusted closing price, log returns, and OHLC variables. These features provide basic information about INTC stock. The second set is the technical indicators that demonstrate various characteristics of the stock's behavior. The third set is about indexes: S&P500, CBOE Volatility Index, and PHLX Semiconductor Sector Index. Figure 2 describes the details of these variables. All of the inputs and output are scaled between 0 and 1 before we feed them into the models.

Input Features	Definition/Explanation
	Category 1. Daily Trading Data
Open/Close Price	Nominal daily open/close price
High/Low Price	Nominal daily highest/lowest price
Trading Volume	Daily Trading Volume
	Category 2. Technical Indicator
RA	Rolling average: standard deviation in 5/10 days windows
MACD	Moving Average convergence Divergence: a display trend following characteristics and momentum characteristics
CCI	Commodity Change: an identification of cyclical trend
ATR	Average True Range: a measurement of the volatility of price
BOLL	Bollinger Band: two standard deviation from the moving average
MA5/MA10	5/10 days moving average
MTM1/MTM3	1/3 month Momentum: the difference between current price and the price 1 or 3 months ago
ROC	Price rate of change: the momentum divided by the price 3 month
WPR	William Percent Range: a measurement of the buying and selling pressure
	Category 3. General Market Index
S&P 500	An American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE/ NASDAQ
VIX	The CBOE Volatility Index: a popular measure of the stock market's expectation of volatility
SOX	The PHLX Semiconductor Sector: an index composed of companies primarily involved in the design, distribution, manufacture, and sale of semiconductors.

Fig. 2: Descriptions of the input features [2]

IV. Methods

A. Long Short-Term Memory

Long Short-Term Memory (LSTM) was first developed by Hochreiter & Schmidhuber (1997) as a variant of Recurrent Neural Network (RNN). Over the years, LSTM has been applied to various problems that involve sequential data, and research has demonstrated successful applications in such fields as natural language processing, speech recognition, and DNA sequence.

Like RNN, LSTM has a recurrent structure where each cell not only outputs prediction $\hat{y}_t$ but also transfers activation h_t to the next cell. The striking feature of LSTM is its ability to store, forget, and read information from the long-term state of the underlying dynamics, and these tasks are achieved through three types of gates. In the *forget gate*, a cell receives long-term state c_{t-1}, retains some pieces of the information by amount f_t, and then adds new memories that the *input gate* selected. The *input gate* determines what parts of the transformed input g_t need to be added to the long-term state c_t. This process updates long-term state c_t, which is directly transmitted to the next cell. Finally, *output gate* transforms the updated long-term state c_t through $\tanh(\cdot)$, filters it by o_t, and produces the output $\hat{y}_t$, which is also sent to the next cell as the short-term state h_t.

The equations for LSTM computations are given by

$$
\begin{aligned}
i_t &= \sigma(W_{xi}^T \cdot x_t + W_{hi}^T \cdot h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}^T \cdot x_t + W_{hf}^T \cdot h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}^T \cdot x_t + W_{ho}^T \cdot h_{t-1} + b_o) \\
g_t &= \tanh(W_{xg}^T \cdot x_t + W_{hg}^T \cdot h_{t-1} + b_g) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\
\hat{y}_t &= h_t = o_t \otimes \tanh(c_t)
\end{aligned}
$$

where $\otimes$ is element-wise multiplication, $\sigma(\cdot)$ is the logistic function, and $\tanh(\cdot)$ is the hyperbolic tangent function. The three gates open and close according to the value of the gate controllers f_t, i_t, and o_t, all of which are fully connected layers of neurons. The range of their outputs is [0, 1] as they use the logistic function for activation. In each gate, their outputs are fed into element-wise multiplication operations, so if the output is close to 0, the gate is narrowed and less memory is stored in c_t, while if the output is close to 1, the gate is more widely open, letting more memory flow through the gate. Given LSTM cells, it is common to stack multiple layers of the cells to make the model deeper to be able to capture nonlinearity of the data. Figure 3 illustrates how computation is carried out in a LSTM cell.

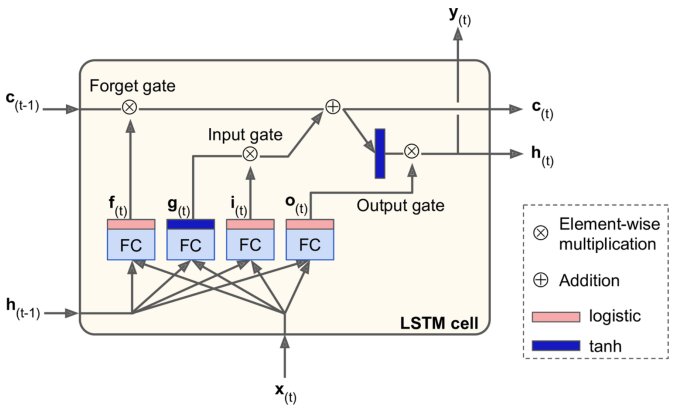

Fig. 3: LSTM cell [3]

We choose Mean Squared Error (MSE) with L^2-regularization on the weights for the cost function:

$$L(\theta) = \frac{1}{m}\sum_{t=1}^{m}(y_t - \hat{y}_t)^2 + \lambda\left(||W_i||_2 + ||W_f||_2 + ||W_o||_2\right)$$

where θ is the set of parameters to be trained including the weight matrices for each gate $\{W_i = (W_{xi}; W_{hi}), W_f = (W_{xf}; W_{hf}), W_o = (W_{xo}; W_{ho})\}$, and the bias terms $\{b_i, b_f, b_o, b_g\}$.

Since this is a RNN, LSTM is trained via *Backpropagation Through Time*. The key idea is that for each cell, we first unroll the fixed number of previous cells and then apply forward feed and backpropagation to the unrolled cells. The number of unrolled cells is another hyperparameter that needs to be selected in addition to the number of neurons and layers.

B. Locally Weighted Linear Regression:

Locally Weighted Linear Regression is a nonparametric model that solves the following problem at each target point x_0:

$$\theta(x_0) = \underset{\theta}{\operatorname{argmin}} \frac{1}{2}(X\theta - y)^T W(x_0)(X\theta - y).$$

$X \in R^{m,n}$ is the matrix of covariates, and $y \in R^m$ is the outcome. The weight matrix $W(x_0) = \text{diag}\{w_1, \cdots, w_m\}$ measures the closeness of x_0 relative to the observed points $x_t, t = 1, \cdots, m$. The closer x_0 is to x_t, the higher the value of w_t. The Gaussian kernel is chosen as the weight:

$$w_t(x_0) = \exp\left(-\frac{(x_t - x_0)^2}{2\tau^2}\right)$$

where τ is a hyper parameter that controls the window size. The closed form solution of the optimization problem is

$$\hat{\theta}(x_0) = (X^T W(x_0) X)^{-1} X^T W(x_0) y.$$

The predicted value at the new target point x is

$$\begin{aligned}\hat{y} &= x^T \hat{\theta}(x) \\ &= x^T (X^T W(x) X)^{-1} X^T W(x) y.\end{aligned}$$

Although the model is locally linear in y, computing $\hat{y}$ over the entire data set produces a curve that approximates the true function.

V. Trading Strategy

We consider a simple algorithmic trading strategy based on the prediction by the model. At day t, an investor buys one share of INTC stock if the predicted price is higher than the current actual adjusted closing price. Otherwise, he or she sells one share of INTC stock. The strategy s_t can be described as:

$$s_t = \begin{cases} +1 & \text{if } \hat{y}_{t+1} > y_t \\ -1 & \text{if } \hat{y}_{t+1} \leq y_t \end{cases}$$

where y_t is the current adjusted closing price of `INTC` and $\hat{y}_{t+1}$ is the predicted price by the model. Using the indicator variable s_t, we can calculate a daily return of the strategy at day $t+1$:

$$r_{t+1} = s_t \times \log\left(\frac{y_{t+1}}{y_t}\right) - c$$

where c denotes transaction cost, and the cumulative return from $t = 0$ to m is

$$r_0^m = \sum_{t=0}^{m-1} r_t.$$

We also consider the Sharpe Ratio to compare the profitability of the strategy with different models. The Sharpe Ratio is defined as

$$\text{SR} = \frac{E(r) - r_f}{\sigma(r)}.$$

$E(r)$ is the expected return of a stock, r_f is the risk-free rate, and $\sigma(r)$ is the standard deviation of the return.

VI. Experiment and Results

A. Setup for the Experiment

We use *TensorFlow* to perform our experiment on the dataset. The choice of hyperparameters and optimizer are listed in Table I:

Categories	Choice
Library	*TensorFlow*
Optimizer	*AdamOptimizer*
The number of Hidden Layers	5
The number of Unrolled Cells	10
The number of Neurons	200
The number of Epochs	5000

TABLE I: Hyperparameters/Optimizer for the LSTM Model

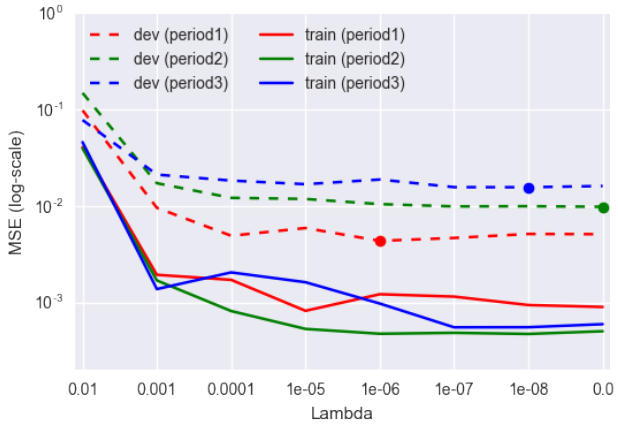

Fig. 4: MSE on the training and dev sets with different values of λ (The points indicate the lowest MSE on the dev set for each period)

We choose 5 layers in line with the networks used by Arevalo et al., (2016), Bao et al., (2017), and Takeuchi, & Lee (2013). Regarding the number of unrolled cells, 10 (days) is assumed to be sufficient for the LSTM to predict the next day's stock price and avoid the vanishing gradient problem. The number of neurons is determined by try and error.

Since the architecture of the LSTM is 5 layers with 200 neurons, which is deep and wide, it is necessary to introduce regularization as discussed in Section IV in order to avoid overfitting and improve the predictive accuracy. For each period, we train the network with different values of λ that controls the degree of regularization and compute MSE on the training and dev set. The set of values of λ we consider is $0.1^n (n = 2, \cdots, 8)$ and 0 that corresponds to no regularization. The MSE on the training and dev sets with different λ's is illustrated in Figure 4. We select the λ that minimizes the MSE on dev set which is depicted as points in the plot.

AdamOptimizer is selected because it is suitable for deep learning problems with large dataset and many parameters. As for the parameters of *AdamOptimizer*, we use the default values provided by *Tensorflow*.

B. The results of the experiment

Figure 5 shows the predicted stock price by both models and the actual price of INTC. The predicted prices of the LSTM are closer to the actual price than the ones of the LWR for all the three periods. It is important to note that the LSTM seems to be able to predict the stock price more accurately when the price does not exhibit a clear trend such as the first and third periods than when the price is boosting like the second period where the LSTM constantly underestimates it.

To further evaluate the predictive performance of the models, we calculate two measurements and examine

Fig. 5: Predicted and actual price of Intel for three periods (scaled)

the profitability of the algorithmic trading strategy.

1) Mean Squared Error of Predicted Price:

$$\text{MSE} = \frac{1}{m}\sum_{t=1}^{m}(y_t - \hat{y}_t)^2$$

where y_t and $\hat{y}_t$ denote the actual and predicted prices of INTC at day t. The MSE of the both models are listed in Table II. The MSE of the LSTM on the test sets turns out to be small and lower than that of the LWR Models for all the three periods. This result substantiates that the LSTM achieves higher predictive accuracy than the LWR.

Period	Train Error		Dev Error		Test Error	
	LSTM	LWR	LSTM	LWR	LSTM	LWR
I	0.00122	0.05571	0.00436	0.10617	0.00564	0.05362
II	0.00051	0.00702	0.00984	0.04488	0.00772	0.01548
III	0.00056	0.09385	0.01568	0.03331	0.01146	0.02147

TABLE II: The MSE of the LSTM and LWR for the three periods

2) Directional Accuracy:

$$DA = \frac{1}{m-1}\sum_{t=1}^{m-1} \mathbb{1}\{(y_{t+1} - y_t)(\hat{y}_{t+1} - \hat{y}_t) > 0\}$$

$\mathbb{1}\{\cdot\}$ is the indicator function. This measures the proportion of the days when the model forecasts the correct direction of price movement. The DA on the test set is summarized in Table III. The LSTM accomplishes

80.328% and 70.968% for the first and third periods, while the prediction by the LWR is slightly higher than random guess in the same periods. The second period in which the price of INTC continued to rise appears to be a challenging moment for both models to make an accurate prediction.

Period	LSTM	LWR
I	80.328	57.377
II	59.616	45.902
III	70.968	57.377

TABLE III: Directional accuracy(%) of the LSTM and LWR for three periods

3) Cumulative Daily Returns and Sharpe Ratio:

$$r_0^m = \sum_{t=0}^{m-1} r_t \text{ and } SR = \frac{E(r) - r_f}{\sigma(r)}$$

as defined in Section V. The cumulative daily returns and the Sharpe Ratio for the strategy based on the LSTM and the LWR are shown in Table IV. The transaction cost of each trade is assumed to be 1 basis point. 13 week treasury bill rate (IRX) is used as the risk-free rate. As a comparison, we also consider a Buy-and-Hold strategy in which an investor buys one share of INTC at the beginning of a test set and holds it until the end of the period. The LSTM-based strategy produces promising cumulative returns and the Sharpe Ratio. In particular, the strategy yields a 54.8% cumulative daily return and a 0.649 Sharpe Ratio during the first period despite the negative return of INTC (i.e., Buy-and-Hold strategy). The LWR also generates higher returns and Sharpe Ratio than the Buy-and-Hold strategy, but the LSTM substantially outperforms the LWR. Figure 6 illustrates the cumulative daily returns of the strategies. The return of the LSTM-based strategy stably increases over time in the first and second periods.

Period	LSTM		LWR		Buy& Hold	
	Returns	SR	Returns	SR	Returns	SR
I	54.8%	0.649	25.5%	0.292	-8.3%	-0.108
II	28.6%	0.257	15.3%	0.141	10.4%	0.088
III	16.9%	0.267	5.4%	0.128	-6.6%	-0.085

TABLE IV: Cumulative daily returns and the Sharpe Ratio of the strategies for three periods

VII. CONCLUSION

In this project, we implement Long Short-Term Memory Network to predict the stock price of INTC and apply the trained network to the algorithmic trading problem. The LSTM can accurately predict the next day's price of INTC especially when the stock price is lack of a trend. The strategy based on the prediction by the LSTM makes promising cumulative daily returns, outperforming the other two strategies. These results, however, are limited

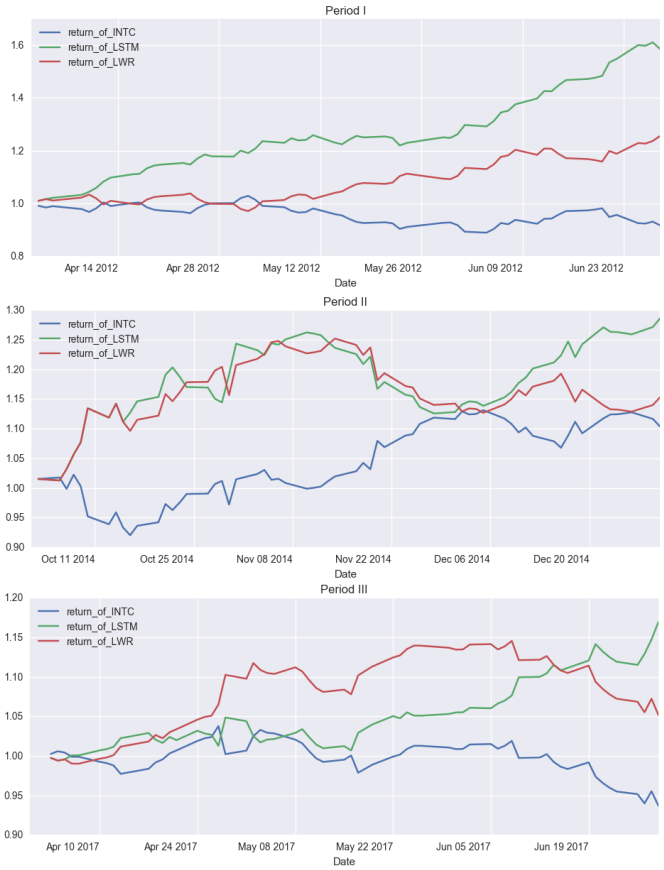

Fig. 6: Cumulative daily return of the Buy-and-Hold (blue line), the LSTM-based (green), and the LWR-based (red line) strategies.

in some respects. We assume that it is always possible to trade at the adjusted closing price every day, which is not feasible in practice. Yet, our study demonstrates the potential of LSTM in stock market prediction and algorithmic trading.

VIII. FUTURE WORK

Due to the computational limitation, we were unable to conduct a comprehensive experiment to train various architectures of the LSTM such as different numbers of neurons and layers. Also, one of the biggest weaknesses of our LSTM is that it cannot capture a sheer trend like the one observed in the test set of the second period. Thus, a possible extension of our approach can be to increase the number of layers to make the network even deeper and build the trading strategy combined with reinforcement learning that takes into account the current state of the market. Another approach would be to make the network event-driven so that it can respond to structural changes in the financial market.

IX. CONTRIBUTION

All team members contributed to the progress of the project. Specific work assignment is as follows. Takahiro Fushimi implemented LSTM using TensorFlow, created figures, and wrote the final report. Yatong Chen wrote the milestone and final reports, performed analysis, and made

poster. Guanting Chen helped processing and testing data, advised machine learning methods, and implemented machine learning models.

REFERENCES

[1] Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016, August). High-frequency trading strategy based on deep neural networks. In International conference on intelligent computing (pp. 424-436). Springer International Publishing.

[2] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.

[3] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.

[4] Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. Expert Systems with Applications, 38(8), 10389-10397.

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[6] Sheta, A. F., Ahmed, S. E. M., & Faris, H. (2015). A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. Soft Computing, 7, 8.

[7] Takeuchi, L., & Lee, Y. Y. A. (2013). Applying deep learning to enhance momentum trading strategies in stocks. Working paper, Stanford University.