Recurrent Neural Networks for Financial Time-Series Modelling

Gavin Tsang, Jingjing Deng, and Xianghua Xie
Computer Science Department, Swansea University, United Kingdom
{658679, J.Deng, X.Xie}@swansea.ac.uk
http://csvision.swan.ac.uk/

***Abstract*—The prediction of financial time series data is a challenging task due to the unpredictable behaviours of investors that are influenced by a multitude of factors. In this paper, we present a novel deep Long Short-Term Memory (LSTM) based time-series data modelling for use in stock market index prediction. A dataset comprised of six market indices from around the world were chosen to demonstrate the robustness in varying market conditions with an aim to forecast the next day closing price. With experimental results showing an average annual profitability performance of up to 200%, our method demonstrates its feasibility and significant results in time-series modelling and prediction of financial markets.**

I. Introduction and Related Work

Stock market data, and by extension any financial time series data, is highly complex and difficult to predict. With investors being influenced by volatile and unpredictable market conditions, financial market trends tend to be non-linear, uncertain and non-stationary. Great strides in financial data modelling and prediction have occurred over the past 50 years with very early research declaring the task almost impossible [1]. Recently, there have been many attempts made, which have shown the negative judgement was overstated and great improvements in accuracy have been observed in price prediction for financial products [2], [3]. Sequential machine learning methods have been widely adopted for financial time-series prediction, where pricing models are built to find the correlations among target values, movement trends, technical indicators and social sentiments. *Choudhry et. al.* [4] proposed a hybrid method based on Genetic Algorithm (GA) and Support Vector Machines (SVM) for stock market prediction, which makes use of the correlation between stock prices of different companies and technical indicators. A GA is used for feature selection among 35 technical indicators, where a binary SVM classifier is followed to predict drop or rise movement of target stock price. Similarly, *Iacomin* [5] proposed a combination of SVM and feature selection using Principal Component Analysis (PCA) to predict 16 foreign exchange stocks around the world. Instead of predicting price movement, *Iacomin* [5] uses SVM to make the selling or buying decision a binary classification problem in order to maximize profit. In addition, there are many works investigating the influences of social sentiments on price movement. For instance, a $L1$ regularised logistic regression model was proposed by *Luo et. al.* [6] using specific events extracted from target companies together with financial, macro and technical indicators to model price fluctuations after major announcements, but a limited success was observed. It is still controversial on what variables have the most significance in prediction performance. However, traditional machine learning methods heavily rely on hand-crafted feature selections which have proved to be non-trivial tasks and have limited success in forecasting the actual product price directly.

Recent advances in Deep Learning (DL) [7], more specifically Deep Neural Networks (DNNs) [8], have shown great successes in the fields of computer vision, speech recognition and health informatics to name a few. However, there are a limited number of published works on financial modelling using deep learning methodologies reported in the field [9]. Existing works show that Neural Networks (NNs) are able to provide effective means of modelling markets through its ability to learn robust and non-linear correlation, and it also draws a parallel to that of investment decisions where various verdicts are made based upon many known factors and the complex associations between them. As a result of representation learning, the limitation caused by feature hand-crafting no longer exists. The family of Recurrent Neural Networks (RNN) have recursive feedback connections between neuron cells forming a directed cycle, which are able to retain and leverage information from past data to aid the prediction of future events. Similar to Hidden Markov Models (HMMs), such recurrent architectures by nature are suitable for modelling sequential data with delayed temporal correlations. Many recent works with RNN family have shown good promise in stock price prediction using either technical indicators [10] or social sentiments [11].

Among those works, of significant interest is the work by *Bao. et. al.* [12] showing significant results of a Wavelet Stacked AutoEncoders-Long Term Short Memory (WSAEs-LSTM) model integrating wavelet transforms, Stacked AutoEncoders (SAEs) and Long Short Term Memory (LSTM). WSAEs-LSTM consists of a three stage pipeline. Initially, a Haar wavelet transform is used to denoise the original highly oscillated signal and produce localised coefficients in frequency domain as inputs for the next stage. A five layer SAE with sigmoid activations is trained using standard gradient descent with Mean Average Error (MAE) loss and Kullback-Leibler (KL) Divergence sparsity penalty. The further generalised representations of technical indicators are given by SAE encoder, and then passed into a 5 layer, 10 unit

LSTM network to forecast next day closing price.

In this work, we show that the WSAEs-LSTM method suffers from detailed feature loss due to denoising with hard thresholding, poor initialisation and under-training issues. We propose a financial time-series prediction method which follows the overall concept of preprocessing followed by feature generalization and then sequential modelling using SAEs and LSTMs. However, compared to [12], several improvements have been made to the overall model. Firstly, preprocessing and denoising were performed using soft *Symlet* wavelet thresholding to produce a denoised variant of original market data. This was then used to pre-train a 5 layer L2 regularised SAE using Exponential Linear Unit (ELU) activation functions and trained using the *AdaDelta* optimiser [13]. Such combination ensures the learning of a deep representation throughout the deep architecture without the drawbacks of vanishing gradients due to the use of sigmoid activations. The trained SAE is then incorporated into a 2 layer LSTM network to perform final prediction. Training, using the *AdaDelta* optimiser, was a matter of fine tuning the overall network to each dataset. As seen in the experimental results of this paper, our proposed model greatly improves upon prediction and profitability performance of WSAEs-LSTM across all 6 datasets.

The structure of paper is organised as follows. In Section II, the algorithms and properties used in the presented framework are described. Section III introduces the experimental dataset and provides evaluation procedure and metrics. Section IV summarises the observed results, and provides our concluding remarks.

II. Method

The framework presented in this paper comprises of a 3-stage pipeline as illustrated in Fig. 1. At the preprocessing stage, the orginal financial time-series data is denoised in the spatial and spectral domain jointly using wavelet thresholding. Market modelling utilises SAEs and finally one-day-ahead prediction of market closing price using LSTM are then followed in stage 2 and 3 respectively. Further details of each stage are presented within the remainder of this section.

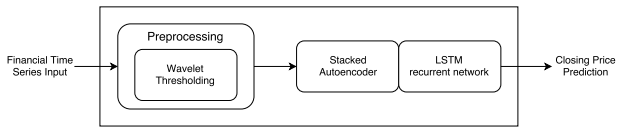

Fig. 1. The pipeline of proposed model.

A. Preprocessing

Each subset was normalised based only upon the minimum-maximum values of their corresponding training set in order to eliminate any prior knowledge of overall scale as would occur in real-time prediction. Wavelet thresholding was used for signal denoising due to its capability in handling highly irregular financial time series data. Through the ability to denoise whilst preserving localised features in both space and time, wavelet thresholding is superior to regular Fourier based denoising methods in regards to time series data.

Wavelet thresholding consists of a three stage process consisting of multilevel wavelet decomposition, soft thresholding of coefficients and finally signal reconstruction. Wavelet decomposition involves the transformation of a basis function, called the *Mother Wavelet*, into closely matching the original signal using the following equation:

$$W_\psi f(k,s) = \frac{1}{\sqrt{|k|}} \int_{-\infty}^{\infty} \overline{\psi\left(\frac{t-k}{s}\right)} f(t)dt \qquad (1)$$

where $W_\psi f(k,s)$ is the transformed mother wavelet, ψ is the mother wavelet (*Symlet-8*), k is the wavelet shift along the time axis, s is the scaling factor along time and t the current point in the time series data. An important aspect to note is that s is inversely proportional to t, resulting in an inverse relationship between the wavelet frequency and wavelet scale along time. Accordingly, wavelet decomposition captures both long, low frequency trends over a long period of time and short term high frequency noise. The generation of wavelets was performed through the use of the *Mallat Algorithm* which involves passing a signal through a series of high and low pass convolution filters derived from the mother wavelet [14]. The two signals are then down-sampled to half size to generate a detail and approximation coefficients accordingly. The approximation coefficients are then recursively devolved to the desired or maximum level.

Soft thresholding consists of reducing detail coefficients at each level below a adaptive threshold to zero and reducing the remaining by said threshold. Threshold selection per level was performed using the *SURE-Shrink* method [14], due to it being smoothness adaptive and as such, able to handle irregular signals. Reconstruction involves passing the altered detail and approximation coefficients back through up-sampling and a series of reconstruction convolution filters to arrive back at the denoised signal.

B. Stacked Autoencoder

Autoencoders are used to pre-train a network into learning latent encoding in data. Once trained, the encoding layers of an AE can be incorporated into a larger network which can make use of the consistent representations learnt. SAEs have proven to be an effective pre-training method consistently outperforming non-encoded networks [15]. SAEs consist of iteratively layering single-layer AE on top of each other where the encoded output of the previous layer is used as the input for the next. Each layer is individually and progressively trained much like a regular AE until the desired depth is achieved.

The aim of AE is to non-linearly transform the input into a higher level abstraction before reconstructing back into the original input. Sigma function selected for all encoding layers within this framework is the Exponential Linear Unit (ELU) providing all the advantages of Rectified Linear Unit (ReLU) function such as eliminating vanishing gradients on deep neural networks while also removing the drawback of dying neurons due to highly negatively weighted inputs. A

L2 weight regularisation parameter is also included within the objective function to reduce over-fitting:

$$C = C_o + \frac{\lambda}{2n}\sum_w w^2 \quad (2)$$

$$w \rightarrow w' = w\left(1 - \frac{\eta\lambda}{n}\right) - \eta\frac{\partial C_o}{\partial w} \quad (3)$$

where c_o is the chosen loss distance function, w are the weight parameters across the network, n is the number of weight parameters and λ is the $L2$ regularisation hyper-parameter used to tweak the regularisation amount. Eq. 3 is the derived weight change $w \rightarrow w'$ during back-propagation, influenced by η the learning rate and reduced by the regularisation term. Once trained, the decode layer is discarded whilst the encoded activation produced in the encode layer will be used as the input for the next single layer SAE.

C. LSTM

LSTM is an extension of RNN, a class of neural network in which each node contains the addition of a weighted, time delayed, unidirectional feedback link that loops back in as input vectors forming a recurrent connection [16]. By delaying the recurrent signal, each node is able to store past information and thus be referred to during future predictions. This short term memory is especially effective in time series data which would otherwise lose time dimensionality. However, RNNs in practice are limited to only referring back several steps due to the vanishing and exploding gradient problem. As a result, RNNs would be unable to capture long term trends within the dataset. LSTMs provide a solution to said problem through the use of forget and update gates to modify memory cell state. LSTM units comprise of four components: the forget, input, update and output gates as shown in Fig. 2. The update gate (Eq. 4) dictates update of the previous cell state C_{t-1}. The forget gate (Eq. 5) controls memory loss, f_t. Throughout this section, h_{t-1} is the activation of the previous time-step, X_t is the input vector from the previous network layer, while W & b denotes weight and bias respectively of the corresponding gate.

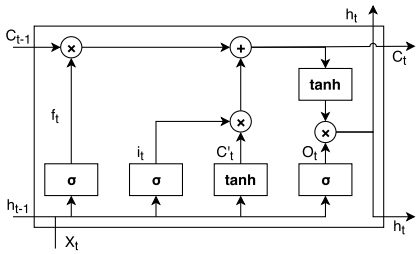

Fig. 2. Model of LSTM Cell.

$$C_t = f_t C_{t-1} + i_t C'_t \quad (4)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f) \quad (5)$$

The input gate encodes, C'_t (Eq. 6) and selectively incorporates, i_t (Eq. 7) input from $[h_{t-1}, X_t]$ into the new cell state.

$$C'_t = tanh(W_c \cdot [h_{t-1}, X_t] + b_c) \quad (6)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i) \quad (7)$$

The output gate calculates the activation signal h_t by rescaling the updated cell state using (Eq. 8) before being filtered by O_t, calculated using (9).

$$h_t = O_t \cdot tanh(C_t) \quad (8)$$

$$O_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o) \quad (9)$$

As shown in Fig. 3, the proposed framework contains a ten LSTM unit hidden layer receiving encoded input from the aforementioned AE. This layer uses a *tanh* activation function with a delay of four time-steps. The final output layer contains a single LSTM unit with a non-linear activation function which produces a normalised prediction of the next day market closing price.

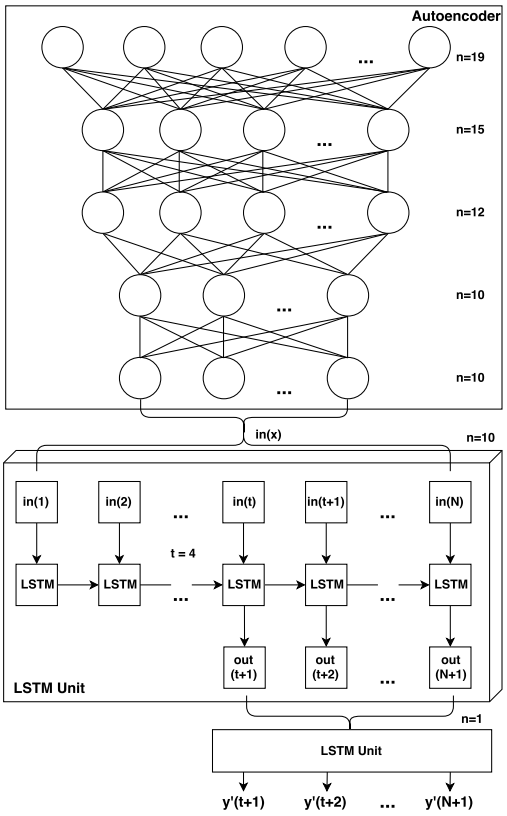

Fig. 3. Diagram of the proposed neural network architecture.

III. Experiment and Discussion

A. Dataset

The dataset provided by [12], comprises of six market indices from countries in varying states of development in order to demonstrate robustness from the varying market conditions. The market indices provided are *S&P 500* and *DJIA* from New York, *CSI 300* from China, *Nifty 50* from

TABLE I
DESCRIPTION OF DATASET VARIABLES

Daily Trading Data	
Open/Close Price	Nominal daily open/close price
High/Low Price	Nominal daily highest/lowest price
Trading Volume	Daily trading volume
Technical Indicators	
MACD	Moving average convergence divergence
CCI	Commodity channel index
ATR	Average true range
BOLL	Bollinger band
EMA20	20 day exponential moving average
MA5/MA10	5/10 day moving average
MTM6/MTM12	6/12 month momentum
ROC	Price rate of change
SMI	Stochastic Momentum Index
WVAD	Williams' variable accumulation/distribution
Economic Factors	
Exchange Rate	US dollar index
Interest Rate	Interbank offered interest rate

India, *Hang Seng* from Hong Kong and *Nikkei 225* from Tokyo. For each market, a selection of 19 daily variables separated into 3 sets of categories the first of which being the historical trading data of each index representing basic trade information. The second set of categories are the technical indicators which show a statistical evaluation of each index. The third set provides a broad economic overview of the region. Individual variables and their definition are shown in Table I. The time period available within the dataset is from 2008-07-01 to 2016-09-30 giving just over 2000 time-steps.

B. Training

In order to facilitate a continuous prediction cycle over the dataset, subsets of data were extracted using sliding window style with a length of two and a half years and a step size of three months. Each subset were then further split into a training set for the first two years, a validation set for the following three months and a test set for the final three months. The resulting combined test set for each market index spans a continuous period of six years or 24 quarterly test runs. The SAE was trained on the first test run with the *AdaDelta* optimiser using an MAE loss function with L2 regularisation. The initial learning rate and batch size chosen through experimentation were 1.0 and 100 respectively. Training was done until the minimum MAE delta between time-steps was 0.0001 where the model providing the lowest validation loss was chosen as the candidate for testing. The combined encode and LSTM layers are then retrained with the Adam optimiser [13] using an initial learning rate of 0.001, using an MAE loss function. Again, training was performed until a minimum loss delta of $1e^{-5}$ and the best model, based upon validation loss, chosen for testing. Subsequent test runs are fine-tuned on the same model with a learning rate of $1e^{-4}$.

C. Evaluation

This section presents the evaluation metrics used to judge framework performance. Notation wise, N is the total number of time-steps within the testing duration, y_t is the true closing price at time-step t while y'_t is the predicted closing price at t.

1) MAPE: Mean Absolute Percentage Error measures the relative percentage size of error and as such a smaller value indicates better performance. The scale independence of MAPE meant it was chosen over more traditional evaluation metrics such as RMSE or MAE allowing for direct comparison against various market indices.

$$MAPE = \frac{\sum_{t=1}^{N} |\frac{y_t - y'_t}{y_t}|}{N} \tag{10}$$

2) PCC: Pearson's Correlation Coefficient measures the linear correlation between two variables. Values range between +1 and -1, where +1 is a positive linear correlation, -1 is an inverse linear correlation and 0 is no linear correlation.

$$PCC = \frac{\sum_{t=1}^{N}(y_t - \bar{y_t})(y'_t - \bar{y'_t})}{\sqrt{\sum_{t=1}^{N}(y_t - \bar{y_t})^2 (y'_t - \bar{y'_t})^2}} \tag{11}$$

3) UC: Uncertainty Coefficient measures the degree of association between two variables and performs similarly to that of MAPE while also emphasising large relative errors. As such, a smaller value indicates better performance.

$$UC = \frac{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - y'_t)^2}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t)^2} + \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y'_t)^2}} \tag{12}$$

4) PP: In order to evaluate the Profitability Performance of the proposed framework, a trading strategy is developed which is able to leverage the predicted results into producing a profit. Following the strategy proposed by *Yao et. al.* [17], trading is based upon the relative difference between predicted and current levels. As such the strategy is as shown in (13). Simply put, buy when an increase in price is predicted and sell when a decrease is predicted.

$$\text{if } (y'_{t+1} - y_t) > 0, \text{then } buy \text{ else } sell \tag{13}$$

As such, the profit earned through this trading strategy can be dictated by Eq. 14 following *Bao et. al.* [12]. At the suggestion of *Yao et. al.* [17], the addition of transaction costs T of 0.01% was also introduced.

$$P = \sum_{t-1} \begin{cases} (y_{t+1} - y_t - T(y_t + y_{t+1}))/y_t, \text{ for } buy \\ (y_t - y_{t+1} - T(y_t + y_{t+1}))/y_t, \text{ for } sell \end{cases} \tag{14}$$

Profitability performance will be compared against two classical benchmarks for simulated profit. The first being the buy and hold method, in which an index is bought on the first day of testing and sold on the last day, providing a baseline trend to determine framework profitability and non-profitability.

TABLE II
COMPARISON OF PREDICTIVE ACCURACY

CSI 300			
	MAPE	PCC	UC
Proposed	**0.009±0.009**	**0.994±0.003**	0.014±0.009
WSAEs-LSTM [12]	0.019±0.007	0.944±0.041	**0.013±0.005**
LSTM [12]	0.056±0.015	0.617±0.239	0.035±0.009
Nifty 50			
	MAPE	PCC	UC
Proposed	**0.005±0.001**	**0.999±0.010**	**0.003±0.001**
WSAEs-LSTM [12]	0.019±0.002	0.841±0.110	0.013±0.002
LSTM [12]	0.034±0.006	0.569±0.301	0.023±0.004
Hang Seng			
	MAPE	PCC	UC
Proposed	**0.006±0.002**	**0.996±0.003**	**0.004±0.002**
WSAEs-LSTM [12]	0.015±0.004	0.931±0.022	0.011±0.004
LSTM [12]	0.024±0.002	0.847±0.076	0.015±0.001
Nikkei 225			
	MAPE	PCC	UC
Proposed	**0.008±0.004**	**0.999±0.005**	**0.007±0.003**
WSAEs-LSTM [12]	0.017±0.002	0.937±0.043	0.011±0.001
LSTM [12]	0.031±0.003	0.814±0.137	0.020±0.002
S&P 500			
	MAPE	PCC	UC
Proposed	**0.004±0.001**	**0.999±0.005**	**0.003±0.001**
WSAEs-LSTM [12]	0.011±0.002	0.946±0.036	0.007±0.002
LSTM [12]	0.017±0.004	0.875±0.085	0.012±0.003
DJIA			
	MAPE	PCC	UC
Proposed	**0.003±0.002**	**0.999±0.018**	**0.003±0.002**
WSAEs-LSTM [12]	0.011±0.003	0.949±0.022	0.007±0.002
LSTM [12]	0.021±0.003	0.809±0.083	0.013±0.002

- Benchmark 1 is calculated using Eq. 15 where m is the testing duration in months.

$$P = \left(\frac{y_N}{y_0}\right)^{\frac{12}{m}} - 1 \tag{15}$$

- Benchmark 2 is the trend following method in which the decision to buy, sell or hold is determined by the trend set by the previous three price indices as shown in Eq. 16 while profit earned is calculated using Eq. 14.

$$\begin{aligned} &\text{if } (y_{t-1} > y_{t-2}) \cap (y_{t-2} > y_{t-3}), \text{ then } buy \\ &\text{if } (y_{t-1} < y_{t-2}) \cap (y_{t-2} < y_{t-3}), \text{ then } sell \end{aligned} \tag{16}$$

D. Discussion

Evaluation of results are shown in Fig. 4, Table II and Table III. All values in Table II are mean performance with standard deviation over several experimental runs. Best performance in each category has been highlighted. As seen in Table II and Table III, our proposed framework outperforms the original model proposed by *Bao. et. al*. [12] across almost every category over all six stock markets with significantly lower average and deviation. While our model suffers from underestimating large low frequency increases, as evident in the *CSI 300* index subplot of Fig. 4 and in the limited improvement of the uncertainty coefficient, correlation between prediction and truth remain exceedingly high. This translates especially well to a high profitability performance with an average profit increase of 91% across the six datasets. In regards to performance across each dataset, the Hang Sent index provided the best predictive accuracy while *CSI 300* provided the worst. However, this does not follow profitability performance metrics, with the *Nikkei 225* index providing the most profit and *DJIA* the least. Both benchmark models also provide a lack of correlation between profitability of the model and the market capacity for profit. This is due to classical prediction evaluation metrics not translating well into the evaluation of market trend prediction as proposed by *Yao et. al.* [17]. Fig. 4 also shows that the large errors are mainly come from time period 1000 to 1200 in both *CSI 300* and *Hang Seng*, where the predictions are deviated from the truth values when large and high frequent pricing are fluctuating. An reasonable explanation is that such pricing fluctuation patterns have never occurred in previous training periods, and the model itself is not able to make correct prediction. We also observed that the predicted signals trend to be smoothed in its local time window, which may not suitable for the scenarios where high-frequency trading strategies are involved.

IV. CONCLUSION

This paper improves upon a previous framework while also providing the novel use of adapting SAEs to pre-train deep LSTM networks for use in financial time series predictions. It provides a highly effective prediction model of various stock markets across the globe of varying market conditions. Through the use of more advanced optimisation methods and an improved network structure, the proposed framework outperformed the original by significant margins in both profitability and predictive accuracy.

REFERENCES

[1] E. F. Fama, L. Fisher, M. C. Jensen, and R. Roll, "The Adjustment of Stock Prices to New Information," *Int. Econ. Rev.*, vol. 10, pp. 1–21, 1969.

[2] B. Krollner, B. Vanstone, and G. Finnie, "Financial time series forecasting with machine learning techniques: A survey," in *Eur. Symp. on Artificial Neural Networks: Computational and Mach. Learning*, 2010.

[3] A. Sharma, D. Bhuriya, and U. Singh, "Survey of stock market prediction using machine learning approach," in *Int. Conf. of Electron., Commun. and Aerospace Technol.*, vol. 2, 2017, pp. 506–509.

[4] R. Choudhry and K. Garg, "A Hybrid Machine Learning System for Stock Market Forecasting," *World Academy of Sci., Eng. and Technol.*, vol. 2, no. 15, pp. 315–318, 2008.

[5] R. Iacomin, "Stock Market Prediction," in *Int. Conf. on System Theory, Control and Computing*, 2015, pp. 200–205.

[6] S.-S. Luo, Y. Weng, W.-W. Wang, and W.-X. Hong, "L1-regularized logistic regression for event-driven stock market prediction," in *Int. Conf. on Comput. Sci. and Edu.*, August 2017, pp. 536–541.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] L. Deng, "Three classes of deep learning architectures and their applications: a tutorial survey," *APSIPA transactions on signal and information processing*, 2012.

[9] J. Heaton, N. Polson, and J. Witte, "Deep learning in finance," *arXiv preprint arXiv:1602.06561*, 2016.

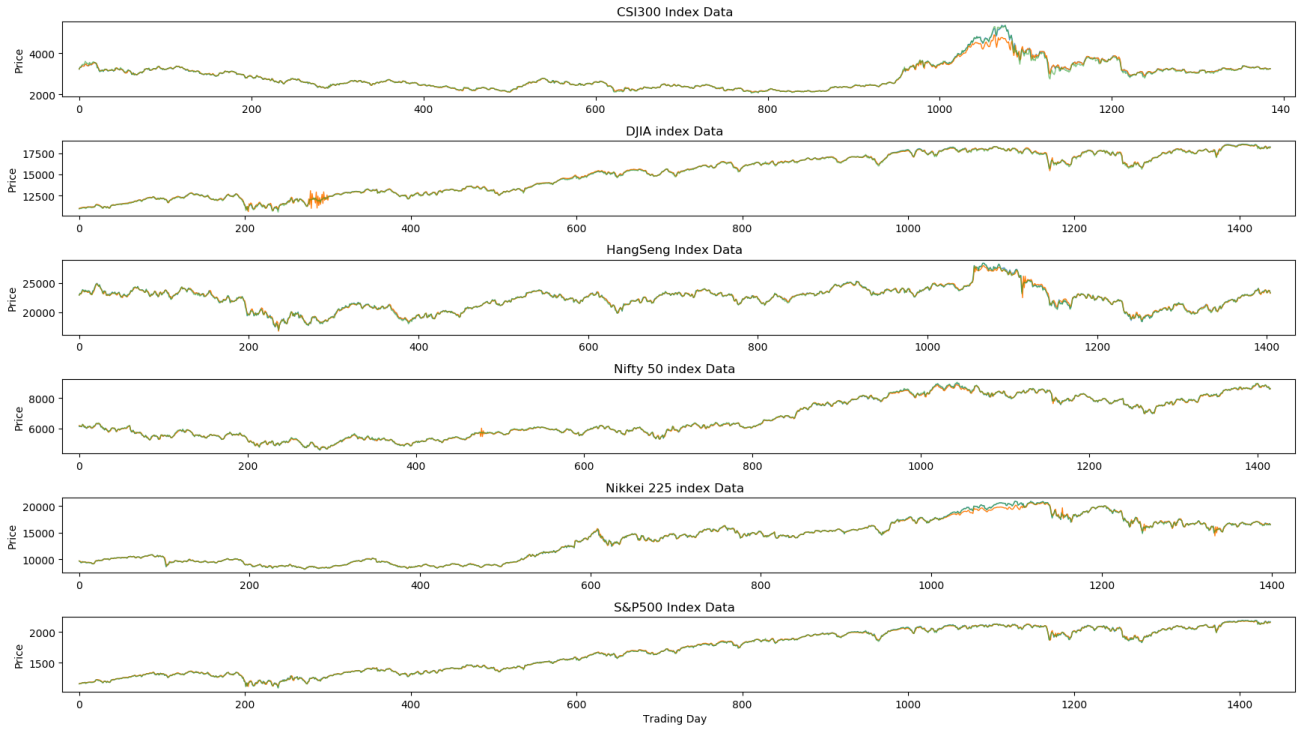

Fig. 4. Graph of true daily closing price and predicted closing price. Each subgraph displays the six year test results of each market index. Blue indicates the true closing price, orange shows predicted based upon the proposed framework and green, the predictions given by [12].

TABLE III
COMPARISON OF PROFITABILITY PERFORMANCE

	Y1	Y2	Y3	Y4	Y5	Y6	Y1	Y2	Y3	Y4	Y5	Y6
	CSI 300						**Nifty 50**					
Proposed	**191.26**	**187.91**	**176.52**	**161.34**	**223.79**	**193.48**	**208.58**	**179.33**	**169.21**	**128.46**	**128.51**	**145.31**
WSAEs-LSTM [12]	46.43	59.58	65.69	71.90	63.95	70.61	56.91	51.71	66.77	37.11	31.88	28.13
Benchmark 1	-19.63	-12.60	3.07	0.29	35.62	3.43	-23.56	13.93	-1.51	30.89	-2.79	5.48
Benchmark 2	-14.12	-17.22	-1.82	5.09	27.64	-40.26	6.98	-14.99	1.04	-5.21	13.54	-15.76
	Hang Seng						**Nikkei 225**					
Proposed	**192.38**	**197.52**	**144.18**	**122.75**	**132.00**	**171.79**	**196.18**	**177.24**	**250.95**	**179.34**	**106.21**	**247.58**
WSAEs-LSTM [12]	75.84	81.89	54.69	49.72	64.73	59.95	53.46	37.85	81.03	48.83	59.42	76.03
Benchmark 1	-27.32	16.34	7.47	-2.82	-9.26	9.46	-19.37	1.57	49.34	9.79	6.37	-6.73
Benchmark 2	5.95	13.45	12.39	9.20	-16.23	-29.78	-39.42	-2.78	-19.51	-28.69	13.49	-22.21
	S&P 500						**DJIA**					
Proposed	**150.01**	**161.87**	**102.17**	**82.53**	**123.55**	**126.71**	**78.87**	**69.02**	**52.74**	**51.9 0**	68.74	71.90
WSAEs-LSTM [12]	71.32	48.35	39.24	9.94	61.19	45.95	76.74	55.43	41.70	47.03	**82.50**	**80.52**
Benchmark 1	-12.27	22.76	13.21	13.75	-5.36	9.82	-7.86	19.07	9.73	9.62	-7.14	9.36
Benchmark 2	6.33	0.22	-18.40	-8.46	-21.49	-22.97	15.90	18.93	16.37	18.43	26.25	17.81

[10] M. Billah, S. Waheed, and A. Hanifa, "Stock Market Prediction Using an Improved Training Algorithm of Neural Network," *Comput. & Telecommunication Eng.*, vol. 8, no. December, pp. 8–10, 2016.

[11] W. Chen, Y. Zhang, C. K. Yeo, C. T. Lau, and B. S. Lee, "Stock market prediction using neural network through news on online social networks," in *Int. Conf. on Smart Cities*, 2017, pp. 1–6.

[12] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS One*, vol. 12, no. 7, 2017.

[13] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Int. Conf. on Learning Representations*, pp. 1–15, 2014.

[14] D. L. Donoho and I. M. Johnstone, "Adapting to Unknown Smoothness via Wavelet Shrinkage," *J. of the Amer. Statistical Assoc.*, vol. 90, no. 432, 1995.

[15] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," *Advances in Neural Inform. Process. Syst.*, vol. 19, no. 1, p. 153, 2007.

[16] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *Trans. on Neural Networks and Learning Syst.*, 2017.

[17] J. Yao, C. L. Tan, and H.-L. Poh, "Neural Networks for Technical Analysis: a Study on Klci," *Int. J. of Theoretical and Appl. Finance*, vol. 02, no. 02, pp. 221–241, 1999.