A LSTM-based Quantum Finance Predicting System

Frank (1730026119), Zeke (1730026129), Sarah (1730026139), Kevin (1730006028)

***Abstract*—With the development of artificial intelligence, using AI to predict the stock and financial market is bound to be a more and more popular trend in the industry. Among different sorts of AI techniques, Artificial Neural Network is widely implemented in practice, because of its flexibility, which theoretically enabling it to simulate almost all functions in the world. There is a type of NN that named RNN, whose structure is suitable to be used to process time series data, exactly one of the most important data we need to model the financial market. Except for the NN model itself, there are several factors that may affect the performance of the prediction, for example, the activation function. Thus, we attached great importance to the study of activation function, and the comparison of the predicted results that given by different NNs. In this project, several activation functions, namely Sigmoid, Tanh, ReLU, PReLU, and ELU, are firstly been analysed and compared. Next, the analysis of the structure of different NNs give the result that DeepLSTM would theoretically have the best performance. After implementing the experiment, we proved that DeepLSTM has the best performance on predicting the close price (among FFBPN, DNN, LSTM, MultiBiLSTM), which reaching 3.86×10^{-6} for training set and 2.66×10^{-6} for test set, respectively.**

I. INTRODUCTION

The financial and stock market is long been fluctuating and containing a lot of noises, which making it hard to precisely forecast the market trend. Different research teams have contributed a lot of efforts on this problem, like using type-1 fuzzy sets to do secondary factor induction. However, the performance for it to capture the world's uncertainly is still limited. There is another tool, the ANN, which is already playing a key role in solving real world problems with data. RNN is a kind of ANN that can recursively process the output of one layer with a certain weight and put it back to the input to be a feedback, in which way can improve the prediction performance. Despite, drawback is still inevitable. According to Hochreiter, using traditional RNN to forecast the time series will encounter problems during the process of gradient descent - it is hard to find the global minimum for a dataset that is covering a long period of time, which resulting in a bad convergence of the model when handling long time series data. Fortunately, several of Hochreiter and Eunsuk's researches also proved a kind of RNN that can solve this problem, which we called LSTM (Long Short-term Memory). The LSTM perfectly solve the drawback of the traditional RNN by introducing several components, namely the forget gate, input gate, and output gate, which enabling it to recursively try to converge in different local minimum, so as to eventually figure out the global minimum with comparison. Therefore, LSTM has been becoming the most popular time series modelling tool in recent years. Moreover, the ANN itself is not the only factor that affecting the performance of the forecasting. Activation function is another vital factor that may contribute to different forecast results. Therefore, in this paper, we are focusing on the ANN and activation function at the same time. We are going to implement an experiment and study the performance of several sorts of ANNs, namely the MLP, DNN(FFBPN), LSTM, and then try to optimise the model by using MultiBiLSTM and DeepLSTM to do closing price prediction. Meanwhile, we are also going to analyse and compare the features of different activation functions, such as Sigmoid, Tanh, ReLU, PReLU and ELU, in order to understand how these functions will affect the model.

II. LITERATURE REVIEW

RNN, which can recursively give one or multiple weighted feedbacks back into the input, provides an approach to do neuro-evolution (Mehreen et al., 2014). Mehreen introduced a RCGPANN (Recurrent Cartesian Genetic Programming ANN) that hoping to use CGP method to find out a better RNN generation, in order to solve the drawback that an RNN maybe converge to a local minimum. The training process is started with 500 days of data from US currency, which are used to train ten RNNs and five seeds are used for each NN to ensure the feature of each single seed will not affect the final solution. The sigmoid function is chosen to be the activation function while each node consists of five nodes. The mutation rate is then set to be 10%, according to another research performed by (Chen et al., 2008) on the hybrid forecasting model for foreign exchange rate based on a Multi-Neuro Network that indicates a 10% of mutation rate is better in comparison to others. Next, 10 consecutive currency value is used as input, while each neuron will have 5 inputs, in which way enabling the 5 inputs can either be the input of the preceding neurons or the input of the system. The CGP is used to evolve the suitable RNN, with the method of $1 + \lambda$ evolutionary strategy, where λ is noted as 9. To compare and select the generations, MAPE (Mean Absolute Percentage Error) and fitness, which respectively denoted as:

$$MAPE = \frac{1}{N}\sum i = 1toN\left(\frac{|L_F - L_A|}{L_A}\right) \times 100$$

Fitness = 100 – MAPE

are used as the performance metric criteria, and all resulting offspring are going to be assessed. The one fittest offspring will be selected according to the above metrics, which will later be used for up gradation to the next generation. In Mehreen's experiment, three values of feedbacks are tested, namely single feedback, five feedbacks, and ten feedbacks. All these RNNs are trained with 1000 days of data and required predict the eleventh day's data for five currencies (Yen, NZD, CAD, KRW, IDR) on the basis of 10 days of histories. According to the result, for single feedback and five feedbacks, the RNN that with 500 nodes provided the best performance with the accuracy of all five currencies are above 98%, and the highest accuracy is in five feedback predicting IDR that reaching 98.872%. However, there is a slightly different on the result of ten feedbacks, whose performance is better in the situation with 400 nodes, and the prediction result of IDR is also 98.872%.

Similarly, (Bao W, Yue J, and Rao Y, 2017) think that a deep nonlinear topology should be applied to do time series prediction. This team introduced an approach that combining three techniques, namely WT (Wavelet Transforms), SAEs (stacked autoencoders) and LSTM, in order to perform the forecasting of stock closing price. In this research, the financial time series data are processed by multi-resolution discrete wavelet transformation after the input. This step is doing decomposition, so as to denoise the data. Wavelet transformation is suitable to be used to extract features based on time series, because of the characteristic that wavelet only have waves in a certain part of time series, which we called locality, unlike sine that has continuous wave.

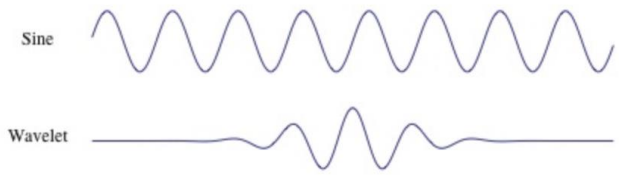

Fig. 1. The Different between Sine and Wavelet. The wavelet's wave has locality in reference of time.

This characteristic enabled WT to find out more precise information about the time, except for frequency information. The denoised time series data are then be processed by the autoencoder. A Single layer AE is a three-layer NN, whose layers are used to generate deep features for reconstruction of the data with different weight that it learned. Training of SAE aims at minimising the error between input vectors and the reconstructed vectors. The forward propagation can be divided into two steps: 1) Mapping the input layer to the hidden layer, which shows by the rectangular part of the figure. 2)Perform reconstruction by mapping the hidden layer to the reconstruction layer. The two steps have the corresponding formula:

$$a(x) = f(W_1 x + b_1)$$

$$x' = f(W_2 a(x) + b_2)$$

where $x \in R^k$ and $x' \in R^k$ are respectively the input vector and the reconstructed vector. $a(x)$ is the hidden vector generated by the single layer AE. W_1 and W_2 are respectively the weight of the hidden layer and the reconstruction layer.

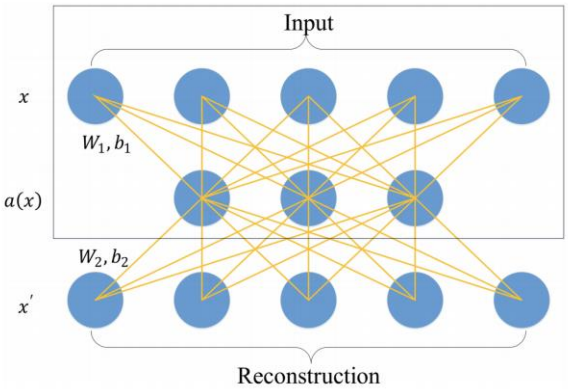

Fig. 2. Single layer Autoencoder. Mapping input vector from input layer to the hidden layer, and then map the vector from hidden layer to reconstruction layer. (Bao Y, 2017)

In this case, the SAE (or Stacked AE) is used, so as to do a more sophisticated pre-training that can again extract features from the data and get the initial value of the weight matrix. The encoded data are still containing the original features. During the decode process, the extracted features will be kept, while other noises will be gone because the AE's encoding is a lossy compression. The structure of the SAE is shown below:

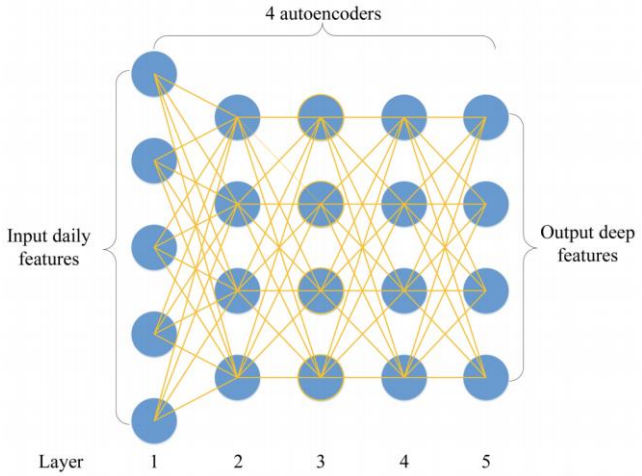

Fig. 3. Five-layer SAE that trained by four AEs. (Bao Y, 2017)

In the last step, the data pre-processed by WT and SAEs are

then be put into the LSTM according to time. The input gate, forget gate, and the output gate are going to work together to process the data recursively, so as to compare and find the best value (or the minimum value) of MAPE, which means finding the global minimum. The experiment is implemented in three types of market, the developing market (pattern not that stable), the relatively developed market (relatively stable), and the developed market (most stable among the three). It is not out of expectation that the performance of predicting developed market is the best, as the market's pattern is more stable. However, the result also shown some vertical comparison that compared several ANNs, namely RNN, LSTM, WLSTM and WSAEs-LSTM. The result shows an LSTM is performing better than the traditional RNN, for example, in the prediction of CSI300, RNN and LSTM have 0.066 and 0.056 of MAPE, respectively. And among all tested ANNs, WSAEs-LSTM provided the best result, around 0.019 for developing market, 0.015 for relatively developed market (0.015 for Hang Seng Index and 0.017 for Nikkel 225 Index), and 0.011 for developed market. Thus, these results show the potential of LSTM to be further optimised and deployed to solve the time series forecasting problem.

A team from PKU has done a similar job, using a four-layer NN to model the customers' buying decision. The first layer is introduced as the input layer. The user's feature vector f_{jt} will be extracted from each click c_t^u performed user u on item i. There are two kinds of vectors that can be categorised into: I) numerical feature vector $x_t^n \in \mathbb{R}^N$, where N denotes the total number of numerical features. II) embedding feature vectors $(x_{t,1}^e, x_{t,2}^e, \ldots, x_{t,E}^e)$, where $x_{t,c}^e \in \mathbb{R}^{V_c}$ is a one-hot-vector representation of the c^{th} embedding feature. Then the second layer will receive the result from the first layer, concatenate them, and denote them as e_t. The third layer contains one or more LSTM layers in a cascade structure. The inner memory state in LSTM's hidden layer is updated at each time step, whose output would be represented as $h_t = f(e_t, h_{t-1}, c_{t-1}, W_{RNN})$, where f is a function unit like a LSTM layer, and W_{RNN} are model parameters. The last layer is the output layer. Qiaolin and Zhifang regards this task as classification problem on sequential data. So, for the prediction of the stage ϕ_k, the probability would be:

$$P(\phi_k \mid t, c_1^u, c_2^u, \ldots c_t^u) = g_{k,t} = \sigma(V_k h_t + b_k)$$

where $\sigma(x)$ is sigmoid function that equals to $\frac{1}{1+e^{-x}}$. Finally, a function that can predict the probability of 'direct buy', 'will buy' or others will be reached, by introducing SoftMax function, which has a good feature for derivation, and cross-entropy cost function (or NLL). Two functions are listed below:

The probability of class ω_i:

$$P(\omega_i \mid t, c_1^u, c_2^u, \ldots c_t^u) = y_{i,t} = \frac{e^{(W_s h_t + b_s)_i}}{\sum_{j \in \Omega} e^{(W_s h_t + b_s)_j}}$$

The objective function:

$$\mathscr{L}_{buy}(z) = -\frac{1}{M} \sum_{t=0}^{M} \sum_{t=1}^{|\Omega|} z_{i,t} \ln(y_{i,t}) + \frac{\lambda}{2} ||\Theta||^2$$

According to the experiment, this model has slightly better performance, having 0.8589 of AUC in comparison of the 0.778, 0.842 and 0.843 in Tobias and Matthias's research, which indicates by using more layers of LSTM and embedding more feature for pre-processing, it would provide a relatively better result.

III. Proposed AI-Based Quantum Finance System

A. *Some traditional ANN structures*

As what is mentioned, FFBPN, DNN and LSTM are used in this experiment to predict the opening price, in order to study the performance of the relatively traditional ANNs.

FFBPN (Feed Forward Back Propagation Network)
The FFBPN is mentioned in

Training Algorithm (4.3) – FFBPN (Fausett, 1994; Patterson, 1996)

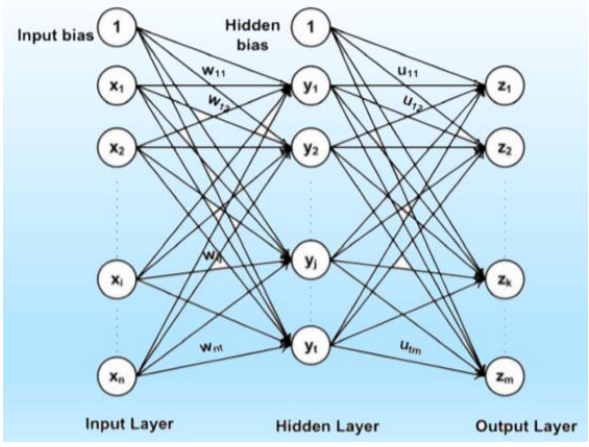

Fig. 4. FFBPN

FFBPN is a three-layer neural network. For the i^{th} neuron of the hidden layer has:

$$yi = fy(SUM(wji * xj + wb0 * input\ bias))$$

Similarly, for the output layer:

$$zi = fz(SUM(uji * yj + wb1 * hidden\ bias)$$

Where $fy\ fz$ are activation function (e.g. Sigmoid function), which enable the network to learn the non-linear target function.
FFBPN use backward propagation to learn parameters.

Training Algorithm (4.3) – FFBPN (Fausett, 1994; Patterson, 1996)

Step 1: Network weight initialization.
Set all network weights w_{ij}, u_{jk} to a small random number between 0 and 1.

Step 2: While error ≥ threshold value, do the following:

Step 2.1: For each training pair (**x**, **z**) do Steps 2.1.1 to 2.1.6.

Feedforward Procedure

Step 2.1.1 Calculate the input state of each hidden node:

$$y_{in_j} = \sum_{i=0}^{n} x_i w_{ij}$$ where x_0 is the input bias

Step 2.1.2 Calculate the activation value for the hidden node:

$$y_j = f_y(y_{in_j})$$ where $f_y(\,)$ is the activation function

Step 2.1.3 Calculate the input state of each output node:

$$z_{in_k} = \sum_{j=0}^{t} y_j u_{jk}$$ where y_0 is the hidden bias

Step 2.1.4 Calculate the activation value for the output node:

$$z_k = f_z(z_{in_k})$$ where $f_z(\,)$ is the activation function

Backpropagation Procedure

Step 2.1.5 For each output node:

(a) Calculate the error with the target value

$$\zeta_k = (z'_k - z_k) f'_z(z_{in_k})$$ where $f'(\,)$ is df_z/dz

(b) Calculate the correction errors

$$\Delta u_{jk} = \alpha \zeta_k y_j$$ where α is the learning rate

Step 2.1.5 For each hidden node:

(a) Calculate the accumulated errors in the hidden node

$$\lambda_{in_j} = \sum_{k=1}^{m} \zeta_k u_{jk}$$

(b) Calculate the correction errors in hidden node

$$\lambda_j = \lambda_{in_j} f'_y(y_{in_k})$$ where $f'(\,)$ is df_y/dy

(c) Calculate the weight adjustments

$$\Delta w_{ij} = \alpha \lambda_j x_i$$ where $f'(\,)$ is df_y/dy

Step 2.1.6 Update all weights for the two layers (simultaneously):

$$w_{ij}(new) = w_{ij}(old) + \square w_{ij}$$

$$u_{jk}(new) = u_{jk}(old) + \square u_{jk}$$

Step 2.2: Check the stopping criteria.

FFBPN is already able to converge to a relatively simpler non-linear function. However, its layer amount is limited. Thus, the ability of FFBPN to show non-linear features is still needs to be improved.

DNN (Deep Neural Network)

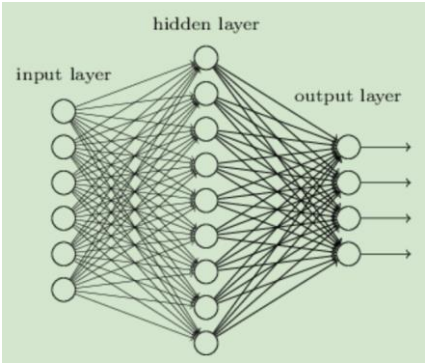

Fig. 5. The structure of DNN

DNN added a hidden layer based on the FFBPN. In common situation, DNN regards the forward propagation neural network that has more than two hidden layers. When the hidden layer increases, the ability of showing non-linear features is stronger. We use gradient descent in DNN to renew the network's parameters. The Chain rule of derivation is used in backward propagation.

$$\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx}$$

When the input value x is given, we can use the loss function gradient in x of the network, and let the gradient of wij in given x is Δwij, then the gradient descent algorithm can be described as:

$$wij = wij - \eta \Delta wij$$

where η is learning rate

RNN (Recurrent Neural Network)

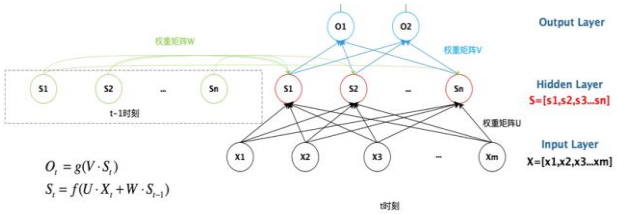

Fig. 6. Structure of RNN

The main difference between RNN and ANN is that the weight of RNN is shared. The input of RNN is passing through the network in a specific sequence, but not simultaneously. Therefore, RNN is more powerful on time-series modelling and solve the problem of the input that has an unfixed length.

When the RNN is training, a vanishing gradient and exploding gradient problem always happen.

$$\frac{\partial J}{\partial U} = \sum_{k=1}^{t} \frac{\partial J}{\partial O_t} \frac{\partial O_t}{\partial S_t} \frac{\partial S_t}{\partial S_k} \frac{\partial S_k}{\partial U},$$

Noted that:

$$\frac{\partial O_t}{\partial S_t} \frac{\partial S_t}{\partial S_k} = \prod_{i=k+1}^{t} \frac{\partial S_i}{\partial S_{i-1}} = \prod_{i=k+1}^{t} f'(W)$$

where f is activation function, when the derivation of activation function is continuously larger than 1 or smaller than 1, $\prod_{i=k+1}^{t} \frac{\partial S_i}{\partial S_{i-1}}$ would become very big or small, so as to causing the vanishing gradient and exploding gradient, which making RNN is hard to capture long series feature or the long series feature would have a serious loss.(sequence size > 3)

LSTM (Long Short-term Memory)

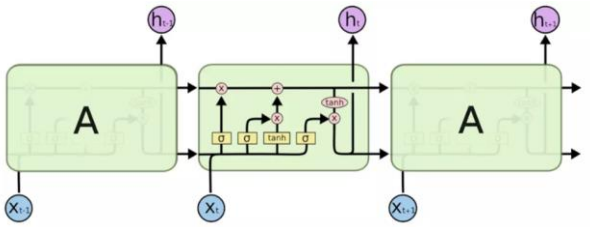

Fig. 7. Structure of LSTM

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

(1)

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

(2) (3)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(4)

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

(5) (6)

LSTM introduced gate, solve the vanishing gradient and exploding gradient problem of RNN, which enabling it to capture long distance dependent.

B. Traditional Model's Feature Analysis and Comparison

FFBPN: Because of the limitation of layers, the non-linear representation ability is not strong enough. Although the four-layer perceptron is proved to be closed to any non-linear functions, for complex function, a lot of neurons are needed to be set in the hidden layer of the perceptron, which usually cause the model to be over-fitting, losing the generalization ability. Thus, for the stock market that in a complex situation, FFBPN has poor prediction performance.

DNN: By adding the hidden layer, DNN is able to represent more complex non-linear functions. However, different from the logic of the RNN, DNN's input is consist of neurons with same weight, which means DNN is bound to have better non-linear representation ability, but it is still ignoring the time series factor. Under the circumstances that the training set have the same error rate, DNN is easier to be over-fitted than RNN, thereby the generalization ability is poorer.

LSTM: Although, Singe LSTM Cell is weight shared, and more suitable for series modelling, it still have serious problem. In the equation (1), (2), (3), (4), (5) above, from the input x to the output H, LSTM only perform three layers of non-linear transformation to x, which is similar to FFBPN that is not good for non-linear representation power.

According to the analysis, we can extract the advantage of these models, and avoid the disadvantage, so as to design a better model to forecast the market.

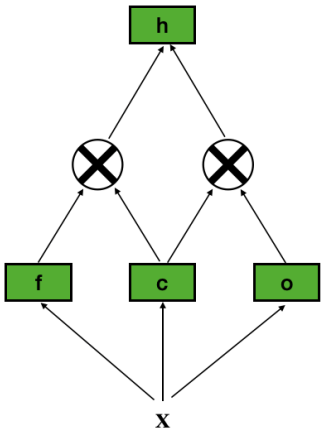

Fig. 8. LSTM

C Multi-layer Bidirectional LSTM

Multi-layer:
In order to solve the LSTM unit's poor in non-linear representation problem, we put the layers of LSTM together. According to the experiment, multi-layer LSTM has better non-linear approximator, and enabling each layer to learn different features.

Bidirectional:
Except for the consideration of historical information, future information should also be considered. In solving time series problem, bidirectional model is proved to be better in NLP. For example:

My <son> likes watermelon

If the model wants to understand the meaning of <son>, the thing that the model needs to consider is not only the <My>, but also the <watermelon>. Hence, the model may understand the semantic better. It is "my son likes watermelon here", but not "my cat like watermelon". The understanding to <son> in NLP is actually encoding the <son>. Similarly, in the stock's time series, if we want to encode the price of one day reasonably, we need to consider the information from the future.

In the paper of ELMO, its author used bidirectional LSTM, making the accuracy of the language model has significant improvement. We can use similar method to deal with the stock time series. There are three bidirectional layers are put together in our model. The last layer will do forward and backward output in consideration of the prediction value. The basic LSTM structure is used here, and the time series information is kept. Meanwhile, we regard the last single directional layer can highlight the differences of left-to-right and right-to-left, so as to show the time relationship of the 'next day'.

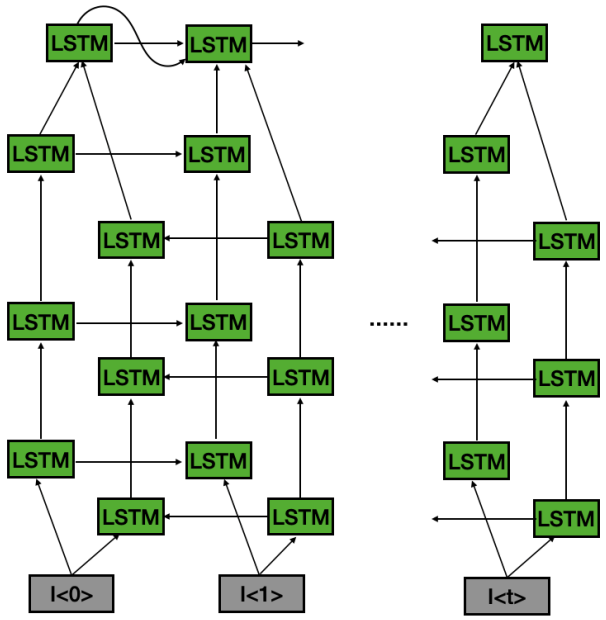

Fig. 9. Multi-layer Bidirectional LSTM

D Deep LSTM

Despite MBLSTM is more suitable to be used in the stock market prediction because of its structure, the converge speed of MBLSTM is slow, which means then training time cost is high. (The computational volume of LSTM is large, while the training speed of multilayer is slow). In order to solve this problem, we designed another model called DeepLSTM.

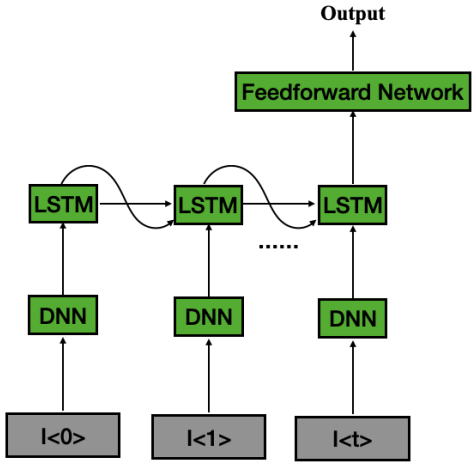

Fig. 10. Deep LSTM

After experimenting the effect of the single LSTM, we discovered that the performance of LSTM is far worse than DNN. When we actually deploy the model, use five day's OHCL as the input and feed to the model. From the perspective of value, the training set has two features:

1. The differences between the value of the OHCL is not that large.
2. If the non-coded data was put into the LSTM's gate, the value would be affected. Because there is only one layer of linear transformation from x to the gate, which indicates the smaller the value the higher possibility that the gate will tend to forget it (or the value=0), or otherwise, the opposite situation. Apparently, this is not reasonable.

We hope that before the input x reach the gate, we can encode it, making its value can represent the feature for xt in series, thereby we added an auto-encode layer. Different from the SAE in WASE-LSTM, SAE required the output layer and the hidden layer to be smaller than the input layer, which means mapping from high dimension to low dimension. By using DNN's auto-encoder, the encoded vector can have a higher dimension, and carry more useful information.

IV. Experiment Result

We used EURUSD with 2048 days of data to train our NN and divide the EURUSD data into training set and test set.

FFBPN

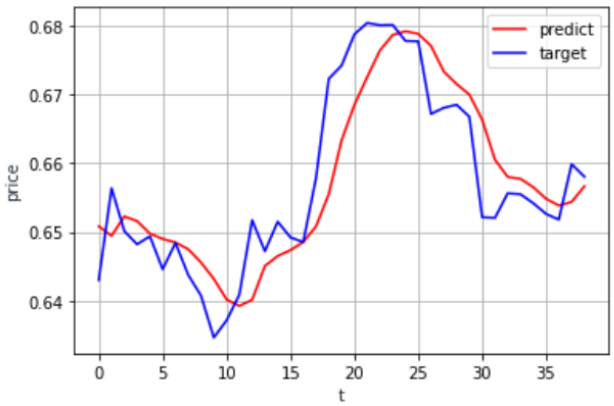

Fig. x. Result of FFBPN

FFBPN (for 1500 epoch)	
Training Set	Test Set
3.16×10^{-5}	3.84×10^{-5}

Table 1

DNN

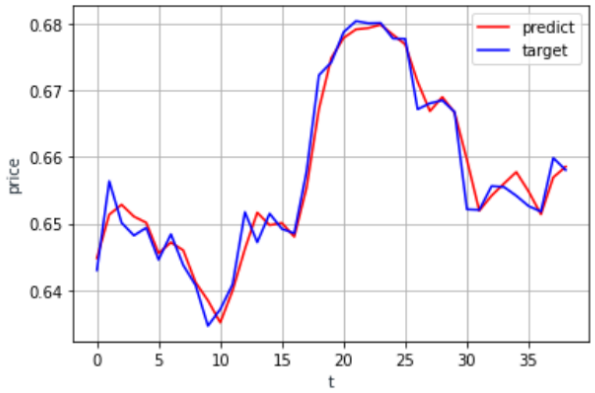

Fig. x. Result of DNN

DNN (for 1500 epoch)	
Training Set	Test Set
4.39×10^{-6}	6.27×10^{-6}

Table 2

LSTM

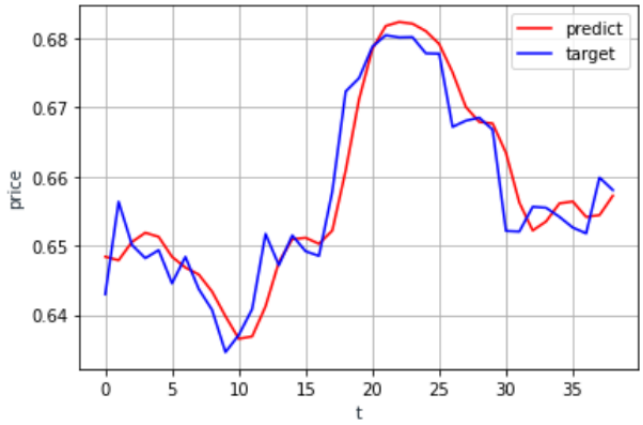

Fig. x. Result of LSTM

LSTM (for 1500 epoch)	
Training Set	Test Set
1.87×10^{-5}	1.96×10^{-5}

Table 3

DeepLSTM

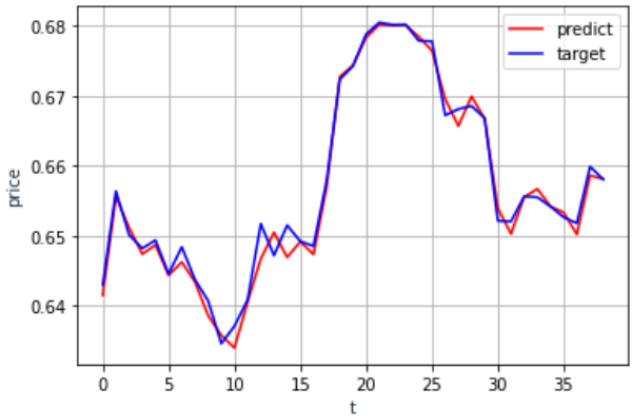

Fig. x. Result of DeepLSTM

DeepLSTM (for 1500 epoch)	
Training Set	Test Set
3.86×10^{-6}	2.66×10^{-6}

Table 4

MultiBiLSTM

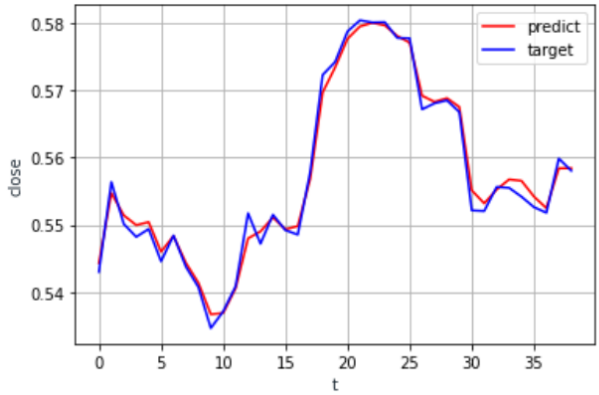

Fig. x. Result of MultiBiLSTM

MultiBiLSTM (for 1500 epoch)	
Training Set	Test Set
4.76×10^{-6}	4.41×10^{-6}

Table 5

By comparing the result, we can find that the DeepLSTM has the best performance that having an MSE in $\mathbf{3.86 \times 10^{-6}}$ for training set and $\mathbf{2.66 \times 10^{-6}}$ for test set.

Comparison of all NN above

NN	Training Set (MSE)	Test Set (MSE)
FFBPN	3.16×10^{-5}	3.84×10^{-5}
DNN	4.39×10^{-6}	6.27×10^{-6}
LSTM	1.87×10^{-5}	1.96×10^{-5}
DeepLSTM	$\mathbf{3.86 \times 10^{-6}}$	$\mathbf{2.66 \times 10^{-6}}$
MultiBiLSTM	4.76×10^{-6}	4.41×10^{-6}

Table 6

V. SUMMARY

To predict the financial and stock market is difficult even for nowadays industry, but it is quite important to provide information to make trading strategies. Therefore, find out a better model for training is vital. We have studied that different activation functions are affecting the performance of the neural network in different ways. Furthermore, according to our analysis of the models, LSTMs, or a type of RNN, is popular in nowadays industry, for solving changing and non-linear problems. Among the LSTMs we used in the experiment, DeepLSTM should have the best performance, while MultiBiLSM would be a little worse as the approach of feeding data is different. Our experiment also revealed the same result. The MAE is used to be the metrics of the performance of the model. DeepLSTM has the lowest MAE that only 3.86×10^{-6} for training set and 2.66×10^{-6} test set, respectively, while MultiBiLSTM has the relatively higher counterpart that 4.76×10^{-6} and 4.41×10^{-6}, respectively. Here we find the generalisation ability of LSTM is better than DNN, as DNN's MSE of training set is smaller than the test set. The predicted data provided can later be used in the trading process and be a reference for the trading system. It is quite an exciting thing to find a better solution to solve time series problem among all approaches. And this result indicates the potential for the LSTM to be further optimised, maybe with algorithms like CGP, or use multi-layer LSTM, which will do feature extraction before using LSTM to process. The LSTM should be no doubt one type of tool that worth studying, in order to find out an efficient way to predict the financial market with the time series data.

VI. BIBLIOGRAPHY

[1] P. J. ,. F. S. ,. Y. Z. ,. X. W. ,. a. Z. S. Qiaolin Xia, "Modeling Consumer Buying Decision for Recommendation Based on Multi-Task Deep Learning," CIKM'18, Torino, 2018.

[2] Mehreen Rehman, Gul Muhammad Khan, Sahibzada Ali Mahmud, "Foreign Currency Exchange Rates Prediction using CGP and Recurrent Neural Network," *IERI Procedia*, vol. 10, pp. 239-244, 2014.

[3] An-Pin Chen, Yu-Chia Hsu, Ko-Fei Hu, "A Hybrid Forecasting Model for Foreign Exchange Rate Based on a Multi-neural Network," *Fourth Int. Conf. on Natural Computation*, pp. 293-298, 2008.

[4] Wei Bao1, Jun Yue, Yulei Rao, "A deep learning framework for financial time series using stacked autoencoders and longshort term memory," *PLoS ONE*, vol. 12, p. e0180944, 2017.

[5] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark , Kenton Lee , Luke Zettlemoyer, "Deep contextualized word representations," Allen Institute for Artificial Intelligence, 2018.