Operation of Recurrent Neural Network Variant, Multivariable Linear Regression and Multivariable Non-Linear Regression

Ha G. Huy

STAT3013.N12.CTTT – EN

University of Information Technology

20521385@gm.uit.edu.vn

Dau Dinh Q. Anh

STAT3013.N12.CTTT – EN

University of Information Technology

20521059@gm.tui.edu.vn

Tran D. Duy

STAT3013.N12.CTTT – EN

University of Information Technology

20521248@gm.uit.edu.vn

Abstract – Deep learning algorithms have been implemented in several application domains and as such new algorithms have been developed to improve the performance. This paper presents a comparative study of three Recurrent Neural Networks including Long Short-Term Memory with Attention Mechanism, Bidirectional LSTM and Gated Recurrent Unit, on the accuracy aspect. The research revealed that RNN Models are in the most powerful Models for Time Series Data Prediction

I. INTRODUCTION

Recurrent Neural Network (RNN) models are the most effective sequence model and they have been applied in many applications with temporal or sequential data. For performance improvement and specific purposes, people have developed many variants of RNN algorithms including Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), Long Short-Term Memory with Attention Mechanism (A-LSTM) and Gated Recurrent Unit (GRU). RNN variants have been applied successfully in sequence application. For example, people have been used extensively in polyphonic music processing, namely speech signal processing [1], natural language processing [2], and sequence generation [3].

The main element leads to the success of them is the gating network signals that is used to control the current input values and old memory to update the current activation neural layer [4]. The set of weights in Gated RNN can be changed after each time step.

Multivariable Linear Regression (MLR) and Multivariable Non-linear Regression (MNLR) are the primitive algorithms presenting simple functions. Regression Models are mostly used for functional Data Analyst. By D. Nguyen et al in 2011, Linear Regression Model has been used for Author age prediction via text dataset. In 2008, KW. Lau et al, have implemented Non-Linear Regression Model for Local Prediction on complex time series.

In this paper, we focus on the RNN models, Multivariable Linear Regression (MLR) model and Multivariable Non-Linear Regression on a public dataset. Using the VNM dataset, we will predict the stock value after each day based on the “Close” column. The observational data will be put in a 987 x 1 matrix. In next section, we will introduce to some related research about Model Comparation and application of RNN Models in real-world. A short review of Models’ architectures and the dataset’s source will be also showed in the paper for the closer look of Models’ mechanic. We will apply Relu activation function in to RNN Models and take the results recorded for the summary at the last part. RNN model will be used for predict the value in next one month and compare to the actual situation.

II. RELATED WORK

Since RNN is introduced in 1986, there was many research papers about this concept and its variant. Some researchers revealed that some complex model such as RNN are more trustworthy than some simple one.

M. Ravanelli at al. [1] and Nana liu [2] applied RNN Models to process sound and speech signals.

A. Graves [4] showed that the power of a RNN Model to predict complex sequences just by processing a data point at a time.

Recently, A. Kumar [5] showed how important of model complexity in model selection. The article also showed the disadvantage of simple models (such as Linear Regression).

Nan Liu at al. [6], the research concluded that deep learning models are good at identifying objects but they need to be improved more to understand the relationship of objects. This means we need to build a complex architecture model to apply to a complex dataset.

Concluded by K. Cho at al. [7] and G. Chrupala at al. [8], a RNN Model can automatically learn a grammatical structure in a sentence.

J. Chung at al. [9] revealed that a GRU RNN Model is comparable to LSMT Model. Showing that GRU Model is also one of the best model in Time Series.

III. DATA SOURCE AND MODEL ARCHITECTURES

A. Data Source

This paper focuses on predict stock price was recorded daily from 11/31/2018 to 11/30/2022 of VNM Dataset

Figure 1. Stock price from VNM Dataset after Normalization

B. Recurrent Neural Network

Sequential data types are typically processed by RNN. The RNN models have a recurrent hidden state as in

$$h_t = g(Wx_t + Uh_{t-1} + b)$$

where x_t is an *m-dimension* input vector at the current time (t), g is the *activation function*, such as logistic function, or the Rectified Linear Unit (ReLU) [2, 10]. W, U *and* b are defined as sized parameters where W is a n x m matrix, U is an n x n matrix, and b is a vector with n elements. These sized parameters in this case are treated as two weights and one bias.

Tsungnan Lin at al. [11] showed that the gradients may vanish or explode after a number of timesteps if use such as a simple RNN. In [2-4], the idea of using some variants of RNN (LSTM and GRU) to solve the problem. We will present these two models below in details for our purposes.

C. Long Short-Term Memory (LSTM)

S. Hochreiter at al. [12] showed the idea of making a path that can let the gradient flow for a long time. F.A. Gers at al. [13] revealed that a crucial improve of LSTM model is make the *weight parameters* can be adaptively change through each time step by making *gated self-loop* to control the *weights* and the time for each integration can be adapted dynamically. The time in each integration can be also changed by *fixed parameters (weights)* with a set of suitable input values since the output of the model is the time for each time step.

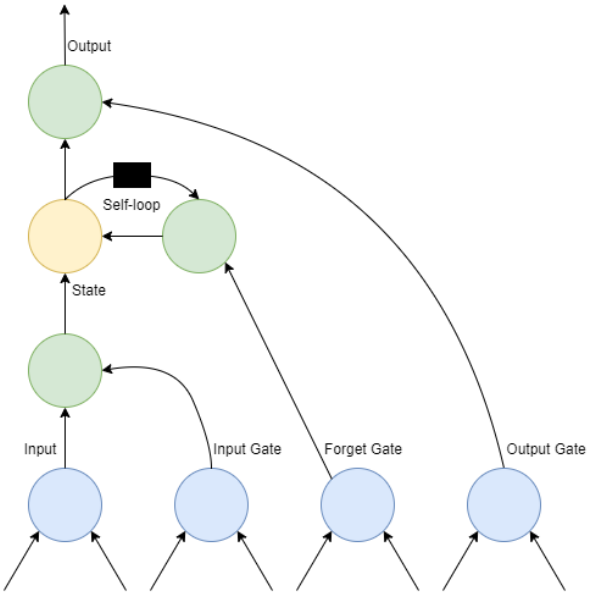

Figure 2. Diagram of a "cell" in LSTM model. Each cell connects with each other instead of regular hidden unit as in Graph Neural Network. Each input features are the result of a regular neural unit. The values can be stored in the State Unit of the cell. The State Unit has a linear self-loop with the weights are controlled by Forget Gate. All the Gate-Units can use non-linear sigmoid function, while the Input can use any non-linear function. The black square is the delay of each time step.

The abstract of a cell of LSTM model show in the *Figure 1*.

LSTM RNN use "LSTM cells" that have a self-loop. Each cell has the input and output values like a primitive recurrent network with more parameter and use *a system of gates* to control the information flow. As mentioned in the previous part, the *parameters (weigths)* of the self-loop are controlled by the *Forgot Gate* $f_i^{(t)}$(time step t and cell i) to

calculate the parameter within 0 to 1 using Sigmoid function.

$$f_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f x_i^{(t)} + \sum_j W_{i,j}^f h_i^{(t-1)})$$

where $x^{(t)}$ is the current input vector and $h^{(t)}$ is the current vector in hidden layer, contains the output values of the cell. b^f, U^f *and* W^f is the *bias value*, *weight* of input and *recurrent weight* of forget gate. The LSTM internal state is updated as follows:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma(b_i + \sum_j U_{i,j} x_i^{(t)} + \sum_j W_{i,j} h_j^{(t-1)})$$

where $g_i^{(t)}$ is calculated like the forget gate but with its own parameter

$$g_i^{(t)} = \sigma(b_i^g + \sum_j U_{i,j}^g x_i^{(t)} + \sum_j W_{i,j}^g h_i^{(t-1)})$$

The output $h_i^{(t)}$ can be shut off by the output gate $q_i^{(t)}$ and can be calculated by:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)}$$

$$q_i^{(t)} = \sigma(b_i^q + \sum_j U_{i,j}^q x_i^{(t)} + \sum_j W_{i,j}^q h_i^{(t-1)})$$

D. Gated Recurrent Unit (GRU) RNN

Get the idea from LSTM architecture, Gated Recurrent Unit inherited some its necessary features. Answered in [7, 8], the main difference between LSTM and GRU is that GRU have a single gating unit to update the state unit and control the forgetting factor.

By J. Chung at al. [9], the result showed that GRU RNN are much more advance than LSTM in most cases. There is a variant of GRU RNN, e.g. the Minimal Gated Unit (MGU) RNN which only use one gate equation and give the compatible performance (in some cases) to the LSTM RNN.

The update equations:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma(b_i + \sum_j U_{i,j} x_j^{(t-1)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)})$$

where *u* is update gate and *r* is reset gate. Update gate value is followed by the equation:

$$u_i^{(t)} = \sigma(b_i^u + \sum_j U_{i,j}^u x_i^{(t)} + \sum_j W_{i,j}^u h_i^{(t)})$$

and the reset gate value equation is:

$$r_i^{(t)} = \sigma(b_i^r + \sum_j U_{i,j}^r x_i^{(t)} + \sum_j W_{i,j}^r h_i^{(t)})$$

In this paper, we will focus on GRU RNN only and compare with the very basic Linear Regression model.

E. Bidirectional LSTM

In the traditional RNNs, the state at time *t* only captures the data from the past $x^1, \ldots, x^{t-1}$ and the current input x^t. However, in real world situations, many applications are also need *the whole input sequence* for the prediction y^t. For instance, in speech recognition, the correct interpretation of the current word may depend on the next few words because the context of a sentences can be decided by a totally random word.

People have implemented successfully (14) in many applications such as handwriting recognition (15) and bioinformatics (16).

In figure 3, a basic Bidirectional RNN is described by the combination of a sub-RNN that move from the beginning and a sub-RNN from the end of the sequence. The h^t is stand for the RNN that move forward and the g^t is stand for the RNN that move backward.

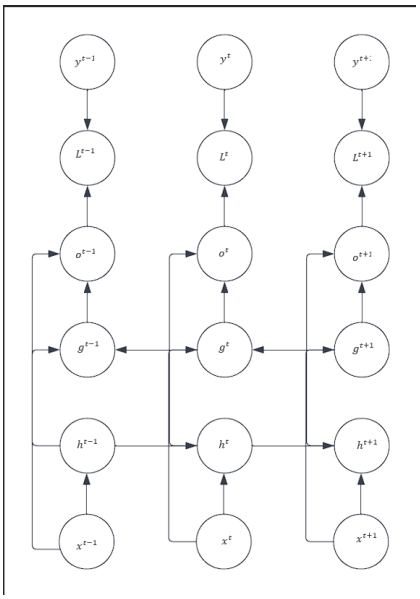

Figure 3. Bidirectional LST

F. Attention Mechanism

In Natural Language Processing, capturing the sematic of a very long sentence is very hard. There is an efficiency approach to this problem is *Attention Mechanism.* By this method, the RNN Models can read the whole sentence to get the general context and encode the words one at a time, each time focus on a specific part of the input sentence to predict the next word in the output sentence.

In figure 4 describes an abstract view of *Attention Mechanism* introduced by [17]. h^t is the state of a RNN Model at time t and a^t is the attention function at time t. People take the average weight of $weights\ h^t$ and $weights\ a^t$ to form the $vector\ c$ as a context vector.

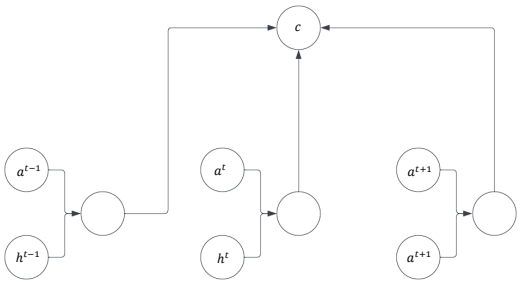

Figure 4. LSTM with Attention

G. Multivariable Linear Regression (MLR)

Multivariable Linear Regression (MLR) models have been used in various area to predict or classify data. MLR is a very basic model that take a vector $x \in R^n$ containing input values and vector $w \in R^n$ containing the weight values of each feature. We can define the output value by the following formula:

$$\hat{y} = w^T x$$

where $\hat{y}$ is the predicted value at the output.

In this case, w_i is a parameter(weight) to multiply with feature x_i and sum up all the value from feature x_i to get the predicted value ($\hat{y}$). Each weight in a MLR model is show that how importance of the data. If the weight is a positive number, increasing the value of its feature will also increase the predicted value. The predicted value in output neural will also decrease if the feature's value is increased while the weight is negative. This also concluded that if the weight is *zero*, the feature is mean nothing in our model.

In real-world situation, the MLR is sometimes more complicated with *intercept parameter (bias):*

$$\hat{y} = w^T x + b$$

where b is the bias value.

This will keep our model is still described as a straight line but does not need to go through the *origin point.*

Linear regression, basically, is a very basic algorithm and has a lot of disadvantages. But it will give us an overall view of *Machine learning algorithm.*

H. Multivariable Non-Linear Regression (MNLR)

Multivariable Non-Linear Regression is a version of regression analysis. The observational data are put in a non-linear model. We can define the output of the model by the following formula:

$$y \sim f(x, b)$$

where y is the output value. The function f is nonlinear and takes x and b as the vector of independent variables and bias values respectively.

IV. MODEL SETTINGS

A. RNN

In this task, we built three RNN models with three layers. The first RNN Model is applied the Gated Recurrent Unit in each layer, the second RNN Model is a standard Bidirectional LSTM and the last one is a basic LSTM model with Attention Mechanism. Each layer has 20% dropout rate to prevent the result from overfitting and the activation function is set as *relu* function. The model is performed by Python language using *Keras library* with *Tensorflow* as the backend library. The hyperparameter of this model is set as Learning rate: 0.0001, Hidden Unit: 64, Batch size: 32, Epoch: 100, Optimizer: Adam.

In the dataframe, we divined the dataset into three set. 80% for training dataset, 10% for validation dataset and 10% for testing dataset.

Training data set:

Figure 5. Training set (80%)

Validation data set:

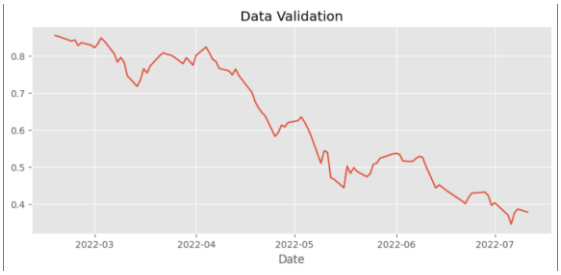

Figure 6. Validation set (10%)

Testing data set:

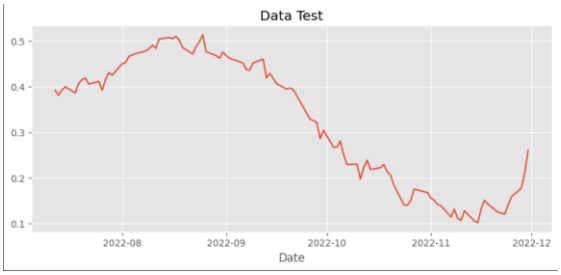

Figure 7. Testing set (10%)

B. MLR Model

In this test, we run a very basic MLR Model using Sklearn library in Python language. The dataset is split to training set (80%) and testing set (20%).

C. MNLR Model

In this model, we used Random Forest Regression on the test with 500 of trees in the forest, 100 at random state and return Out of Bag score. The dataset is split to training set (80%) and testing set (20%).

V. RESULT AND CONCLUSION

In this section, we will show the test results on three mentioned models and evaluate the result by MAPE Loss, RMSE Loss and MAE Loss.

Table 1. Evaluate Metric based on MAPE, RMSE and MAE score

	MAPE	**RMSE**	**MAE**
GRU	14.1%	0.3	0.2
B-LSTM	14.3%	0.26	0.2
A-LSTM	14.5%	0.29	0.2
MLR	1.4%	0.28	0.2
MNLR	19%	0.3	0.2

Based on the results recorded, we revealed that RNN Models brought a very good result. The MAPE score is under 15% and the models' settings predicted well on the dataset. MNLR Model based on Random Forest Regression is reliable but cannot be comparative to RNN Model based on the 19% of MAPE Score. Overfitting is happened on MLR Model and cannot be reliable. The comparison result on test data is showed as below:

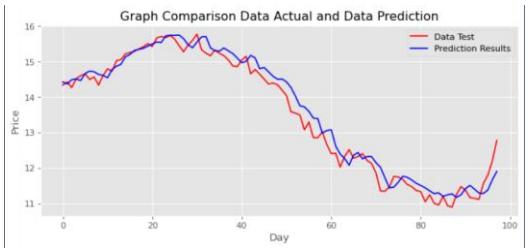

Figure 8. Compare the actual value and predicted value in GRU Model

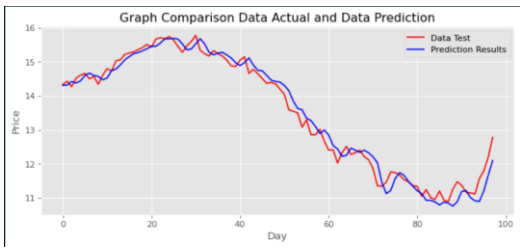

Figure 9. Compare the actual value and predicted value in Bidirectional LSTM

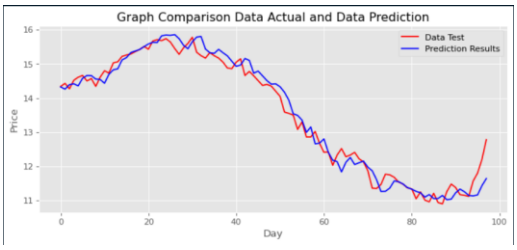

Figure 10. Compare the actual value and predicted value in LSTM Model with Attention mechanism

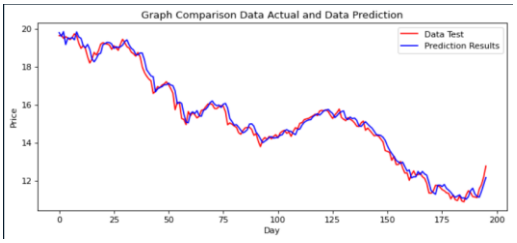

Figure 11. Compare the actual value and predicted value in Multivariable Linear Regression Model

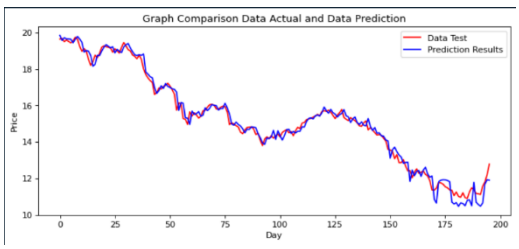

Figure 12. Compare the actual value and predicted value in Multivariable Non-Linear Regression Model

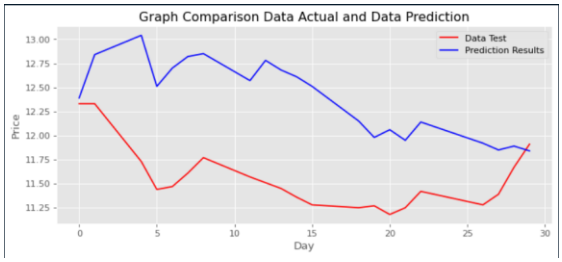

Figure 13. Predicted values of next 30 days of GRU Model

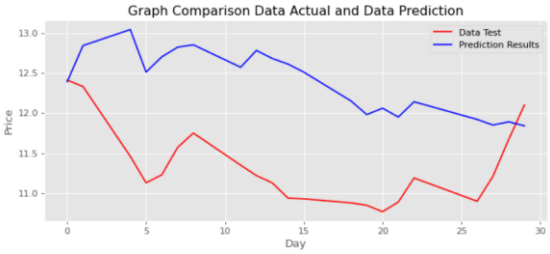

Figure 14. Predicted values of next 30 days of B-LSTM Model

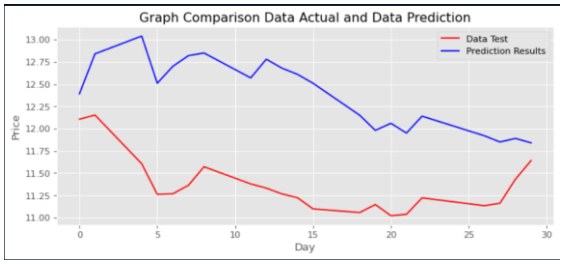

Figure 15. Predicted values of next 30 days of A-LSTM Model

In Figure 13, 14 and 15; the results of predicted values in next 30 days are described as the blue line and the actual values is described as the red line. Based on the overall view, we can see that the model is predicting well on unseen data when the different from the actual outputs and predicted output is in a close range. They are also showed the correct trend of the graph's path. The result revealed that RNN Variants is comparative in Time Series Dataset.

REFERENCES

[1] M Ravanelli. Light Gated Recurrent Units for Speech Recognition. arXiv:1803.10225, 2019

[2] Nana Liu. Music Emotion Recognition Model Using Gated Recurrent Unit Networks and Multi-Feature Extraction. Hindawi || Article ID: 5732687, 2022

[3] A Graves. Generating Sequences With Recurrent Neural Networks. arXiv:1308.0850, 2013

[4] Ian Fellow. Deep Learning Book pg 407-410. 2015

[5] A Kumar. Model Complexity & Overfitting in Machine Learning. Model Complexity & Overfitting in Machine Learning - Data Analytics (vitalflux.com), 2022

[6] Nan Liu, Shuang Liu and Yilun Du. Learning to Compose Visual Relations. arXiv:2111.09297, 2021

[7] K Cho, B v MerrienBoer, D Bahdanau and Y Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259, 2014

[8] G Chrupala, A Kádar and A Alishahi. Learning language through pictures. arXiv:1506.03694, 2015

[9] J Chung, C. Gulcehre, K Cho and Y Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555, 2014

[10] Quoc V Le. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. arXiv:1504.00941, 2015

[11] Tsungnan Lin, Bill G. Horne, Peter Tino, C. Lee Giles. Learning long--term dependencies is not as difficult with NARX recurrent neural networks, 2001

[12] S Hochreiter and J Schmidhuber. Long short-term memory. Neural Computation 9(8):1735-80, 1997

[13] F A Gers, J Schmidhuber and F Cummins. Learning to forget continual prediction with LSTM. IEEE Xplore, 10.1049/cp:19991218, 2002

[14] Alex grave. Supervised Sequence Labelling with Recurrent Neural Networks. 2012

[15] Alex Graves, Jürgen Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. 2008

[16] Pierre Baldi, KurtHornik. Neural networks and principal component analysis: Learning from examples without local minima. 1989

[17] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473. 2016