

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

A picture containing game

Description automatically generated

**Lecture: Nguyễn Đình Thuân**

**Nguyễn Minh Nhật**

**Authors: Hà Gia Huy**

**Đậu Đình Quang Anh**

**Trần Đức Duy**

### [STAT3013.N12.CTTT](https://courses.uit.edu.vn/course/view.php?id=10066)

**LAB02 REPORT**

**Intermediate Statistic Analysis**

# A. Model architecture

## 1. Recurrent Neural Network

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

where is an *m-dimension* input vector at the current time (*t*), is the *activation function*, such as logistic function, or the Rectified Linear Unit (ReLU) [1, 2]. are defined as sized parameters where 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. [3] showed that the gradients may vanish or explode after a number of timesteps if use such as a simple RNN. In [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.

## 2. Long Short-Term Memory (LSTM)

S. Hochreiter at al. [5] showed the idea of making a path that can let the gradient flow for a long time. F.A. Gers at al. [6] 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.

Diagram

Description automatically generated

Figure 1. 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* (time step t and cell i) to calculate the parameter within 0 to 1 using Sigmoid function.

where is the current input vector and is the current vector in hidden layer, contains the output values of the cell. is the *bias value, weight* of input and *recurrent weight* of forget gate. The LSTM internal state is updated as follows:

= +

where is calculated like the forget gate but with its own parameter

The output can be shut off by the output gate and can be calculated by:

## 3. 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:

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

and the reset gate value equation is:

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

## 4. Bidirectional LSTM

In the traditional RNNs, the state at time *t* only captures the data from the past and the current input However, in real world situations, many applications are also need *the whole input sequence* for the prediction 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 [10] in many applications such as handwriting recognition [11] and bioinformatics [12].

In figure 2, 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 is stand for the RNN that move forward and the is stand for the RNN that move backward.

## 5. 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 [13]. is the state of a RNN Model at time and is the attention function at time t. People take the average weight of and to form the as a context vector.

Diagram

Description automatically generated

Figure 2. LSTM with Attention

Diagram

Description automatically generated

Figure 3. Bidirectional LST

# B. Model implementation

## 1. Data prepare

* In this task, we will use the data from VNM dataset which contains 988 observations. In figure 4, we visualized the dataset after the Min-max scaler:

Graphical user interface, text

Description automatically generated

Chart, line chart

Description automatically generated

Figure . Data Normalized

* Split the data set into three sub-sets: 75% of training set, 15% of testing set and 10% of validation set and put them into three array by sliding window algorithm

Text

Description automatically generated

Figure . Split dataset into training, testing and validation sets

Text

Description automatically generated

Figure . Sliding window

## 2. Model’s hyperparameters

Table . Hyperparameter

|  |  |
| --- | --- |
| Hyperparameter | Values |
| Learning Rate | 0.0001 |
| Dropout | 20% |
| Hidden Unit | 64 |
| Batch size | 32 |
| Optimizer | Adam |
| Epoch | 100 |
| Activation Function | Relu |

* In Table 1, we applied the following hyperparameter in all of three RNN Variants

### a. Bi-LSTM

* Training model has 5 layers, 1 input layer, 3 bidirectional LSTM, 1 dropout layer and 1 output layer. It will compiling the gated recurrent unit and Fitting ke data training dan data validation
* Perform manual selection of the values ​​of Hyperparameters.

Text

Description automatically generated

* Create the graph of Model Loss with x label is ‘loss’ and y label is ‘epoch’

Text

Description automatically generated

* And here is the result:

Graphical user interface

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

* Implementation model into data test and Invert normalization min-max.

A picture containing text

Description automatically generated

* Use the model and make predictions

Text

Description automatically generated

* Graphical user interface, table

  Description automatically generatedHere is the result.

Evaluate predicted models and actual prices through graph and the result.

Chart, line chart

Description automatically generated

* Calculate MAPE, RMSE and MAE.

Graphical user interface, text, application, email

Description automatically generated

### b. GRU

In figure 7, we applied the hyperparameters in to the GRU Model

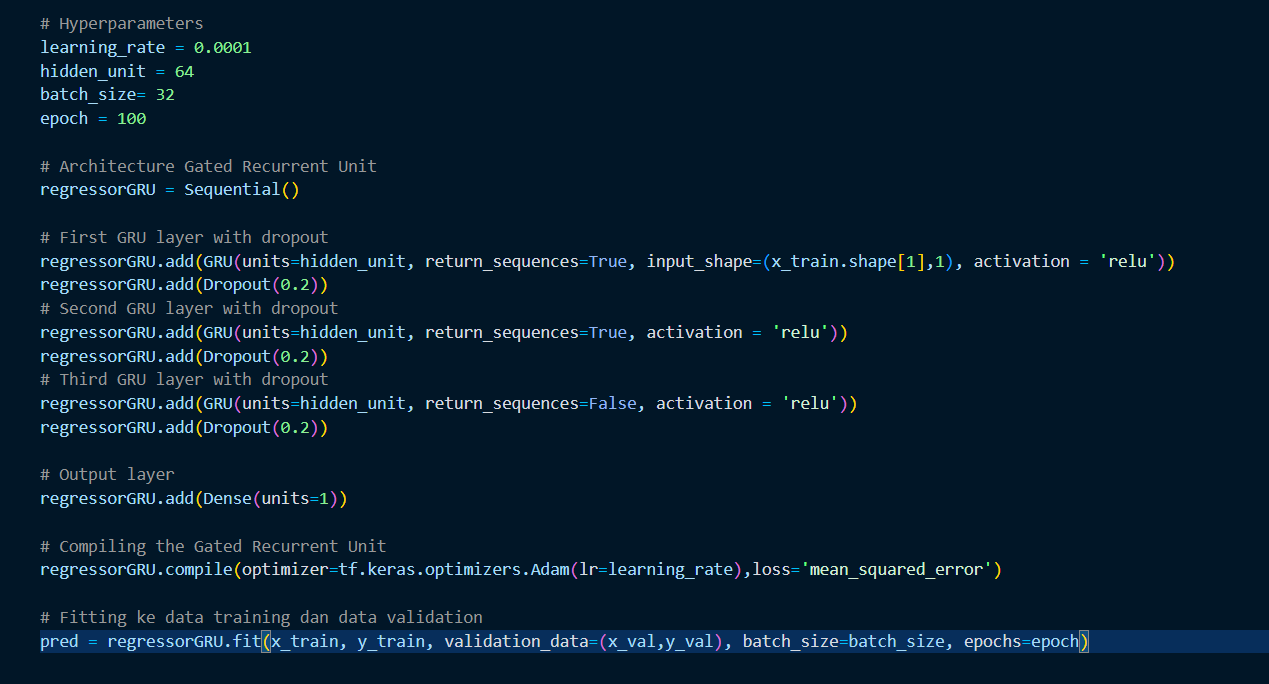


Figure . Build Model

* Use predict function to use the model to perform the prediction on the test set

Text

Description automatically generated

Figure . Model prediction

* Compare the actual value and the predicted value

Graphical user interface, text

Description automatically generated

Figure . Compare the actual and predicted output

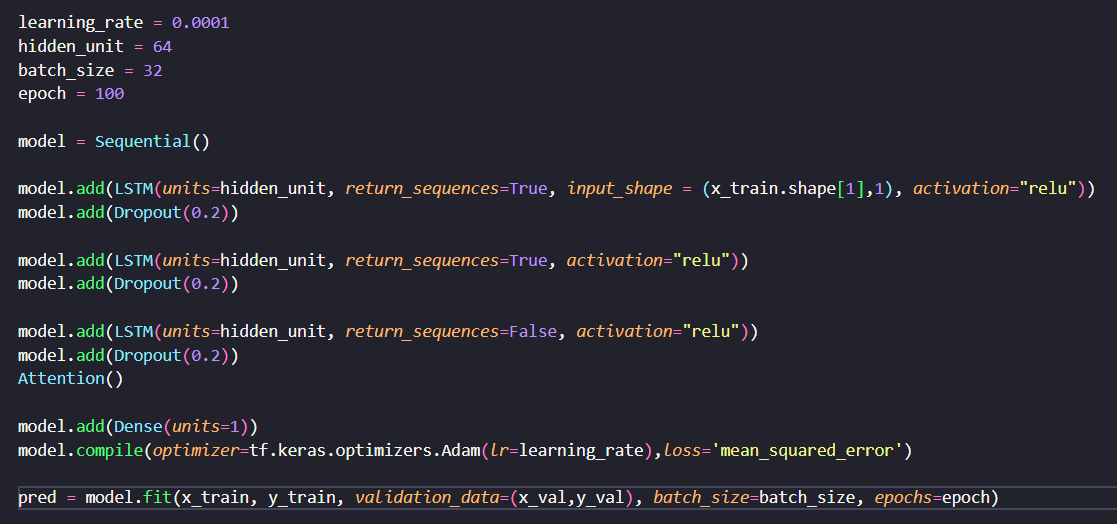
* Visualize the comparison

Chart

Description automatically generated

### c. A-LSTM

* Training model has 5 layers, 1 input layer, 3 bidirectional LSTM, 1 dropout layer and 1 output layer. It will compiling the gated recurrent unit and Fitting ke data training dan data validation
* Perform manual selection of the values ​​of Hyperparameters.



* Use predict function to use the model to perform the prediction on the test set

Text

Description automatically generated

* Compare the actual value and the predicted value

Text

Description automatically generated

* Visualize the comparison

Chart, line chart

Description automatically generated

WORK SHEET

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Job | Status | Source |
| Hà Gia Huy | GRU, evaluate and parameter selection for model | Done |  |
| Đậu Đình Quang Anh | B-LSTM and A-LSTM | Done |  |
| Trần Đức Duy | Multivariable linear regression, multivariable nonlinear regression, RNN theory and Simple LSTM | Done |  |

# Reference

[1] Nana Niu, "Music Emotion Recognition Model Using Gated Recurrent Unit Networks and Multi-Feature Extraction", Mobile Information Systems, vol. 2022, Article ID 5732687, 11 pages, 2022. <https://doi.org/10.1155/2022/5732687>

[2] Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton, “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units”, arXiv:1504.00941, 2015

[3] Alex Graves, “Generating Sequences With Recurrent Neural Networks”, arXiv:1308.0850, 2014.

[4] M Ravanelli, “Light Gated Recurrent Units for Speech Recognition”, arXiv:1803.10225, 2019

[5] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997 Nov 15;9(8):1735-80. doi: 10.1162/neco.1997.9.8.1735. PMID: 9377276.

[6] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: continual prediction with LSTM," 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), 1999, pp. 850-855 vol.2, doi: 10.1049/cp:19991218.

[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] Grzegorz Chrupała, Ákos Kádár, Afra Alishahi, “Learning language through pictures”, arXiv:1506.03694, 2015

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”, arXiv:1412.3555, 2014

[10] Alex Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”, 2012

[11] Alex Graves, Jürgen Schmidhuber, “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”, 2018

[12] Pierre Baldi, Kurt Hornik, “Neural networks and principal component analysis: Learning from examples without local minima”, Volume 2, Issue 1, Pages 53-58, ISSN 0893-6080, 1989, <https://doi.org/10.1016/0893-6080(89)90014-2>.

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