COMP_SCI_7318 - Assignment 3 - RNNs for Stock Price Prediction

Hong Phuc Pham - Student ID: a1843625
The University of Adelaide
230 North Tce Australia SA 5005
hongphuc.pham@student.adelaide.edu.au

Abstract

Recurrent neural network (RNN) [19] is a class of artificial neural networks that reuse the previous outputs while having hidden states. The RNNs process variable input sequences to predict an observed chain's potential outcome. In this assignment, RNNs, experiments, and performance on the daily stock price prediction will be built. Specifically, the Sony Group Corporation stock price will be observed from 2015 to 2022. The different RNN models will be implemented and tested. Activities are conducted to understand the mechanism of RNNs. These models will be implemented with Python, using the Keras library [24] to build the network. Some tests will be tried to determine the algorithm's accuracy on the chosen data. The main task is predicting the stock price the next day based on the previous day's price. The mean error has reached 1.23, the maximum error is 12.25, and the minimum error is 0.00 through several tests. The maximum error found is 26.97 with the lengthened sequence data.

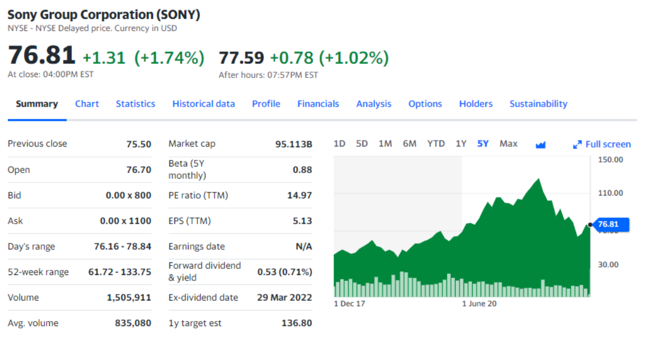
1. Introduction

RNN is initially developed in the 1980s. Nevertheless, due to the undeveloped state of technology at that time, the RNNs concept did not gain much concern from the public as it is nowadays. This algorithm is now widely used for task handling with sequential data and natural language processing. With the current state-of-the-art technology and internal memory, the concepts of RNN got their place and can be found in several applications such as Apple's Siri [25], or Google'voice search [11]. As part of the application of it in the sequential data, this concept application will be brought up and observed on how well it performs on the stock price data, which uses the stock history to predict next-day price. Due to the economic system's development, the company's ownership can be divided and exchanged in the market. Everyone with fine financial ability can own a stock and become a part-owner of that company. However, the values of stocks are not stable; they vary in many ways and change for multiple reasons which are not purely attached to economic status. Buying and selling stocks do not just stop at the general perception of a good. These activities have grown to a new level and become a job as well as gaining the attention of the public and companies themselves. With the rise of stock interest and control ambition, people are seeking the method to predict the stock value for their trading purpose or company valuation. RNN will be implemented to study the appropriateness of stock price prediction. In this assignment, there are three main algorithms will be used, which are traditional RNNs [10], LSTM [10], and GRU [10]. Also, these main types of RNNs will be tested on tuning parameters with different sequence input lengths, batch size, trick and network structuring to improve the performance. For these experiments, the main aim will be to minimize the prediction error on the stock price. Therefore, the assignment is focused on the implementation, testing, and observation of the RNN in stock price prediction, not the stock trading decision.

2. Dataset

The Longman Bussiness English dictionary defines stock in the financial term, which comprises the share of divided corporation or company ownership [8]. Depending on the total number of shares, the fractional company ownership of a single share of the stock gain on that proportion. Certain stocks will be entitled/ able to involve with or without voting rights, enhancing voting rights, priority claiming for receiving profits or liquidation proceedings. The stock exchange can be private or public and follow government regulation as the liquidity of stocks, potentially high return values, out-space of inflation, and the risers of passive income concepts, more and more people get involved in this. Stock investing is not a walk in the park. The values keep changing through time, and some scale of events worldwide can affect the price like war, financial status, someone's speech, Twitter, news, and even the stock owners exchange activities [7, 14, 17, 23].

Place all the relative factors aside; the focus is now narrowed to the price transition. The experiments are conducted to seek the possibility of overseeing the outcome of the continuous stock value or the close value to it. The data will be used for these experiments is Sony Group Corporation, or Soni Gurupu Kabushiki gaisha [21]. This multinational corporation is famous for its electronic product, one of the world's largest technological manufacturers. Sony is involved in many industries like entertainment (picture and music), electronics, and interactive entertainment (games, finance, creatives, semiconductor solutions, and education). The diversity and worldwide branding will bring up several factors that affect the stock. The data all collected from Yahoo! Finance [27] - a media network providing news, data, and commentary about finance, including press releases, financial reports, and stock quotes. The data is the collective of the daily stock price with a record of the high, low, open, close, volume and adjusted close in seven years (from January 1, 2015, to January 1, 2022). For the RNN demo, the close price with the last transaction of the day will be used for the model training and prediction. Yahoo illustrates the summary generated of Sony stock over five years! Finance in figure 1.


Sony Group Corporation (SONY)
NYSE - NYSE Delayed price. Currency in USD

76.81 +1.31 (+1.74%) **77.59** +0.78 (+1.02%)
At close: 04:00PM EST After hours: 07:57PM EST

Previous close	75.50	Market cap	95.113B
Open	76.70	Beta (5Y monthly)	0.88
Bid	0.00 x 800	PE ratio (TTM)	14.97
Ask	0.00 x 1100	EPS (TTM)	5.13
Day's range	76.16 - 78.84	Earnings date	N/A
52-week range	61.72 - 133.75	Forward dividend & yield	0.53 (0.71%)
Volume	1,505,911	Ex-dividend date	29 Mar 2022
Avg. volume	835,080	1y target est	136.80

Figure 1. Sony stock market summary over five years. Image taken from Yahoo! Finance at 16:50, November 9, 2022 [28].

3. Method

The dataset is straightforward with the features. For the upcoming experiments, only the close price is used for this assignment. Therefore the close price will be extracted. For training and test splitting, the ratio of 7:3 will be applied—the price collected from 1763 days. After splitting, the train data size will initially be 1234 and 529 for testing. Next step, the dataset needs to adjust to make it workable with RNNs. For the simple model test, the time interval of 30 days will be used. The dataset will be processed to have a stock price of 30 days as an input, and the target value will be kept the same. After the processing, the size of the training and testing dataset becomes 1204 and 499. Three main types of RNNs will be tested traditional RNN, LSTM, and GRU. The initial modes will be trained with the same batch size of 32 and 50 epochs, using Adam optimiser, with the mean squared error loss and default learning rate (0.001 for Keras). The best will be chosen for continual improvement through the three model's performance. Hyper-parameter tuning, interval changing, tricks and layers, and model reconstruction to improve the model's performance. The original Sony stock price is shown in figure 2 as the target values are on a broad range of values, which does not have a certain number of classes. Therefore, the best metric to be considered along with accuracy is the error to observe the difference between the prediction and the truth values. The min, max, and mean errors on the test set will be checked in every model.

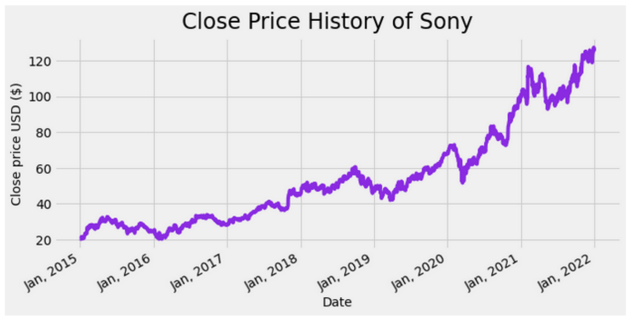


Figure 2. Sony daily stock close price from January 1, 2015 to January 1, 2022. Data collected from Yahoo! Finance.

Three initial investigative models will be set up with four RNN-type layers and one total connected layer at the end to get the output. All four RNN layers will use the Tanh activation function [10, 26] and set with 30 units. The three first layers will return the sequences. No dropout or batch normalisation; regularisation will be set in these first initial models. The models will be run through manual tests with different batch sizes (16, 32, 64) and different sequence lengths (30, 50, 90). The sigmoid function will be compared with the Tanh. The normalisation, dropout, and gradient clipping will be added. Then the automation tuning will be run and tested with the most mentioned parameter and different units number. The run time of models will be observed in this assignment.

Although the code will use the Keras library to build up the model, train, and test, the mathematical logic for each cell, hidden state, and how the result will be generated will be discussed in the subsection 3.2 for RNN, 3.3 for LSTM and GRU in subsection 3.4 of this part. Notably, the train data-generating method will be shown in the following subsection.

3.1. Generate training data

For RNNs, the dataset needs extra preprocessing. The training data is a chain of information rather than a single piece, as CNN's. The concept would be related to the sliding window. We will have a similar size to the sliding window depending on the desired sequence length. On the sequence of the stock price, it will iterate through its length and take a copy of the data sequence. The previous data will be used to predict the next date price. The idea of generating training data will follow pseudo-code 1 (which is written for Python language). The mechanism of how the output will be calculated will be shown in the following subsection.

Algorithm 1 RNN training data generating algorithm

```
Require:
  $\eta$: time step or the sequence length
  S: data sequence
  N: data size
  X: training data array
  y: truth values array
Ensure:
  for t = $\eta$ to N do:
    $X_t \leftarrow S[t-\eta : t]$
    $y_t \leftarrow S[t]$
```

3.2. Traditional RNN

One of the first observed models is the traditional RNN. For one sentence explanation, the RNNs is a generalisation of feed-forward neural networks with internal memory. It functions identically for every data input. The output will be generated based on the computation of the recent past outcome and current input. Then the current network will be updated with the new outcome [1]. The internal memory will be used to process the input sequences. This makes it different to the feed-forward network, where the signals travel one way only. As the stock price is unpredictable, the RNNs feedback loop network gives the dynamic track for the model to follow. Therefore the formula for the current state will be:

$$h_t = f(h_{t-1}, x_t) \tag{1}$$

Where h_t is the current state, h_{t-1} is the previous state, and x_t is the input state. Tanh activation function is a popular choice for the hidden layers of RNN. Tanh activation will have to save the gradient computation. It produces a zero-centred output, which supports the backpropagation process. In the case of ReLU, it will swiftly eliminate the negative gradient (by shifting them to 0), which will stop the network's learning, where the output will be produced the same. The formula for Tanh activation function will be:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2}$$

The current state 1 with the application of activation function 2 will become:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{3}$$

W_{hh} is the weight at the recurrent neuron and W_{xh} is the weight at input neuron. Therefor the output result will be with the weight at output layer W_{hy}:

$$y_t = W_{hy}h_t \tag{4}$$

3.3. LSTM

LSTM, or Long short-term memory, is a subtype of recurrent neural network. Although RNNs are designed with internal memory to retain the previous information, it still struggles with lengthy sequences [4]. LSTM memory cell has three gates: input, forget, and output. They determine whether the information will be saved or not or should be forgotten, which is banked on its importance. Therefore it can read, write, and delete information in memory. The importance of information will be assigned through the weight learned by an algorithm. LSTM also solved the problem of vanishing and exploding gradients as it kept the gradient acceptably steep [16]. LSTM cell architecture is illustrated in figure 3 below. Formula for LSTM could be summarised by

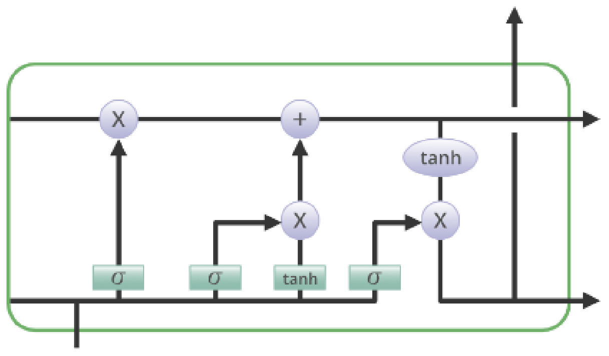


Figure 3. LSTM cell illustration [15]

the following formula in one time step.The input x_t and o_t for the LSTM, h_{t-1} and c_{t-1} are input form the previous time step. What LSTM will do is generating the c_t and h_t for the next time step use. The forget gate f_t will be

$$f_t = \sigma_g(W_f \times x_t + U_f \times h_{t-1} + b_f) \tag{5}$$

The input gate (i_t) will be calculated by this formula where σ_g is symbolised of Sigmoid activate function.

$$i_t = \sigma_g(W_i \times x_t + U_i \times h_{t-1} + b_i) \tag{6}$$

The output gate (o_t) result will be generated by this:

$$o_t = \sigma_g(W_o \times x_t + U_o \times h_{t-1} + b_o) \quad (7)$$

Then we calculate one of internal consumption of LSTM (c'_t) and using Tanh activation function (σ_c)

$$c'_t = \sigma_c(W_c \times x_t + U_c \times h_{t-1} + b_c) \quad (8)$$

Then the current cell state (c_t) will be generated through the element wise multiplication of forget gate with previous cell state and input gate with internal cell consumption.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t \quad (9)$$

Finally the hidden state (h_t) will be

$$h_t = o_t \cdot \sigma_c(c_t) \quad (10)$$

These formula 5, 6, 7, 8, 9, 10 are mechanism computing in a single time step. Biases (b_f,b_i,b_o,b_c) and the weights (W_f, W_i, W_o, W_c, U_f, U_i, U_o, U_c) do not change from time step to another.

3.4. GRU

Gated Recurrent Unit (or GRU) is another upgrade version of traditional RNNs which helps to solve the typical problem - vanishing gradient. GRU has updated the gate and reset the gate. They take responsibility for which information should be passed to the output. They can work on and keep the long history information which does not need to remove or clean irrelevant pieces for prediction through time. A simple illustration of the GRU unit is shown in figure 4. The input x_t will be multiply with the own weight W_z adding up with multiplication of previous hidden state h_{t-1} with it own weight U_z, then go through sigmoid function (σ_g) to get the update gate z_t for the time step t:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1}) \quad (11)$$

This update gate will determine how much the information from the previous time step should be passed to the future. Next, like its name, the reset gate will take care of the past information about how much information from the past should be forgotten. Its formula is the same as the update gate.

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1}) \quad (12)$$

The current memory content will be will gained through this formula:

$$h'_t = \sigma_c(W x_t + r_t \odot U h_{t-1}) \quad (13)$$

And the final memory at current time step will be:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t \quad (14)$$

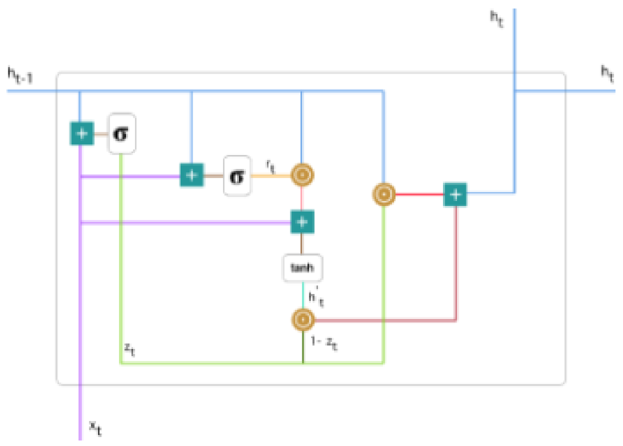


Figure 4. simple GRU unit where σ - Sigmoid function, tanh - tanh function, yellow circle - Hadamard product function [12].

3.5. Other experiments

After choosing the best type of RNN, the model will have brief observations through gradient clipping with SGD optimiser as the learning rate - 0.01, decay rate - 1e-7, the momentum of 0.9 and clip normalisation at 1. Then some layer normalisation [3] and drop out layer [31] will be added to the model. Different sequence lengths will be tested with 14, 50, and 90. Lastly, the original best model's hyperparameters will be tuned and tested to choose the best combinations using the Keras tuner package functions. Different choice of optimisers - SGD and Adam along different learning rates ($0.1, 1e{-}2, 1e{-}3, 1e{-}4, 1e{-}5$). Different choices of drop-out rates will be searched with the following values - $0.0, 0.1, 0.25, 0.5$. The number of units for layers within these variants - $16, 32, 64$ will also be checked. The last fully connected layer will have the tuning for the activation function (ReLU or Sigmoid). The loss will be a mean squared error (mse), and the metrics combine mse and accuracy. The tuning checking for each iteration only goes a maximum of 15 epochs. The objectives will be similar to the metrics.

4. Experimental Analysis

These experiments undergo several trials of model construction, tuning, and applying tricks. These tests were conducted to observe the effect of the different layers on the network. Also, to get the best of the model, tuning is used to find the best hyper-parameters for the model—some attempts with tricks to levitating the performance. The experiments were observed from a traditional RNN layers model and then changed to different RNNs layer types. Next, a gradient clipping will be used with the SGD optimiser, and regularisation or normalisation will be tested. Later, the test will be about the variation of parameters of the unit, batch, and activation. In the end, the tuning will be manual, and the library function will be applied later. The reason for this test is to seek the optimal model structure first and then apply more tricks with parameters.

4.1. Initial models test

The exact structure, optimiser, parameter values, same dataset and data size on both train and test will apply for the first three models to have the most neutral view about their performance. For these first models, their training time, max error, min error, and mean error on the test dataset will be observed. The traditional RNNs have the fastest run time of 62 seconds of the three. LSTM and GRU have close similar run times with only a one-second difference. Despite the run time factor, GRU got the lowest mean, min, and max error compared to the other two. The max error of GRU is 12.25$, some perdition does not make an error at all, and the mean error is 1.23$. The summary of their performance is recorded in the table 1 below.

Model	Max error	Mean Error	Min error	Run times
Simple	13.14	1.63	0.01	62s
LSTM	14.78	2.14	0.00	150s
GRU	12.25	1.23	0.00	149s

Table 1. Test conducted with the simple build on traditional RNNs, LSTM, GRU.

Their accuracy of them in training got low, 8.3e-4 for traditional RNN, 0.0017 on LSTM and GRU. The loss of three models does not drop, or they loop up and down with minor changes. The LSTM model has a small amount of delay on the prediction, with soft prediction, while in GRU and traditional, the prediction is closer to the actual price. GRU also has a softness in prediction but is not too delayed, like the LSTM model in this experiment. Figure 5 show the performance of the initial GRU model.

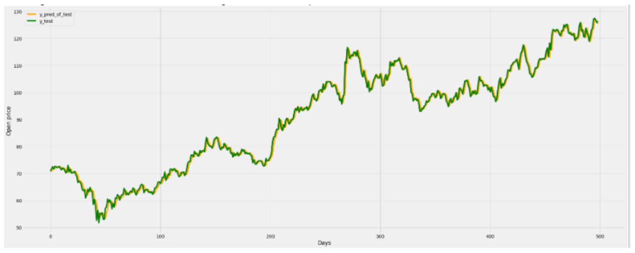


Figure 5. Initial GRU prediction and the truth values.

4.2. Other experiments

As GRU got the best result in the initial test, these models will be used for add-on experiments. In the first test of the gradient clipping, the loss decreased quickly in the first few epochs and then dropped slowly. The performance is similar to the original test; the max error of this model is 16.20, the min error is 0.00, and the mean error is higher with the original model with 2.55. Some Keras dropout layers (with a rate of 0.1) or layer normalisation is added to the model, but the performance did not lift. Between dropout and layer normalisation, layer normalisation got better performance when going with the gradient clipping with the max error as 13.56, and min error remained at 0.00, mean error reduced slightly to 2.37.

The coming test is on the different data lengths—datasets with different intervals of $14, 50, and 90$ days price stock. The model will be used to train the initial GRU model then the results can be compared with the sequence length of 30. None of them beat the initial data length. The length 14 could be said to have the closest performance with the original one with max error at 16.22, min error at 0.01, and mean error at 3. The longer the data sequence the GRU model got to struggle with the prediction. The values are smoothing, losing sharp points, but still catching the orientation of the flow with a short delay which is illustrated in figure 6. In other words, this is more like the train prediction where the line of the oversee values is nearly straightened.

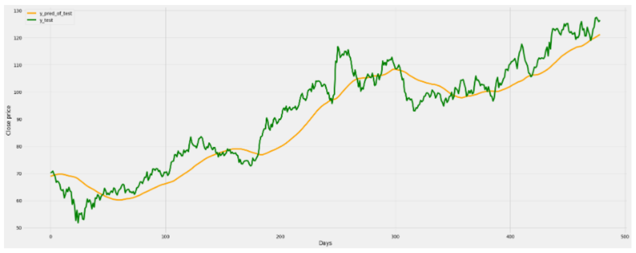
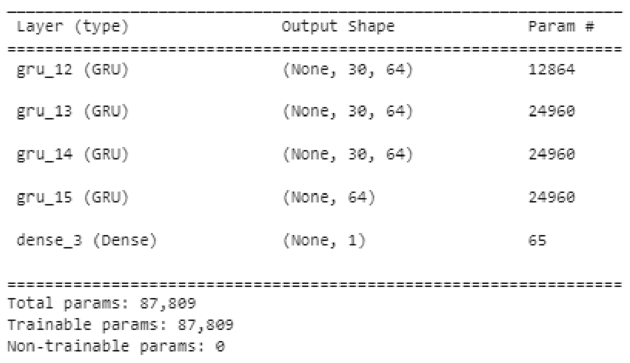


Figure 6. Initial GRU prediction with 90 days stock price sequence length.

The hyper-tuning worked on the different optimiser, learning rate, dropout rate, and unit number of layers with various options and values. The tuner went through 293 combination. The detail of each combination will be shown in the Jupyter notebook file in the link of the coding section 6. The final model got a similar performance to the original GRU model with 12.62$ for the max error, 1.52 for the mean error and the min error is 0.0. The detail of the tuned model is shown in figure 7. The predicted price is a couple of days delay. In the first 250 days, the predicted price is higher than the truth values. However, after that, the price is lower than the true value.

5. Discussion

The three most famous and typical models in the experiment - traditional RNN, LSTM, and GRU- have been built and observed. From the summary of the table 1, the


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_12 (GRU)                (None, 30, 64)            12864

 gru_13 (GRU)                (None, 30, 64)            24960

 gru_14 (GRU)                (None, 30, 64)            24960

 gru_15 (GRU)                (None, 64)                24960

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 87,809
Trainable params: 87,809
Non-trainable params: 0
```

Figure 7. Tuned model summary.

GRU get the best result with the lowest error in all three categories of max, mean, and min difference. Traditional models work well with the quickest run time. However, the traditional RNNs could get into the gradient exploding or vanishing [4]. Therefore either LSTM or GRU would be a better choice to avoid the problem. The experiment also confirmed that GRU is a faster LSTM model with the same dataset [30]. As this dataset is small and straightforward, the run-time recorder did not save time in smaller units when the total length is over one second, which is why the distinction between the two models was not fully present. In several research and application fields of prediction for text, traffic flow, and symbols, GRU performs better with the simple and short length of data [6, 9, 20, 29]. GRU is faster because it has less than one gate than LSTM, making it have fewer training parameters.

The extra setups with gradient clipping, dropout, and normalisation affect the model training. After applying the layer normalisation, the error has been brought down specific values in max error and mean error. Technically, regularisation, gradient clipping, and normalisation are the techniques to prevent gradient exploding and vanishing [**?**, 3, 22]. These techniques prevent the model stick too much to the data details, and then the model will be more generic and will not be over-fitted.

The different lengths of the training data affect the model learning. The more data, the model catches the trend, but the data will be more generic. However, the short sequence will not give enough information for the model to learn that would be either too much noise or too general for the model to give a prediction value. Through the experiment, the 30 days of the stock price is the appropriate length for the model to learn. More units for layers or using LSTM will be better for the long sequence [30].

Hyperparameter tuning improved the performance of the model. The most different value from the initial model is the number of units in the layers, which extends the capacity of the model to learn a more extensive or complicated set of mapping functions [10].

The learning rate directly affects how fast the algorithm learns and will the cost function is minimised. The optimal value is not easily found. It also depends on which optimise function is used. Either higher or lower than optimal values, the model will get stuck in learning, and the cost function will also go high.

The hyper-tuned return the top ten results, with none having a rate different than 0.0. Therefore the setup of these dropout layers is unnecessary. However, some extra implements that could be implemented will be mentioned later in the limitation section 7.

6. Code

The code, detail for the RNNs implementation, testing, PDF coding file, and data set are at following Git hub repository: `https://github.com/hongphuc-pham/CNNs_for_image_classificationhttps://github.com/hongphuc-pham/RNNs_for_Stock_Price_Prediction`

7. Limitation

The implementation RNNs have been quicker and more straightforward with the help of the Keras packages. The machine learning mastery page inspires most of the code and experiments. More examples of the manual build from scratch and more different structures can be found on the following pages: `https://machinelearningmastery.com/` [5]. The assignments focus on understanding and working on the concepts of the RNNs. Three classic RNNs models have been implemented. However, some more types or designs could be checked or applied to solve this case, like Armin [13], Prophet [32], and bidirectional RNNs [18].

Due to the effectiveness of layer normalisation on manual construction, this type of layer can be applied to the final model. As the best-tuned model did not use the dropout, then the dropout could be carried on inside the hidden states. The dropout rate could be tried with a higher value range or dropout with values from 0.5 to 0.8 for hidden layers or input layers as well [22]. However, when used together, normalisation and regularisation should be carefully set, which might conflict [2]. The model is straightforward, with four same-type RNN layers and ends with one Dense layer. More combinations with different setups can be done. The number of layers can be increased to enable the mode to analyse and learn more about the data.

The dataset is purely working on the number and is used to predict only one target. The setup and tuning can become more complex if different datasets are used or a different applied field, like natural language processing. The dataset with the trend is simple; although it has some decline, overall, the dataset trend is gradually increasing. Other stock price histories could have more significant fluctuation. The waver of data will affect the model performance, which might require a more dedicated setup with parameters, layers construction, and argument values.

8. Conclusion

Several experiments have been gone through. These tests include the model construction with three standard types of RNNs - tradition, LSTM, and GRU. Some techniques and layers were used in the experiments, like normalisation, regularisation, gradient clipping, different sequence length trial, and hyperparameter tuning. The best result with the lowest error is the initial GRU model with a max error of 12.25, a min error of 0.0, and a mean error of 1.28. The current optimal model uses the mean squared error loss, Adam optimiser with Keras default learning rate; no dropout or batch normalisation is applied. Through the experiments, the importance of proper structure and parameter values has been exposed. GRU work well in this case; however, in other case or more extended-length data, LSTM will take advantage. Model set-up, choice and configuration are critical to task performance for accuracy, the computing resource economy, and time-saving, as shown in the experiments.

References

[1] Introduction to recurrent neural network, Aug 2022. 3

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 173–182, New York, New York, USA, 20–22 Jun 2016. PMLR. 6

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 4, 6

[4] Y. Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994. 3, 6

[5] Jason Brownlee. How to develop lstm models for time series forecasting, Aug 2020. 6

[6] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. A comparison of LSTM and GRU networks for learning symbolic sequences. *CoRR*, abs/2107.02248, 2021. 6

[7] Tung Lam Dang, Man Dang, Luong Hoang, Lily Nguyen, and Hoang Long Phan. Media coverage and stock price synchronicity. *International Review of Financial Analysis*, 67:101430, 2020. 1

[8] Bob Duckett. Longman business english dictionary (new edition). *Reference Reviews*, 22:26–27, 02 2008. 1

[9] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE, 2016-11. 6

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 1, 2, 6

[11] Thad Hughes and Keir Mierle. Recurrent neural networks for voice activity detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7378–7382, 2013. 1

[12] Simeon Kostadinov. Understanding gru networks - towardsdatascience.com, Dec 2017. 4

[13] Zhangheng Li, Jia-Xing Zhong, Jingjia Huang, Tao Zhang, Thomas H. Li, and Ge Li. ARMIN: towards a more efficient and light-weight recurrent memory network. *CoRR*, abs/1906.12087, 2019. 6

[14] Shuddhasattwa Rafiq. How did house and stock prices respond to different crisis episodes since the 1870s? *Economic Modelling*, 114:105913, 2022. 1

[15] Manu Rastogi. Tutorial on lstms: A computational perspective, Apr 2020. 3

[16] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. 3

[17] Gerald Schneider and Vera E. Troeger. War and the world economy: Stock market reactions to international conflicts. *Journal of Conflict Resolution*, 50(5):623–645, 2006. 1

[18] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 6

[19] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018. 1

[20] Apeksha Nagesh Shewalkar. Comparison of rnn, lstm and gru on speech recognition data, Jan 1970. 6

[21] Sony. Sony group portal. 2

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 6

[23] Chuanwang Sun, Dan Ding, Xingming Fang, Huiming Zhang, and Jianglong Li. How do fossil energy prices affect the stock prices of new energy companies? evidence from divisia energy price index in china's market. *Energy*, 169:637–645, 2019. 1

[24] Keras Team. Simple. flexible. powerful. 1

[25] Siri Team. Deep learning for siri's voice: On-device deep mixture density networks for hybrid unit selection synthesis. 1

[26] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. Chapter 10 - deep learning. In Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal, editors, *Data Mining (Fourth Edition)*, pages 417–466. Morgan Kaufmann, fourth edition edition, 2017. 2

[27] Yahoo. Yahoo finance - stock market live, quotes, business amp; finance news. 2

[28] Yahoo. Sony group corporation (sony) stock price, news, quote amp; history, Nov 2022. 2

[29] Peter T Yamak, Li Yujian, and Pius K Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2019: 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. Association for Computing Machinery. 6

[30] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. pages 98–101, 06 2020. 6

[31] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014. 4

[32] Emir Ž unić, Kemal Korjenić, Kerim Hodžić, and Dženana onko. Application of facebook's prophet algorithm for successful sales forecasting based on real-world data. *International Journal of Computer Science and Information Technology*, 12(2):23–36, apr 2020. 6