ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse383942020.pdf
https://doi.org/10.30534/ijatcse/2020/383942020

Deep Learning with Recursive Neural Network for Temporal Logic Implementation

Ajeet K. Jain[1], Dr.PVRD Prasad Rao [2], Dr. K. Venkatesh Sharma [3]
[1]Research Scholar, Department of Computer Science and Engineering, KoneruLakshmaiah Education Foundation, Vaddeswaram, AP, India; (Association: CSE, KMIT, Hyderabad, India)
[2] Professor, CSE, KLEF, Vaddeswaram, AP, India
[3]Professor, CSE, CVR College of Engineering., Hyderabad, India
[1]jainajeet123@gmail.com, jainajeet1@rediffmail.com,
[2]pvrdprasad@kluniversity.in,[3]venkateshsharma.cse@gmail.com

ABSTRACT

Stock prediction in financial market is one of the exciting applications of deep learning (DL). Stock values fluctuate in accordance with time and hence suitability of Recursive Neural Network (RNN) as one of the model for prediction of stocks is a niche choice as a predictor. The temporal pattern of financial market is investigated with plain RNN and also with Long Short Term Memory (LSTM). In contrast to the classical time series forecasting, the widespread use of deep learning network and algorithms using temporal statistical relations built upon RNN have been finding increasing applications in time series domain analysis and provide a better yield in terms of performance. We have implemented temporal logic using Keras framework and show the results on the sequential data sets of stock prediction market. Our main focus is solely on stock prediction using time series forecasting, however, it can be extended to risk assessment, portfolio management etc. The imperative is to use this as a basis model for further investigation and explore additionally with trend forecasting, crypto currency forecasting and many more. Although this area is matured enough, with machine learning techniques, it provides tremendous opportunities for further investigations and improvements thereon.

Key words: Deep Learning, Long Short Time Memory, Recursive Neural Network, Optimizers, Time Series

1. INTRODUCTION

Time series forecasting is one of AI applications for researchers due to its broad implementation areas and substantial impact in learning the sequential nature. Machine Learning (ML) techniques delved into this area implementing various deep learning models with RNN and significantly outperformed [1, 2]. The time series (TS) analysis comprises methods for analyzing sequential data in order to extract meaningful statistical and other relational characteristics. Further, time series forecasting uses a model to predict future values based on previously observed values. TS encompasses applications in various domains, like statistics, signal processing, pattern recognition, weather forecasting, earthquake prediction, astronomy, stock price, etc.— involving *temporal* measurements. In all such scenarios, the sequential nature imposes an order-preserving on the observations while training the models and making predictions.

1.1 Time Series Model Helps In Stock Prediction

A time series helps to model stock prices correctly, so that stock buyers can rationally decide when to buy and sell them for profitability. To accomplish this, we need learning models that preserves the sequential nature of data and thereby correctly predicting future values.

1.2 Parameterized Network Model Learning

In deep learning (DL), synaptic weights are the parameters of a layer, learnt to find a set of values for all layers in a network to accurately map example inputs to their associated targets. A DL network may contain millions of parameters, so finding the correct

values for all of them may be unnerving task; especially given that modifying the value of one parameter will affect the behaviour of all others! Eventually, it leads to measure how faraway this output is from what we were expecting. Evidently, this is the task of the *loss function* of the system taking the predicted and target values and computing a difference score and thus capturing how well the network has done on this specific example [3]. Typically, these layers are combined into a network (or model) comprising of:

- Input data and corresponding targets
- Loss function (*objective function*), which defines the feedback signal used for learning—the quantity that will be minimized during training. It assesses the success for the task at hand.
- Optimizer, which determines how should learning proceeds based upon loss function. It implements a specific variation of stochastic gradient descent (SGD).

2.RECURRENT NEURAL NETWORKS (RNNS)

A RNN processes by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. In effect, an RNN is a network that has an internal loop as depicted in Fig.1. The state of the RNN is reset between processing two different, independent sequences, so we consider one sequence a single data point: a single input to the network. What changes is that this data point is no longer processed in a single step; rather, the network internally loops over sequence elements.

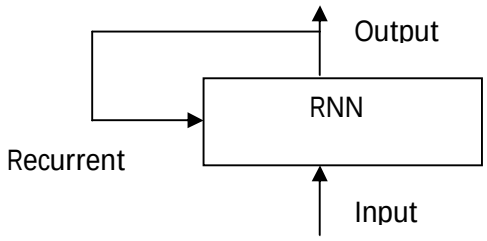


Figure 1: A Simple RNN

The RNN takes a sequence of vectors as input and encodesas a 2D tensor of size **(timesteps, input_features**). It loops over **timesteps**, and at each **timestep**, it considers its current state at **t** and the input at **t** of shape (**input_features**), and combines them to obtain the output at **t**. This in turn sets the state for the next step to be this previous output. For the first timestep, the previous output is undefined; hence, there is no current state. So, we initialize the state as all zero vectors:

```
1     initial_state_t = 0
2     for ts_input_t in input_seq:
3     ts_output_t=(ts_input_t,
initial_state_t)
4     next_state_t =ts_output_t
```

Here in this code fragment, the parameters are indexed with time stamp as:

1 beginning time state—**initial_state_t (**initialized with zero)
2 subsequently looping for the given sequence
3 next time step output is a function of initial state and the input state
4 following state is the output state

The classical RNN's parameters are defined by three weight matrices **U**, **V**, and **W**, corresponding to the input, output, and hidden state respectively, as depicted in Fig. 2.

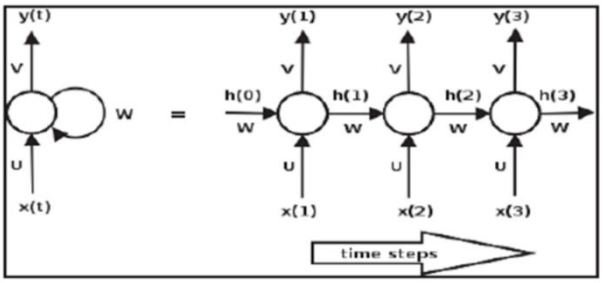


Figure 2: A RNN with a loop

The weight matrices **U**, **V**, and **W**are shared across all the steps,as we are applying the same operation on different inputs at each time step, thereby greatly reduces the number of parameters that the RNN needs to learn. As depicted in Fig.2, the **x**'s are the inputs and **y**'s are the computed outputs (conventionally called as ŷ: y-hat), and the **h**'s hidden values [4, 5, 6].

2.1 ISSUES TO BE DEALT WITH FOR STANDARD RNNS

There are general issues a typical RNN architecture has to deal with:

- Gradient is a partial derivative with respect to its inputs
- Measures how much the output of function changes, if we change the inputs a little bit.
- forhigher gradient values - the faster a model can learn.

- model stops learning when gradient approached zero

2.2 PROBLEM OF LONG-TERM DEPENDENCIES

One of the entreaties of RNN is that they are able to connect previous information to the present task—such as using previous video frames and then understanding of the present frame. However, as the gap grows, RNNs are unable to learn to connect information, as depicted in Fig.3. Here, the $x_1, x_2, \ldots x_n$ represent the input vectors for the A's of the RNN network with corresponding $h_1, h_2, \ldots h_n$ as hidden layers.

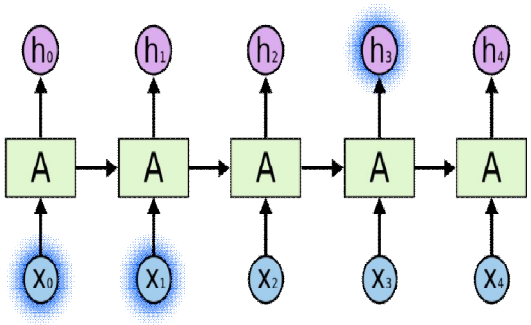
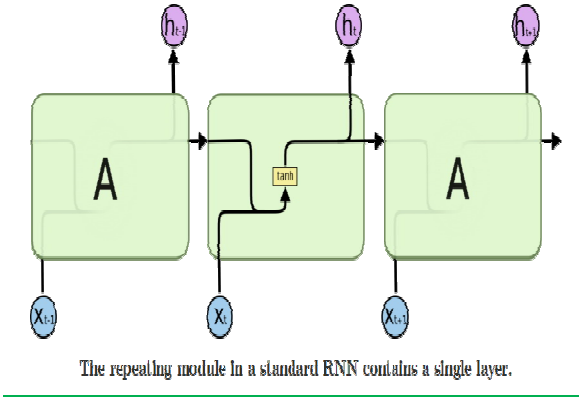


Figure 3: An unrolled RNN

Conceptually, RNNs are capable of handling long-term dependencies. However, unfortunately in practice, RNN internal gating structures need to modify to possess this property. The problem was explored in depth by Hochreiter and Bengio [7, 8,9] and proposed LSTM as problem solver.

2.3 ARCHITECTURE OF LSTM NETWORK

LSTM architecture consists of different memory blocks (rectangles cells) as depicted in Fig. 4. Basically, there are two states that are being transferred to the next cell: the *cell state* and the *hidden state*. The memory blocks store the things and their contents manipulations are done through three gates mechanisms. The symbols and notations used are convectional and standards symbols and blocks of LSTM architecture.



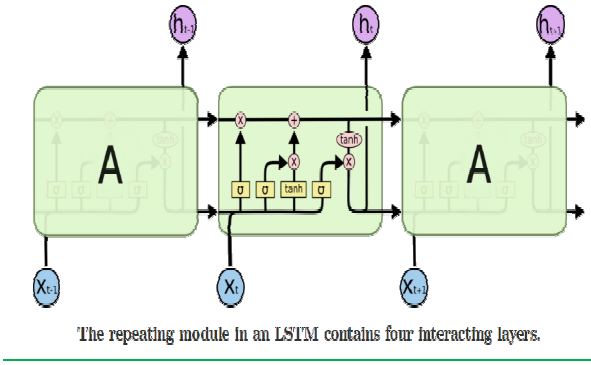


Fig. 4 LSTM architecture [courtesy 8, 9]

memory blocks called cells (rectangles). There are two states that are being transferred to the next cell: the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates.With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies, as:

- previous cell state (i.e. the information that was present in the memory after the previous time step)
- previous hidden state (i.e. this is the same as the output of the previous cell)
- input at the current time step (i.e. the new information that is being fed in at that moment)

These particular features of LSTM make them very attractive to incorporate them into time series models and have been investigated more thoroughly in [8, 9].

3. TIME SERIES IN STOCK MARKET

Stocks are the most important financial instruments where securities of various companies are transacted and itinvolves regulating and controlling the business of buying, selling or dealing with securities. *A security in a financial context is an official document that has a monetary value*. The stock market is characterised as complex, unpredictable and ever changing and moves with a life of its own, reacts to circumstances and leaves investors either reaping rewards or nothing at all. The price of a stock is the input agreed upon by a seller and a buyer in a continuous auction market. The stock price is determined by multiple factors such as supply and demand, opinions and outlooks and technical factors [10].

3.1IMPLEMENTING TIME SERIES USING RNN WITHOUT AND WITH LSTM IN KERAS

(Resource*: httpts://www.kaggle.com/szrlee/stock-time-series -20050101-to-2017123)*

This dataset has stocks time series data of various companies in CSV format and we are using Simple RNN model plus LSTM model using Keras as framework. After the model fitting, we compare the prediction metrics to know which model is the best fit for the prediction.The algorithmic steps are as follows:

1. Load Training data set : Pre-processing Steps use **MinMaxScaler** to normalize the datset for '*Open*', '*Close*', '*High*', '*Low*' values split data into training and test set : first 2500 days for training; last 498 as test set

*2. Simple RNN:*Add dense layers from **Keras** Sequential modelactivation function as **tanh**and **dropout** of **0.15**

*3. Compilation:*Compile with **'ADAM'** optimizer; calculate loss with **MSE**

4. Graph Plot: Plot real and predicted stock price in different colors and analyze

A few typical snapshots of the analysis are as below:

In [12]:

```
rnn_model = Sequential()

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True, input_shape=(X_train.sh
rnn_model.add(Dropout(0.15))

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True))
rnn_model.add(Dropout(0.15))

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=False))
rnn_model.add(Dropout(0.15))

rnn_model.add(Dense(1))

rnn_model.summary()
```

In [13]:

```
rnn_model.compile(optimizer="adam",loss="MSE")
rnn_model.fit(X_train, y_train, epochs=10, batch_size=10)
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tensorflow/py
thon/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
2500/2500 [==============================] - 14s 6ms/step - loss: 0.0949
Epoch 2/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.0190
Epoch 3/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.0100
```

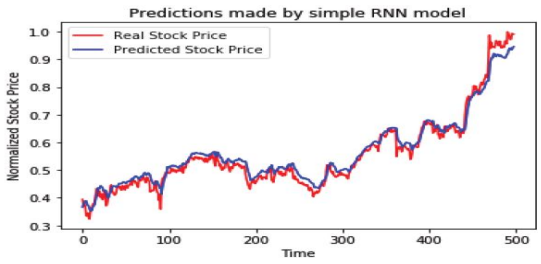
In [14]:

```
rnn_predictions = rnn_model.predict(X_test)

rnn_score = r2_score(y_test,rnn_predictions)
print("R^2 Score of RNN model = ",rnn_score)
```

```
R^2 Score of RNN model =  0.9672359000228702
```

In [15]:

```
def plot_predictions(test, predicted, title):
    plt.plot(test, color='red',label='Real Stock Price')
    plt.plot(predicted, color='blue',label='Predicted Stock Price')
    plt.title(title)
    plt.xlabel('Time')
    plt.ylabel('Normalized Stock Price')
    plt.legend()
    plt.show()

plot_predictions(y_test, rnn_predictions, "Predictions made by simple RNN model")
```



4. CONCLUSSION

In this work, we have presented the applicability of RNN with LSTM network for stock prediction and were able to closely follow the real and predicted stock prices as shown. The fluctuating financial market has been studied by many researchers with classical methods over the past several years. However, our work suggests as to how the RNN models with deep learning techniques can be

effectively utilized. With this intuitiveness in mind, we are able to achieve an accuracy of 96.7 % which evidently shows good-fit results. With the present set up using **Keras**framework augmented with ADAM optimizer and parameters tuning, we are able to get these accurate results. Furthermore by incorporating other optimizers like **RMSProp** and AdaDelta, , the scope of the work can further be extended for knowing *High Frequency Trading* (HFT), *Intraday price movement* (trend), *weekly/monthly closing prices* and likewise. Additionally, other areas of extension are: risk management and portfolio management [10], commodity (oil, gas) price prediction, bond price forecasting, volatility forecasting, crypto currency forecasting, etc. Hence, various models of DL can provide better understanding of the time series forecasting. Moreover, the, CNN approach- like **convnet1D** as a pre-processing can be implemented in order to improve processing speed[11,12,13,14].

The proposed framework can further be expanded to investigate:

- Which other DL models can provide better time series forecasting?
- Whichoptimizer one would perform better for a given dataset?
- application of Gated Recurrent Unit (GRU) for similar purpose as a proliferation of RNN

REFERENCES

[1] Hearty John, Advanced Machine Learning with Python, Packt Publishing, 1st ed. (2016)

[2] http://stanford.edu/~jduchi/projects/DuchiHaSi 10 _colt.pdf

[3] Francois Chollet, Deep Learning with Pythons, Manning Pub.1st ed. (2016)

[4] R. Jozefowicz, W. Zaremba, and I. Sutskever, JMLR, LSTM: A Search Space.(2015)

[5] R. Pascanu, T. Mikolov, and Y.Bengio, Difficulty of Training Recurrent Neural Networks, ICML, pp 1310-1318. (2013)

[6] R. Jozefowicz, W. Zaremba, and I. Sutskever, An Empirical Exploration of Recurrent Network Architectures JMLR.(2015)

[7] YoshuaBengio, Patrice Simard, and Paolo Frasconi, "Learning Long-Term Dependencies with Gradient Descent Is Difficult," *IEEE Transactions on Neural Networks* 5, no. 2. (1994).

[8] SeppHochreiter and Jürgen Schmidhuber, "Long Short- Term Memory," Neural Computation 9, no. 8, 1997

[9] ChristopherOlah's : Understanding LSTMs ;*https://colas.github.io/post/2015-08-Understaing LSTM*

[10] A. Bernal, S. Fok, and R. Pidaparthi, Financial Market Time Series Prediction with Recurrent Neural Networks. (2012)

[11] J. Chung, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelling, ArXiv: 1412.3555. (2014)

[12] P.V.R.D PrasadaRao, Yasin, "A framework for decision making and quality improvement by data aggregation techniques on private hospitals data" Journal of Engineering and Applied Sciences Open Access,Volume 13, Issue 14, 1 July 2018, Pages 4337-4345

[13]Jain, A.K.,Rao, P.V.R.D.P.,Sharma, K.V., Extending description logics for semantic web ontology implementation domains, Test Engineering and Management 83 ,pp.7385

[14] Jain,N., Jain,AK., PrasadRao, P.V.R.D., Venkatesh Sharma, K. , Conglomerating first order, descriptive and modal logics into semantic web – A research,International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue- 6S4, April 2019