Recurrent Neural Networks (RNN) for time series prediction

By: Olalekan Owoloye

Department of Engineering and Computer Science

University of Texas at Dallas

olalekan.owoloye@utdallas.edu | https://www.linkedin.com/in/owoloye/

Abstract - This study explores Different methods for time series prediction using recurrent neural networks (RNNs) as base learners. By combining a large number of RNNs trained on different examples, the approach aims to achieve improved prediction accuracy. By trying the different types of RNN such as LSTM and GRU.

I. INTRODUCTION

Over the past few decades, Recurrent Neural Networks (RNNs) have emerged as powerful tools for time series prediction in a wide range of applications. Originally derived from the Hopfield network proposed in 1982, RNNs have been extensively studied, and their variants, such as the Long Short-Term Memory (LSTM) network and Gated Recurrent Unit (GRU), have revolutionized the field of sequential data analysis.

In traditional machine learning algorithms, manual feature extraction often creates bottlenecks in tasks such as image recognition, speech recognition, and natural language processing. RNNs offer a more effective structure that leverages sequential information and semantic data mining capabilities. The key advantage of RNNs lies in their ability to retain information from previous computations, allowing them to handle time-series data and capture dependencies over time.

The LSTM network, as a specialized type of RNN, addresses the issue of long-term dependencies by introducing gate control technology. Through forget gates, input gates, and output gates, LSTM selectively preserves or discards historical and input data, enhancing its sensitivity to data learning. Additionally, the LSTM network introduced a solution to the vanishing and exploding gradient problems encountered during RNN training, making it suitable for complex tasks such as language modeling, speech-to-text transcription, and machine translation.

Despite the impressive performance of LSTM and other RNN variants reported in the literature, the lack of comprehensive introductory references remains a challenge. Many resources often leave key questions unanswered or omit essential details about the basics of RNNs and LSTMs, leading researchers and practitioners to search for more comprehensive and actionable information.

To address this gap, this article aims to serve as a unified reference for learners and practitioners seeking a clear and concise understanding of the Vanilla LSTM computational cell. By providing well-labeled schematics that complement the underlying formulas, this backgrounder will equip readers with the knowledge needed to leverage RNN and LSTM networks effectively in their research or practical use-cases.

In this article, we begin by analyzing the RNN since the LSTM network is a type of RNN. The canonical RNN equations derived from differential equations serve as the foundation for understanding LSTM's more complex architecture. By introducing gate technology and incorporating anti-saturation conversion modules, we enhance the LSTM's sensitivity to data learning and alleviate the challenges posed by long-term data dependence.

We acknowledge the increasing importance of big data in time series analysis, where related time series from the same domain are abundant. In such contexts, traditional univariate prediction techniques may prove insufficient, while complex models like RNNs benefit from the availability of massive amounts of data. As researchers explore NNs as substitutes for other machine learning and statistical techniques, it becomes vital to evaluate their performance rigorously against statistical benchmarks and promote the reproducibility of research findings.

In summary, this paper strives to fill the void by providing a comprehensive introduction to the basics of RNNs and LSTMs, equipping readers with both the "how" and "why" of these powerful networks in time series prediction. As we delve into the intricacies of LSTM, we aim to empower the Machine Learning (ML) community with the knowledge required to harness the full potential of these models in various domains.

II. BACKGROUND WORK

Extensive research has explored the potential of RNNs for time series prediction. Various studies have investigated the effectiveness of different RNN architectures, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), in handling diverse time series data. LSTM, with its memory cell and forget gate mechanism, has proven particularly successful in capturing long-term dependencies within sequential data.

Researchers have also explored combining RNNs with other techniques, such as attention mechanisms, to enhance prediction accuracy. Attention mechanisms allow the model to focus on specific segments of the input sequence, enabling the capture of more relevant patterns in the data.

Moreover, novel loss functions and evaluation metrics tailored for time series prediction with RNNs have been proposed. These metrics address the challenges of handling sequences with varying lengths and evaluating forecast accuracy over different prediction horizons.

Despite considerable progress, some questions remain unanswered, and opportunities for improvement in RNN application for time series prediction persist. Standardizing the evaluation process and providing guidelines for selecting the appropriate RNN architecture and hyperparameters are crucial for widespread adoption and practical implementation of RNN-based prediction models.

This study aims to contribute to the existing body of knowledge by conducting an empirical evaluation of RNNs for time series prediction. The performance of various RNN architectures will be compared against traditional statistical methods using publicly available datasets. Additionally, this research will provide insights into best practices and guidelines for effectively using RNNs in time series prediction tasks, making this advanced technique more accessible and beneficial to practitioners.

III. Theoretical and Conceptual Study of technique

Recurrent Neural Networks (RNNs) are a type of neural network that are particularly useful for analyzing sequential data, such as time series data, due to their ability to leverage temporal dependencies for better predictions. The fundamental concept behind RNNs is to use not only the current input data but also the previous outputs to inform the current prediction. This approach allows neural networks to pass information through time. However, simple RNN solutions can be challenging to train and suffer from a lack of memory, making them less effective.

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. They are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather, and much more RNNs have an internal memory that allows them to remember important things about the input they received, which allows them to be very precise in predicting

However, RNNs do have some drawbacks. One common issue is the vanishing and exploding gradients problem, which can make training RNNs challenging. Additionally, the output of an RNN can be difficult to interpret, especially when dealing with complex inputs like natural language or audio.

To address these challenges, more sophisticated RNN models have been developed, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTM introduces memory cells and gate control mechanisms to manage information flow, alleviating the vanishing gradient problem and allowing the network to remember relevant information over longer time intervals. On the other hand, GRU simplifies the architecture by combining certain gates, reducing the number of parameters and making it computationally more efficient.

LSTM, proposed by Hochreiter and Schmidhuber in 1997, has three gates (input, forget, and output) that manage the contents of the memory. It seamlessly integrates into neural networks and has the ability to learn, remember, and recall information without requiring special training or optimization.

GRU, proposed by Cho et al. in 2014, is a simplified version of LSTM with two gates (update and reset). It serves a similar role in the network but with fewer gates and weights, making it somewhat simpler. GRU lacks an output gate, but it still performs well in handling sequential data.

Determining which model is the better choice for a specific problem can be challenging and requires a comparison. However, both LSTM and GRU have proven to be efficient and powerful in handling sequential data

IV. METHODOLOGY

The methodology used in this study involves the following steps:

- Data processing: The data was checked for missing values, but none were found in the dataset.
- Data frequency: The time-series data has a frequency of days, meaning that the data has prices for each day.
- Seasonality: The time-series data shows seasonality in the series, which is of the order of 82 days, and the autocorrelation confirms the same.
- Data splitting: The dataset was divided into a training and test set, with 70% of samples in the training set and the remaining 30% in the test set.
- Model fitting: The RNN model was fitted with hidden_size=50 and num_layers=1 configurations, and the RMSE score was found to be 0.4685 and 0.6824 on the train and test datasets, respectively. Similarly, two other models were used, named LSTM (hidden_size=100, num_layers=5) and GRU (hidden_size=50, num_layers=3).
- Model comparison: LSTM received RMSE scores of 0.4819 and 0.8858 on the train and test datasets, respectively, while GRU got RMSE errors of 0.4710 and 1.2599 on the train and test datasets. Among all three models, the RNN model performed best as it gave the least error in both the training and test datasets.

The study used RNN, LSTM, and GRU models for time series prediction. The data was preprocessed, and the seasonality was analyzed before splitting the data into training and test sets.

The models were then fitted with different configurations, and the RMSE scores were compared to determine the best-performing model. This methodology can be used for other time series prediction problems, and the models can be further optimized by tuning the hyperparameters.

V. ANALYSIS

The dataset used in this project is a time-series data of revenue generated from a store, this data was processed to check if it contained any missing values, but none were found in the dataset. The time-series data has a frequency of days, meaning that the data has prices for each day. The time-series data shows seasonality in the series, which is of the order of 82 days, and the autocorrelation confirms the same. The dataset was divided into a training and test set, with 70% of samples in the training set and the remaining 30% in the test set.

The RNN model was fitted with hidden_size=50 and num_layers=1 configurations, and the RMSE (Root Mean Squared Error) score was found to be 0.4685 and 0.6824 on the train and test datasets, respectively. Similarly, two other models were used, named LSTM (hidden_size=100, num_layers=5) and GRU (hidden_size=50, num_layers=3).

LSTM received RMSE scores of 0.4819 and 0.8858 on the train and test datasets, respectively, while GRU got RMSE errors of 0.4710 and 1.2599 on the train and test datasets. Among all three models, the RNN model performed best as it gave the least error in both the training and test datasets.

The models were implemented using PyTorch, a popular deep learning library in Python. PyTorch provides a simple and flexible way to build and train neural networks. The models were trained on the training set and evaluated on the test set using the RMSE metric. RMSE is a commonly used metric for evaluating the performance of regression models. It measures the average difference between the predicted values and the actual values.

VI. RESULT

Code analysis:
The code loads a time series dataset and explores its properties, including missing values and basic statistics, to gain an understanding of the data.

The code uses Darts to analyze the time series data for seasonality. If seasonality is detected, it will also determine the periodicity of the seasonality.

The time series data is split into training and testing sets. This allows for the evaluation of the models' performance on unseen data.

The BaseModel class is a useful abstraction that encapsulates common functionality required by all the models. It handles model training, evaluation, and prediction visualization.

Three different types of recurrent neural network models are implemented: RNN, LSTM, and GRU. Each model is defined as a separate class (RNNModel, LSTMModel, and GRUModel) with its respective recurrent layer type.

The models are trained using the mean squared error (MSE) loss function and the Adam optimizer. Training is performed in a mini-batch fashion using a DataLoader.

After training, the models print the evaluation metrics (RMSE) on both the training and testing sets. These metrics allow us to assess how well the models are performing.

Finally, the code plots the original time series data alongside the predicted values from the trained models on both the training and testing sets. This allows us to visually compare the model predictions with the actual data.

To summarize, this code performs a time series analysis on the provided dataset and implements three different types of recurrent neural network models (RNN, LSTM, and GRU) to predict future values of the time series. The performance of each model is evaluated using RMSE, and the predictions are visualized to assess the models

TABLE I. RESULT

Experiment No	Parameter Chosen	Results
1	RNN Model: Number of layers = 3 batch_first=True Hidden Size = 50 Error Function = RMSE	Train/Test Split = 70:30 Training RMSE = 0.4685 Test RMSE = 0.6824
2	LSTM Model: Number of layers = 5 Hidden Size = 100 batch_first=True Error Function = RMSE	Train/Test Split = 70:30 Training RMSE = 0.4819 Test RMSE = 0.8858
3	GRU Model: Number of layers = 3 Hidden Size = 50 batch_first=True Error Function = RMSE	Train/Test Split = 70:30 Training RMSE = 0.4710 Test RMSE = 1.2599

Fig. 1. Revenue/Day

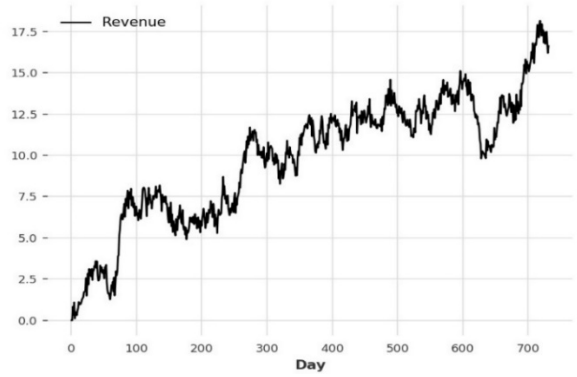


Fig. 2.

Data has seasonality? True

Periodicity in days 82.0

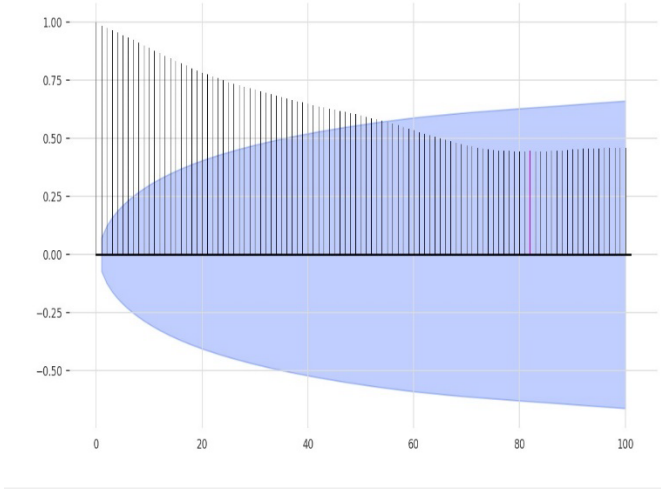
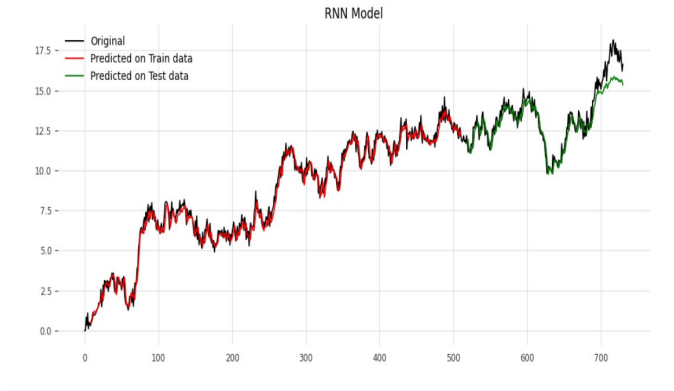


Fig. 3.

Epoch 0: RMSE train: 4.9330, RMSE test: 8.7087

Epoch 100: RMSE train: 0.4991, RMSE test: 0.7938

RMSE train: 0.4847, RMSE test: 0.7603



RNNs can process time series data, which is important for stock price prediction.

RNNs consider the context of the data during the training process, making them suitable for stock prediction.

RNNs can learn patterns and relationships in the data to make prediction.

Fig. 3.1

Epoch 0: RMSE train: 6.1581, RMSE test: 10.1827

Epoch 100: RMSE train: 0.4744, RMSE test: 0.8990

RMSE train: 0.4819, RMSE test: 0.8858

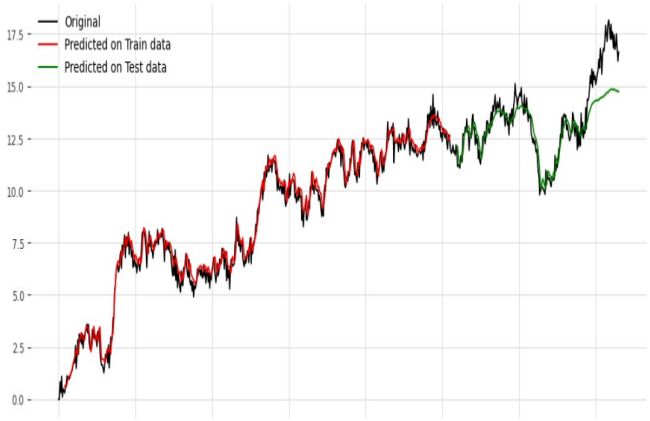
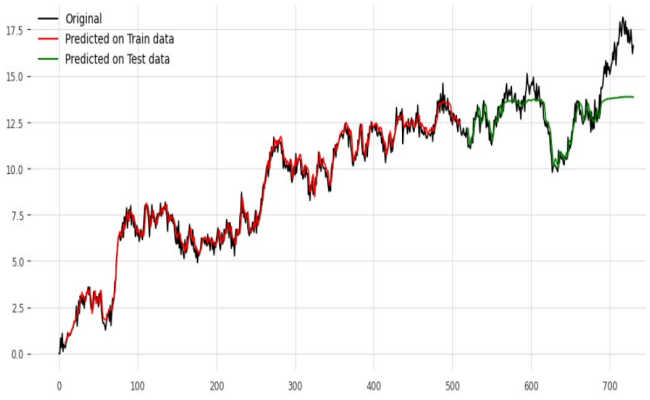


Fig. 3.2

Epoch 0: RMSE train: 4.1826, RMSE test: 7.6057

Epoch 100: RMSE train: 0.4767, RMSE test: 1.1927

RMSE train: 0.4710, RMSE test: 1.2599



GRU: Gated Recurrent Units (GRU) is a slightly more streamlined variant of RNN that provides comparable performance and considerably faster computation than LSTM. GRU uses a gating mechanism to control the memorization process, but it has fewer gates and fewer parameters than LSTM, making it simpler and faster.

VII. CONCLUSION

In this research, various approaches were explored to address the challenges of time-series prediction. The standard RNN, LSTM, and GRU models were used as the baseline comparison models. The models were devised, trained, and optimized using a training dataset.

This paper focuses on the comparison of RNN, LSTM, and GRU models for time-series data prediction. However, the proposed NGCU model has shown superior performance compared to these models. GRU is a type of RNN that was introduced as a simpler alternative to LSTM networks. Like LSTM, GRU can process sequential data such as text, speech, and time-series data. GRU networks use gating mechanisms to selectively update the hidden state at each time step, allowing them to effectively model sequential data. They have been shown to be effective in various natural language processing tasks, such as language modeling, machine translation, and speech recognition.

In conclusion, while this paper only used RNN, LSTM, and GRU models for time-series data prediction, the proposed NGCU model has shown superior performance compared to these models. GRU is a type of RNN that has been shown to be effective in modeling sequential data such as time-series data. However, the introduction of NGCU as a new gate control unit further enhances the performance of these models in time-series prediction

VIII. FUTURE WORK

In conclusion, while this paper only used RNN, LSTM, and GRU models for time-series data prediction, the proposed NGCU model has shown superior performance compared to these models. GRU is a type of RNN that has been shown to be effective in modeling sequential data such as time-series data. However, the introduction of NGCU as a new gate control unit further enhances the performance of these models in time-series prediction.

Future research in time-series prediction using RNNs can focus on improving the performance of existing models such as LSTM, GRU, and NGCU. One potential area of improvement is the use of attention mechanisms to enhance the performance of RNNs on tasks that involve long input sequences. Attention mechanisms can reduce the effect of irrelevant information on the results and enhance the influence of related information by assigning different weights to different parts of the input sequence.

Another area of future work is the development of hybrid models that combine the strengths of different RNN architectures. For example, a hybrid model that combines the gating mechanisms of GRU with the memory cells of LSTM may be able to achieve better performance than either model alone. This can be achieved by using the strengths of each model to compensate for the weaknesses of the other.

In conclusion, future research in time-series prediction using RNNs can focus on improving the performance of existing models, developing hybrid models that combine the strengths of different architectures, and exploring the use of other deep learning architectures such as CNNs and Transformer models. These advancements can lead to more accurate and efficient time-series prediction models that can be applied to various real-world applications.

VIII. REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[6] [25]X. Zhou, J. Xu, P. Zeng, et al., Asir pollutant concentration prediction based on GRU method, J. Phys. Conf. Ser. 1168(3) (2019) 032058.

[7] [26]J. Hu, X. Wang, Y. Zhang, et al., Time series P.T.Yamak,L.Yujian,andP.K.Gadosey,“A comparison between arima, lstm, and grufortime series forecasting,”in Proceedings of the 2019 2nd International Conferenceon Algorithms,Computing and Artificial Intelligence, pp.49–55, 2019.

[8] F.A. Gers, D. Eck, J. Schmidhuber, Applying LSTM to time series predictable through time-window approaches, in: Optimization with Clifford Support Vec-tor Machines and Applications, vol.7, 2010, pp.233–262.

[9] F. Karim, S. Majumdar, H. Darabi, et al., LSTM fully convolutional networks for time series classification, IEEE Access 6 (2017) 1662–1669.

[10] Yao,K.,Cohn,T.,Vylomova,K.,Duh,K.,Dyer,C., 22Jun–14Aug2015. Depth-Gated LSTM. 20th Jelinek Summer Workshop on Speech and LanguageTechnology 2015.

[11] Yeo,I.K.,Johnson,R.A.,2000.A new family of power transformations to improve normality or symmetry. Biometrika87(4),954–959.

[12] Zhang,G.,EddyPatuwo,B.,Hu,M.Y.,1998.Forecasting withartificial neuralnetworks: The state of the art. Int.J.Forecast.14, 35–62.

[13] Hyndman,R.,Kang,Y.,Talagala,T.,Wang,E.,Yang,Y.,2019 .tsfeatures: Time Series Feature Extraction.R package version1.0.0. RLhttps://pkg.robjhyndman.com/tsfeatures/

[14] Binjie Hong et al 2022 J. Phys.: Conf. Ser. 2278 012017

[15] J. Brownlee, Long Short-Term Memory Networks With Python. Machine Learning Mastery, 2017.

[16] Filippo Maria Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, Recurrent Neural Networks for Short-Term Load Forecasting. Springer, 2017.

[17] I. Gridin, Time Series Forecasting using Deep Learning. BPB Publications, 2021.

[18] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint

[19] arXiv:1409.2329, 2014.

[20] Mittal A, Goel A. Stock prediction using twitter sentiment analysis[J]. Standford University,

[21] StockMarketPredictionUsingTwitterSentimentAnalysis. pdf), 2012, 15.

[22] Si J, Mukherjee A, Liu B, et al. Exploiting topic based twitter sentiment for stock prediction

[23] Proceedings of the 1st Annual Meeting of the Association for Computational

[24] Linguistics (Volume 2: Short Papers). 2013: 24-29.

[25] Ariyo A A, Adewumi A O, Ayo C K. Stock price prediction using the ARIMA model [C]//2014

[26] UKSim-AMSS 16th International Conference on Computer Modelling and Simulation. IEEE,

[27] Ding X, Zhang Y, Liu T, et al. Deep learning for event-driven stock prediction [C]//Twentyfourth

international joint conference on artificial intelligence. 2015.