HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 2 (2024)

A Novel Model to Minimize Time Series Forecasting Error Using Convolutional Attention

Pranav Pathak[1] and Daniel Plymire[#]

[1]Westmount Charter School, Canada
[#]Advisor

ABSTRACT

The main consideration of this paper is time series forecasting using machine learning methods. While hundreds of studies have been conducted on this topic, we propose a novel, yet intuitive model, using technical indicators in combination with a focus on lagged relationships and the machine learning technique called BEDCA (Belief Encoder-Decoder with Convolutional Attention). BEDCA takes into consideration technical indicators to improve predictive accuracy. To evaluate our model, we used historical Coca-Cola stock data, room temperature data, and daily data including the number of COVID-19 case data. We found that our model trained relatively quickly and had a high accuracy (low loss). Overall, BEDCA outperformed the standard Long-Short Term Memory (LSTM) network on all three datasets, achieving MAE reduction of 74.1%, 41.3%, and 69.0%, respectively, although it took 3.05, 3.26, and 3.20 times longer to train on each dataset respectively. Our results highlight that convolutional attention is a promising form of attention and that technical indicators are important considerations for time series data.

Introduction

A time series variable is a quantitative value that changes over time. For example, the average temperature in a city is a time series variable. Time series forecasting is the task of using statistical and/or machine learning models to predict the future values of a time series variable. This task has applications in almost every industry, ranging from business to healthcare (Tableau, n.d.).

Two popular neural network models for time series forecasting are the Recurrent Neural Network (RNN) and the Long-Short Term Memory (LSTM). A RNN is a series of cells which pass on time context in the form of a hidden state (Figure 1). It works similar to a deep neural network, with the exception that inputs are with respect to time and that the weights are the same for all layers (IBM, n.d.). RNNs are less widely used due to the vanishing and exploding gradient problems, making them quite difficult to train (Sharkawy, 2020). The following figure shows a diagram of a RNN.

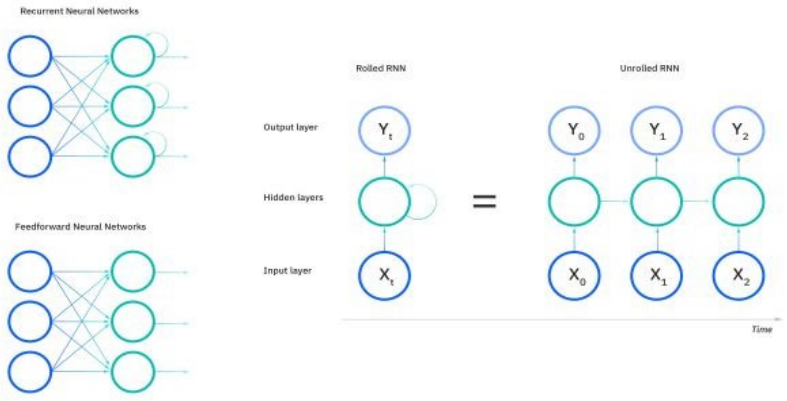

Figure 1. The structure of a neural network as compared to a RNN (IBM, n.d.).

The RNN struggles to capture long range relationships in time series data. A LSTM is a modified RNN that uses cells, made up of gates, to selectively learn and forget information from the data (IBM, n.d.). In this way, the LSTM can better retain information over long sequences, and make better predictions.

Technical indicators are specific values that are calculated from numerical data to describe a relationship in the data. For example, the Simple Moving Average 20 (SMA 20) calculates the average of the 20 most recent data points.

Previous studies have used machine learning models in combination with technical indicators tend to employ LSTMs or CNNs (Li & Bastos, 2020). Additionally, studies that use a combination of models tend to use each model for a different purpose, such as one model for feature extraction and another for prediction. Furthermore, different papers proposed different ways of integrating technical indicators into models. One paper trained a separate LSTM for each of 3 technical indicators (RSI, MACD, and SMA) (Sang & Pierro, 2019). However, we believe that including multiple technical indicators in the same model can improve model accuracy. Another paper normalized five technical indicators and then trained a BiLSTM with attention (Lee et al., 2022). The BiLSTM's predictions were used in a simple stock trading strategy to decide whether to buy or sell a stock. This paper achieved a maximum accuracy of 68.83% for stock trend prediction. Due to the moderate accuracy achieved in this paper, we decided to use attention in our model as well. Lastly, another paper introduced a novel "Time Neural Network" which uses a kernel filter and a time attention mechanism to achieve low loss (Zhang et al, 2023). The kernel filter acts as a convolution and was used to better represent the input data, while the time attention mechanism used a separate neural network to generate attention weights. This paper tested the time neural network against other models and found that the time neural network worked best. As Zhang et al. (2023) discusses, a limitation of this study was to look at lagged relationships, which we improve in our model. Furthermore, we believe that attention weights should be generated within the model, such that the total back-propagation complexity should decrease.

Although most previous research combining machine learning models and technical indicators for time series forecasting focuses on applications for stock price prediction, we believe that this method of forecasting can be generalized for all time series forecasting tasks. We believe this due to the promising results from previous studies, which suggest that a well developed model could learn complex relationships in time series data.

As such, this research aims to accomplish three main goals:

1. Achieve a Mean Absolute Error (MAE) of less than 0.2σ on the testing data.
2. Reduce training time on relatively large datasets.
3. Create a model that does not require more than 5000 training examples to achieve the required predictive accuracy, which also uses technical indicators.

Methods

Datasets

Three datasets, collected from Kaggle, were used for training and testing the model. The first dataset was a Coca-Cola stock price dataset that contained stock prices from 1962-present (Rahman, 2023). The second dataset was a dataset of room temperatures collected from an IOT device (Madane, 2023). The third dataset was a dataset of daily confirmed COVID cases in Kerala, India from January 31st 2020 to May 22nd 2022 (Anandhu, 2022).

These three datasets were chosen to evaluate two main aspects of the model, its ability to predict a volatile or non-volatile time series variable and its ability to predict a cyclic time series variable, two common patterns that we expect a good model to be able to learn. Thus, the first dataset was used to test if the model could learn a relatively stable growth pattern, the second dataset was used to test if the model could learn cyclic patterns, and the third dataset was used to test if the model could learn a combination of both.

Methods

Data collection procedures:

1. Data was collected from the respective Kaggle datasets.
2. Data was cleaned. Missing values and NaNs (not a number) will be replaced by the mean of the previous and next valid values.
3. Technical indicators were calculated for the time series data. Technical indicators were calculated using the TA library in python.

The following technical indicators were chosen:

1. Kaufman's Adaptive Moving Average (KAMA): This technical indicator is used to deal with noise in time series data and can help identify the general trend (Padial, 2022).
2. Bollinger band high and low: This technical indicator is used to identify the value range (high and low) of a time series with a specified number of standard deviations from the moving average (Hayes, 2023). This indicator was used to help the model to detect unusual movements.
3. Aroon: This technical indicator is used to predict when the stock movement is likely to reverse (Padial, 2022). This indicator was included to teach the model to learn possible trend reversal signs.
4. Moving Average Convergence Divergence (MACD): This indicator is the difference between two moving averages with different periods (Padial, 2022). It is used to identify possible signs of trends. This indicator was used to teach the model to follow trends.
5. Triple Exponential Average (TRIX): This indicator calculates the change of a triple exponentially smoothed moving average (Chen, 2022). This indicator helps the model focus on important patterns and to predict the correct magnitude of increase or decrease.

The first 43 rows will be removed since the technical indicator calculations will cause these values to become undefined. Note: the number 43 was chosen, because the TRIX indicator is undefined until row 43. Then, the inputs were normalized by calculating the z-score for each of the 7 technical indicators. Furthermore, the inputs were batched into sequences of length 50, with 7 technical indicators for each training example. In a similar way, the testing dataset was created.

For the BEDCA inputs, the difference between each two consecutive z-scores was calculated and used

as the input and testing data. This was done to ensure that the BEDCA captured the relationships better.

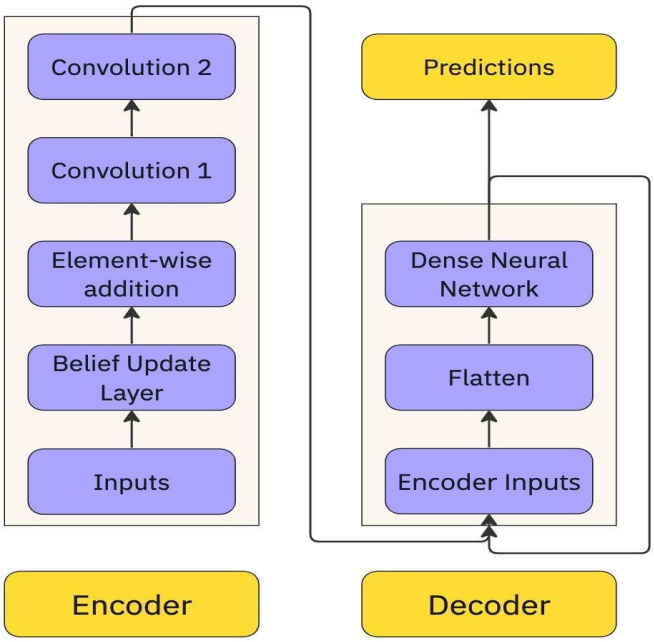

Figure 2. Architecture of the BEDCA model.

An overview of the model is shown in Figure 2.

The forward propagation of the model works as follows: The input is split into training examples of shape (50, 7). This means that there are 50 time series values of 6 technical indicators and the actual time series data. For each training example, the next two values are the y values (outputs). This means that the aim of the model is to predict the next two values given a set of 50 preceding values.

Each technical indicator (including the actual time series data) in each training example, X_i, is input to a belief update layer. A belief update layer takes a matrix X of size 1×50 and calculates $x^T \cdot x$, resulting in a 50×50 matrix. This belief matrix captures lagged relationships in the input. Then, a convolution is performed with 8 filters, a kernel size of 4, and a stride size of 2 (no bias). This convolution acts as an attention mechanism, with each filter acting as a query to learn a separate relation in the belief matrix. Thus, no bias was used. These operations combine to form the belief update layer for a single technical indicator (including the actual time series data).

A belief update layer was applied to each of the 7 technical indicators separately. Element-wise addition was performed on the outputs for each of the technical indicators to obtain a tensor with shape (24, 24, 8).

On this, a convolution was performed with 16 filters, a kernel size of 3, and a stride size of 1. Subsequently, another convolution was performed on the output of the previous convolution. This convolution layer has 32 filters, a kernel size of 2, and a stride size of 2. Finally, max pooling was applied on the result, giving an Encoder representation of the input sequence. The above operations form the Encoder block.

The output of the Encoder was passed to the Decoder. The Decoder flattens its input and feeds it into a DNN (Dense Neural Network) with 1024, 256, 128, 32, 4, and 2 output units in its six layers. The relu, tanh, elu, and linear activation functions were used. The final two outputs of the DNN were the predicted outputs for the next two time steps (still normalized and differenced).

The loss function, which the model tries to minimize, is defined as:

$$L(y_1, y_2, \widehat{y_1}, \widehat{y_2}) = (y_1 - \widehat{y_1})^2 + 0.5 \times (y_2 - \widehat{y_2})^2$$

This loss function ensures that the model is also trying to optimize its predictions for the long term, which is more desirable.

Results

A table comparing the MAE of the LSTM and the BEDCA for each of the three datasets used is provided in Table 1.

Table 1. The MAE of both models for the three testing datasets.

	LSTM MAE	Belief Encoder-Decoder MAE	% Reduction
Stock prices	2.86	0.74	74.13
Temperatures	2.54	1.49	41.34
COVID cases	5806.73	1799.50	69.01

Stock prices and temperature data generally had low values and low variance compared to the COVID cases, which had high values and a much higher variance. Generally, stock and temperatures were quite low as they were measured in dollars and degrees celsius, while COVID cases were measured for a whole state during peak COVID time periods, causing them to be much higher.

Overall, the LSTM contains 80 parameters and the BEDCA contains 2,984,574 parameters.

Coca-Cola Stock Data

The mean and standard deviation of the testing data were 36.16 and 11.54 respectively. The LSTM was trained for 150 epochs, totaling 1 hour and 3 minutes on a NVIDIA L4 GPU, an average of 0.42 minutes/epoch. A plot of the predicted vs actual stock prices is shown below (Figure 3).

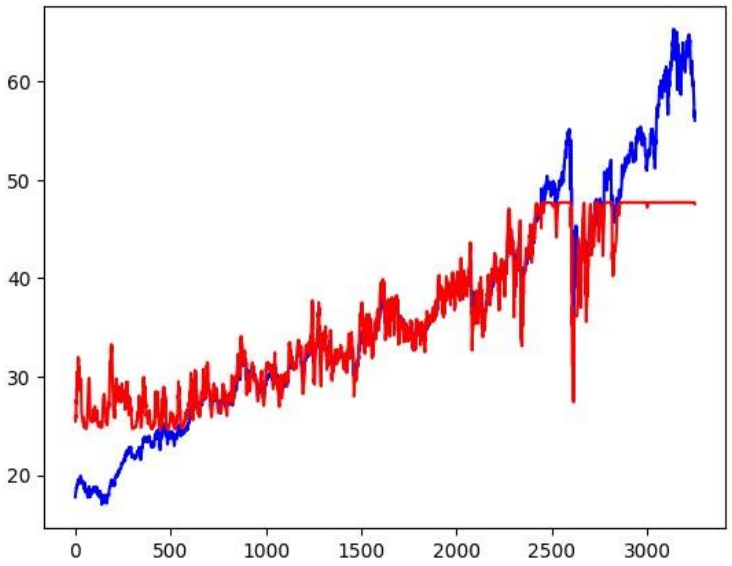

Figure 3. A plot comparing the predicted (red) and actual (blue) stock prices for the test data of 3250 examples using LSTM.

The BEDCA was trained for 75 epochs, taking 1 hour and 36 minutes on a NVIDIA L4 GPU, an average of 1.28 minutes/epoch. A plot of the predicted vs actual stock prices is shown below (Figure 4).

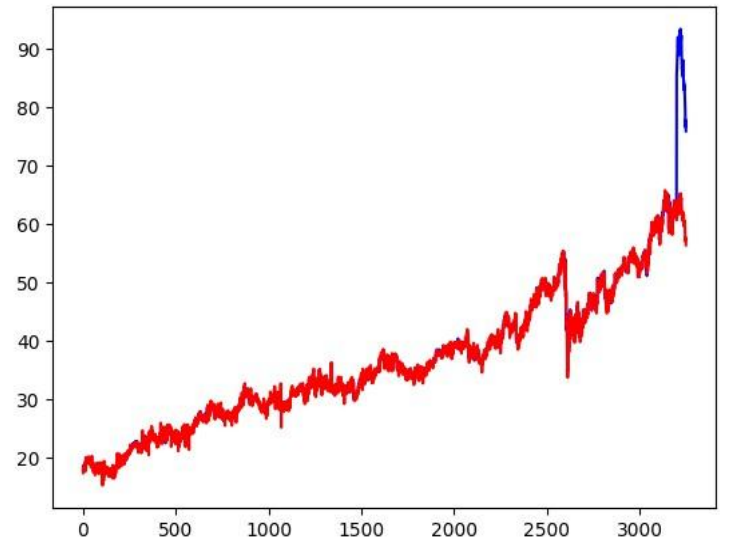

Figure 4. A plot comparing the predicted (red) and actual (blue) stock prices for the test data of 3250 examples using BEDCA.

Temperature Data

The mean and standard deviation of the testing data were 22.50 and 6.75 respectively. The LSTM was trained for 150 epochs, taking 57 minutes on a NVIDIA L4 GPU, an average of 0.38 minutes/epoch. A plot of the predicted vs actual temperature is shown below (Figure 5).

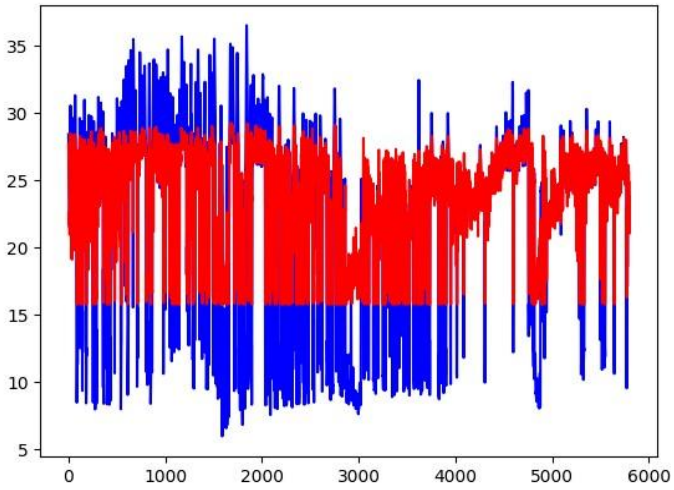

Figure 5. A plot comparing the predicted (red) and actual (blue) temperatures for the test data of 5800 examples using LSTM.

The BEDCA was trained for 75 epochs, totaling 1 hour and 33 minutes on a NVIDIA L4 GPU, an average of 1.24 minutes/epoch. A plot of the predicted vs actual temperatures is shown below (Figure 6).

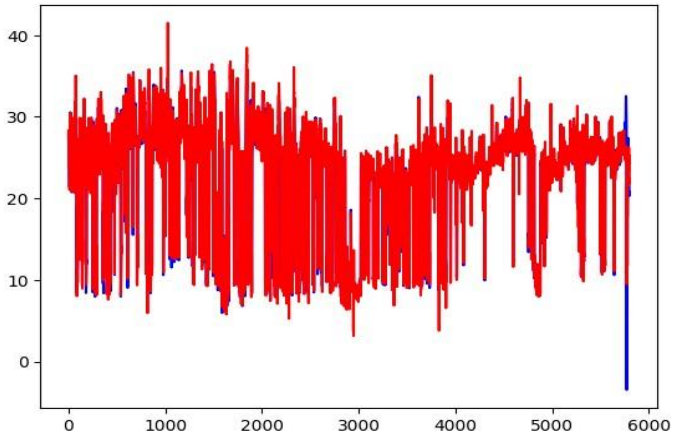

Figure 6. A plot comparing the predicted (red) and actual (blue) temperatures for the test data of 5800 examples using BEDCA.

COVID Data

The mean and standard deviation of the testing data were 10962.84 and 12347.50 respectively. The LSTM was trained for 150 epochs, taking 23 minutes on a NVIDIA L4 GPU, an average of 0.15 minutes/epoch. A plot of the predicted vs actual number of COVID cases is shown below (Figure 7).

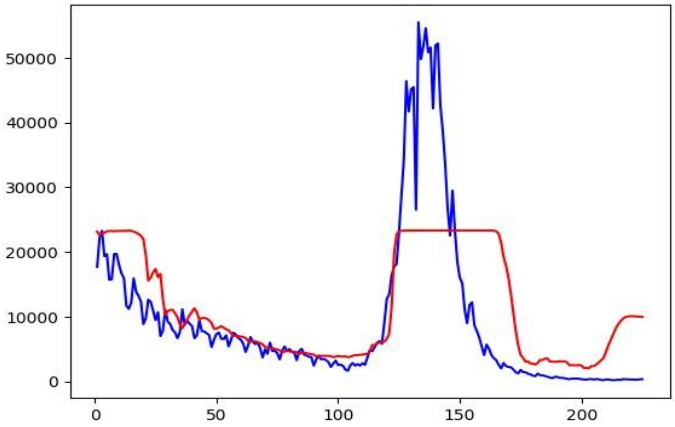

Figure 7. A plot comparing the predicted (red) and actual (blue) number of COVID cases for the test data of 225 examples using LSTM.

The BEDCA was trained for 75 epochs, totaling 36 on a NVIDIA L4 GPU, an average of 0.48 minutes/epoch.

A plot of the predicted vs actual temperatures is shown below (Figure 8).

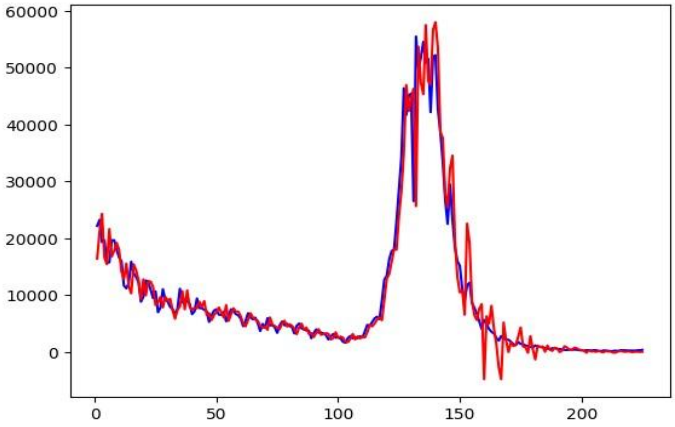

Figure 8. A plot comparing the predicted (red) and actual (blue) number of COVID cases for the test data of 225 examples using BEDCA.

Discussion

From the results, it is evident that the BEDCA has a lower MAE for all three datasets analyzed. The MAE for the BEDCA is 25.9%, 58.7%, and 31.0% lower for each dataset respectively. These results also indicate that the BEDCA is able to learn low and high volatile data well, but struggles to learn cyclic data. Furthermore, it can be observed from the prediction images that the BEDCA's predictions have a higher variance compared to the LSTM's predictions. The BEDCA's predictions are able to take into account more variation in the data, giving BEDCA the ability to learn and predict from more volatile data. This is most evident in the COVID cases data, where the LSTM was unable to predict significantly higher or lower values, while the BEDCA was able to predict with more success. Additionally, in all three of the BEDCA's training loss graphs, it was seen that

the loss was still decreasing. Thus, if trained for more epochs, the BEDCA would almost certainly have performed even better. This was not done due to computational resource limits.

Although the BEDCA was significantly more complex than the LSTM, it trained in a relatively short amount of time when compared to the LSTM. On all three datasets, the BEDCA took less than 3.5 times the amount of time taken for the LSTM to train, although the BEDCA was more than 35,000 times larger. The BEDCA also trains significantly faster than the Transformer. According to Anthony et al. (2023), $\tau T = 6PD$, where τ = FLOPs performed by the training hardware, T = amount of training time, P = number of parameters in the model, and D = the dataset size. Holding τ and D constant, we see that $T \propto P$. Assuming a comparable transformer model contains 15-20 million parameters, the BEDCA trains 5 to 6.7 times faster.

Going back to the three goals laid out at the beginning of this paper, the MAE for the BEDCA were 6.41%, 22.07%, and 14.57% of the standard deviations of the respective datasets. From these, we can see that the model almost achieves the first goal of obtaining a MAE of less than 0.2σ on the testing data. However, the second goal was not achieved, as the BEDCA took longer to train than the LSTM. Lastly, the BEDCA only used 1000 training examples for the first two datasets and 400 training examples for the last dataset, satisfying the third goal.

Conclusion

The above results and analysis indicate that the BEDCA performs significantly better than the LSTM, and is a more practical choice than the Transformer. We believe that still the BEDCA could undergo more testing with more datasets to identify its weaknesses. One such weaknesses identified was that the BEDCA cannot learn properly from non-differenced data. Thus, differencing had to be applied to improve model performance.

Acknowledgments

This project could not have been done without the continued support of Daniel Plymire as well as my parents.

References

Anthony, Q., Biderman, S., & Schoelkopf, H. (2023, August 18). Transformer Math 101. EleutherAI. https://blog.eleuther.ai/transformer-math/

Chen, J. (2022, September 18). Triple Exponential Average (TRIX): Overview, calculations. Investopedia. https://www.investopedia.com/terms/t/trix.asp

Coca Cola stock - live and updated. (2024, January 28). Kaggle. https://www.kaggle.com/datasets/kalilurrahman/coca-cola-stock-live-and-updated/

Hayes, A. (2023, September 30). Bollinger Bands®: What They Are, and What They Tell Investors. Investopedia. https://www.investopedia.com/terms/b/bollingerbands.asp

IBM. (n.d.). What are recurrent neural networks? https://www.ibm.com/topics/recurrent-neural-networks

Latest COVID-19 confirmed cases Kerala. (2022, May 22). Kaggle. https://www.kaggle.com/datasets/anandhuh/covid19-confirmed-cases-kerala

Lee, M. C., Chang, J. W., Yeh, S. C. et al. (2022, January 28). Applying attention-based BiLSTM and technical indicators in the design and performance analysis of stock trading strategies. Neural Comput & Applic 34, 13267–13279. https://doi.org/10.1007/s00521-021-06828-4

Li, A. W., Bastos, G. S. (2020, October 12). Stock Market Forecasting Using Deep Learning and Technical Analysis: A Systematic Review. IEEE, 8. https://doi.org/10.1109/ACCESS.2020.3030226

Padial, D. L. (2022, August 23). Technical Analysis Library in Python Documentation. Technical Analysis Library in Python. https://technical-analysis-library-in-python.readthedocs.io/_/downloads/en/latest/pdf/
Sharkawy, A. N. (2020, August 20). Principle of Neural Network and Its Main Types: Review. Journal of Advances in Applied & Computational Mathematics, 7. https://doi.org/10.15377/2409-5761.2020.07.2

Sang, C., & Pierro, M. D. (2018, November 14). Improving trading technical analysis with TensorFlow Long Short-Term Memory (LSTM) Neural Network. The Journal of Finance and Data Science, 5(1), 1–11. https://doi.org/10.1016/j.jfds.2018.10.003

Tableau. (n.d.). Time Series Forecasting: Definition, Applications, and Examples. https://www.tableau.com/learn/articles/time-series-forecasting#::text=It%20has%20tons%20of%20pr

Time series room temperature data. (2022, November 21). Kaggle. https://www.kaggle.com/datasets/vitthalmadane/ts-temp-1?select=MLTempDataset1.csv

Zhang, L., Wang, R., Li, Z., Li, J., Ge, Y., Wa, S., Huang, S., & Lv, C. (2023, September 13). Time-Series Neural Network: A High-Accuracy Time-Series forecasting method based on kernel filter and time attention. Information, 14(9), 500. https://doi.org/10.3390/info14090500