Pranav Pathak - Science Fair 2024 Logbook

Table of Contents:

Schedule:

Monday, September 11th, 2023:

Today, I made an approximate schedule that I want to follow to successfully complete my project for the science fair.

Here is my schedule that I came up with:
September 12th - September 25th: Research possible topics and come up with a list of my interests. Decide on a particular topic.

September 26th - October 16th: Conduct a detailed literature review/background research on my chosen research topic. Look at some past papers and studies from sources such as Google Scholar.

October 17th - October 22nd: Make a research question based on my literature review/background research. Come up with a hypothesis and support it with reasoning.

October 23rd - November 6th: Brainstorm possible methods and experiments to conduct and implement to help answer the research question. Find datasets that I would use or if I will collect my own data, come up with procedures for obtaining data.

November 7th - December 11th: Conduct experiments (methods section).

December 12th - January 10th: Analyze results in a methodical way. Generate graphs and relevant visuals. Record all relevant discussion points.

January 15th: Conclude project.

January 15th - February 8th: Prepare project for school science fair. Revise logbook content and organization. Summarize the project in a clear way and put it onto the trifold.

February 8th - March 15th: Make presentation and practice for CYSF. Upload all required components to CYSF website.

Choosing a Topic:

Thursday, September 14th, 2023:

- After spending some time thinking about my interests, I decided that I want to do a project that uses machine learning to address a problem.
- I am still unsure whether I want to use machine learning with a business application focus or a healthcare/medical application focus.

Sunday, September 17th, 2023:

- Today, I looked at various machine learning tasks and found that time series forecasting interesting. Time series data is data that is recorded with respect to time. For example, hourly temperatures in Calgary is time series data.
- Time series forecasting is the task of predicting future values of time series data. This topic seems interesting because being able to forecast future values can help to give confidence in decisions.
- After some research, it seems that RNNs and LSTMs are two popular models for time series forecasting, although they each have individual weaknesses. Thus, I want to make a machine learning model that improves on current methods of time series forecasting.
- Rather than just develop my model for a specific time series forecasting problem, such as predicting the number of flu cases, I want to develop a general model that can learn from any type of data.

Wednesday, September 20th, 2023:

- I like my current project idea quite a bit. But, I also want to add something that will make my model really unique. Today, I realized that I can also include technical indicators in my model.
- Technical indicators are specific values that are calculated which describe some aspect of time series data. Generally, technical indicators are usually used in stock price prediction, or technical analysis (which is the broader term). I think that technical indicators will give better information for the model, and might help it to predict more reasonable values.
- Also, I want to clarify that I want to make a model which can learn relationships between data **without** any additional context. This would make my model truly powerful.

Background Research:

Saturday, September 23rd, 2023:

- Today, I looked at ARIMA (Autoregressive integrated moving average). It is a *statistical* method to predict future time series values. It looks at lagged relationships (relationships with previous values at set intervals) and conducts regression on these relationships. Also, it uses Moving Average on these data points.
- I also researched about the RNN (Recurrent Neural Network). It is a large neural network, which is essentially a large function, that splits the input into parts based on the time at which each input was recorded. Then, it processes each in a linear fashion, passing what is called a “hidden state” between each time step.
- So, the RNN also considers the relationship of the input data with time, which is a very important factor. However, a weakness of the RNN is that it cannot capture long term relationships in the input data due to a common machine learning problem called the vanishing gradient problem.

Sunday, September 24th, 2023:

- I researched about the LSTM (Long-Short Term Memory), which improves on the RNN by using "update" and "forget" gates. So, it behaves more like our brain, remembering the crucial information over long input sequences.
- Thus, the LSTM suffers from less issues than the RNN. However, I think that lagged relationships could be better represented in the LSTM.

Saturday, September 30th, 2023:

- I found three Google Scholar articles about projects that made models for time series forecasting while considering technical indicators.
- The first study was about stock price forecasting. It considered 3 technical analysis strategies (RSI, MACD, SMA), and trained a LSTM for each strategy. The paper used LSTMs to figure out if each of these strategies would produce a profit or a loss at a given point in time. The study found that using the LSTMs with each of strategies improved them.
- From the first study, I think that the following could be done to improve the accuracy:
    - Consider all the technical indicators at the same time in the same model.
    - Including more technical indicators to give the model a better understanding of the data.
- The second study was a review article about stock price forecasting projects which included technical indicators. This paper looked at 34 papers that fit multiple criteria. Key conclusions:
    1. Many models use technical indicators to improve accuracy in stock price prediction. Most of the previous studies used LSTMs or CNNs, although some used a combination of models (ex: LSTM + attention and a market vector). Studies that used multiple models usually used one model for information extraction and another for prediction.
    2. Most of the studies analyzed used either accuracy, recall and other related error measurements, or simulated the buying and selling of the underlying asset.
- The key idea from the second paper was that there are many combinations of models that can be used on time series, but finding the correct combination takes logic, math, and testing.
- The third paper normalized five technical indicators in different ways. Then, they fed the stock prices and the normalized technical indicators into a model that used a BiLSTM (Bidirectional LSTM) and attention to output a buy or sell decision. The outputs were used in a stock trading strategy. The study found that using this model architecture, the accuracy increased by a few percent in both short term forecasting as well as predicting the price for the next day.
- From this study, I realized that including attention in my model would be a good idea. Attention is an idea in machine learning models to force models to primarily consider some parts of the data or inputs over others.

Tuesday, October 3rd, 2023:

- I looked at a fourth paper today. This study used a kernel filter to help deal with noise in the data and to better represent the data. The kernel filter is very similar to the convolution used in a CNN. This study also combined the kernel filter with a novel time attention mechanism. The time attention mechanism uses another neural network to generate weights for each of the features.
- By using this combination of models and techniques, the study found that this approach had the best results out of the ones tested, notably the LSTM. Furthermore, the study found that using kernel filters improves the approach and that time attention performed better than other types of attention mechanisms.
- From this paper, I learnt that lagged relationships were something the authors believed could be emphasised more, as mentioned in the Limitations section.

Research Question:

Saturday, October 7th, 2023:

- After some brainstorming, the following research question seemed like the best option: **How can time series forecasting error be minimized by considering technical indicators, lagged relationships, and using machine learning techniques?**

Sunday, October 8th, 2023:

- I also set four main goals for my project.

1. Achieve a MAE of less than 0.2σ on the testing data. I believe that this is a good metric to evaluate the model's predictions, as 0.2σ would make a model quite accurate.
2. Reduce training time on relatively large datasets. This is a problem of other types of models, such as Transformers, which have tens of millions of parameters and take an extremely long time to train.
3. Create a model that does not require more than 5000 training examples to achieve the required predictive accuracy, which also uses technical indicators. This ensures that my model can be useful in more contexts, as in some time series forecasting tasks, 5000 data points may not be available.
4. Create a generalizable model, which is a model that can learn to predict any type of time series data.

Hypothesis:

Wednesday, October 11th, 2023:

- I believe that goals 1 and 3 are relatively easy to meet and that my model will be able to achieve these goals. This is because goals 1 and 3 can be solved by adding more parameters, which will eventually allow the model to learn extremely complex relationships in the input data.
- On the other hand, goals 2 and 4 are harder to accomplish. This is due to the fact that adding more parameters will increase training time and may not make the model more

generalizable. However, I still believe that my model can achieve goal 4, since I will incorporate technical indicators into my model.

Datasets (Materials):

Friday, October 13th, 2023:

- To train and verify the model, I decided to use three datasets to test a certain aspect of the model's predictions. I want to use a dataset that has low volatility at some periods, and high volatility at others. I also want to use a dataset that keeps repeating in cycles, since this pattern is quite useful to be able to learn. Lastly, I want to use a dataset that has high volatility and cyclic patterns.
- For the first dataset, it could be possible to use stock price data, since at different time periods, the stock market behaves differently (high and low volatile). For the second dataset, it could be possible to use weather data, as it tends to repeat, or carnival attendance data. For the third dataset, it could be possible to use disease data, such as COVID daily cases data.

Sunday, October 15th, 2023:

- Today, I looked for datasets on Kaggle, a website where many machine learning oriented datasets are published.
- For the first dataset, I found Coca-cola data from 1962-present. At first, this data grew slowly (low volatility) and then became more volatile in recent years.
- For the second dataset, I found room temperatures collected from an IOT device. This dataset shows a strong cyclic pattern.
- For the third dataset, I found daily COVID cases data in Kerala, India from January 31, 2020 to May 22, 2022. This data has random spikes (high volatility), but also somewhat shows cyclic patterns.

Sunday, February 11th, 2024:

- After deciding yesterday that I wanted to extend my model to healthcare related data for testing, I looked for this type of data.
- Of all the datasets I found today, two were most promising.
    - The first dataset was a dataset of ECG/EKG (electrocardiogram) data of 47 patients at a hospital. This dataset could be used to identify patterns of ECG data of patients with a certain condition (such as arrhythmia), which could then be used to predict conditions for new patients.
    - The second dataset was a dataset of electroencephalogram (EEG) data of 500 subjects. EEG data measures electrical signals in the brain. It is useful for medical diagnosis of diseases such as brain tumors and epilepsy. In this dataset, 100 subjects experienced epileptic seizure during recording. This dataset could be

used to identify patterns in EEG data of patients with epileptic seizures to help predict if and when future seizures could take place, or to help with diagnosis.

- Although both datasets are unique and can be impactful, I believe that the second dataset has more potential to help diagnosis, especially if extended to other types of diseases. For example, could EEG data help with accurate alzhiemer's disease prediction? I think that this dataset has the most potential for future works as well, so I will use this dataset as well for testing my model with another powerful model.
- The other model I will use the EEG data for is N-BEATS, a powerful developed by Oreshkin, B. et al. Testing against this model will give a more comprehensive comparison of my model to other time series forecasting models/methods.

Procedure:

Sunday, October 29th, 2023:

Data collection procedures:

1. Data will be collected from the respective Kaggle datasets.
2. Data will be cleaned. Missing values and NaNs will be replaced by the mean of the previous value and the next value.
3. Technical indicators will be calculated for the time series data. I will calculate technical indicators using the TA library in python.

Data preparation procedures:

1. The first 43 rows will be removed since the technical indicator calculations will cause these values to become NaN. Note: the number 43 is chosen, since the TRIX indicator is NaN until row 43.
2. Then, the inputs will be normalized by calculating the z-score for each of the 7 technical indicators. Furthermore, the inputs will be batched into sequences of length 100, with 7 technical indicators for each training example. In a similar way, the testing dataset will be created.
3. For the BEDCA inputs, the difference between each two consecutive z-scores will be calculated and used as the training and testing data.

My model is **B**elief **E**ncoder-**D**ecoder with **C**onvolutional **A**ttention (BEDCA).

Model training procedures (for each dataset):

1. The LSTM and the BEDCA will be coded using the TensorFlow library in Python.
2. The LSTM will be trained for 150 epochs on a L4 GPU. Training losses will be recorded.
3. Training losses for the LSTM will be plotted.
4. The predictions for the LSTM on the testing data will be plotted along with the actual testing data.

5. Mean absolute error for the LSTM's predictions and the actual testing data will be calculated.
6. The BEDCA will be trained for 75 epochs on a L4 GPU. Training losses will be recorded.
7. Training losses for the BEDCA will be plotted.
8. The predictions for the BEDCA on the testing data will be plotted along with the actual testing data.
9. Mean absolute error for the BEDCA's predictions and the actual testing data will be calculated.

Raw Data:

Saturday, December 16th, 2023:

- My model and the LSTM finished running on the first dataset, the stock price data.
- The Mean Absolute Error (MAE) for the LSTM was 2.86 while the MAE for the BEDCA was 0.74.
- The LSTM took 0.42 mins/epoch on average while the BEDCA took 1.28 mins/epoch on average.

Friday, December 22nd, 2023:

- My model and the LSTM finished running on the second dataset, the room temperature data.
- The Mean Absolute Error (MAE) for the LSTM was 2.54 while the MAE for the BEDCA was 1.49.
- The LSTM took 0.38 mins/epoch on average while the BEDCA took 1.24 mins/epoch on average.

Saturday, January 6th, 2024:

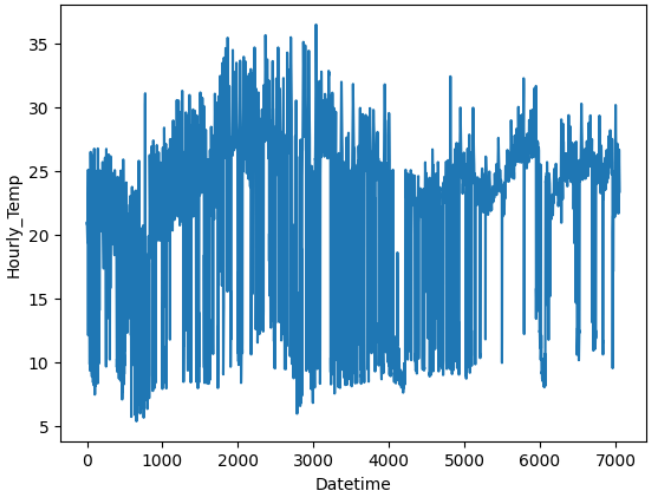
- My model and the LSTM finished running on the third dataset, the daily COVID cases data.
- The Mean Absolute Error (MAE) for the LSTM was 5806.73 while the MAE for the BEDCA was 1799.50.
- The LSTM took 0.15 mins/epoch on average while the BEDCA took 0.48 mins/epoch on average.

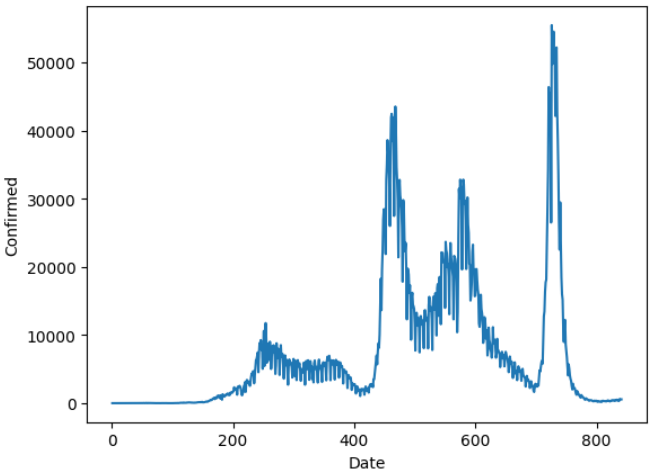
Saturday, February 17th, 2024:

- My model and the N-BEATS model finished training on the EEG dataset. My model took 8 hours to train for 100 epochs, while the N-BEATS model took 36 minutes to train for 300 epochs.

Results:

Saturday, November 4th, 2023:



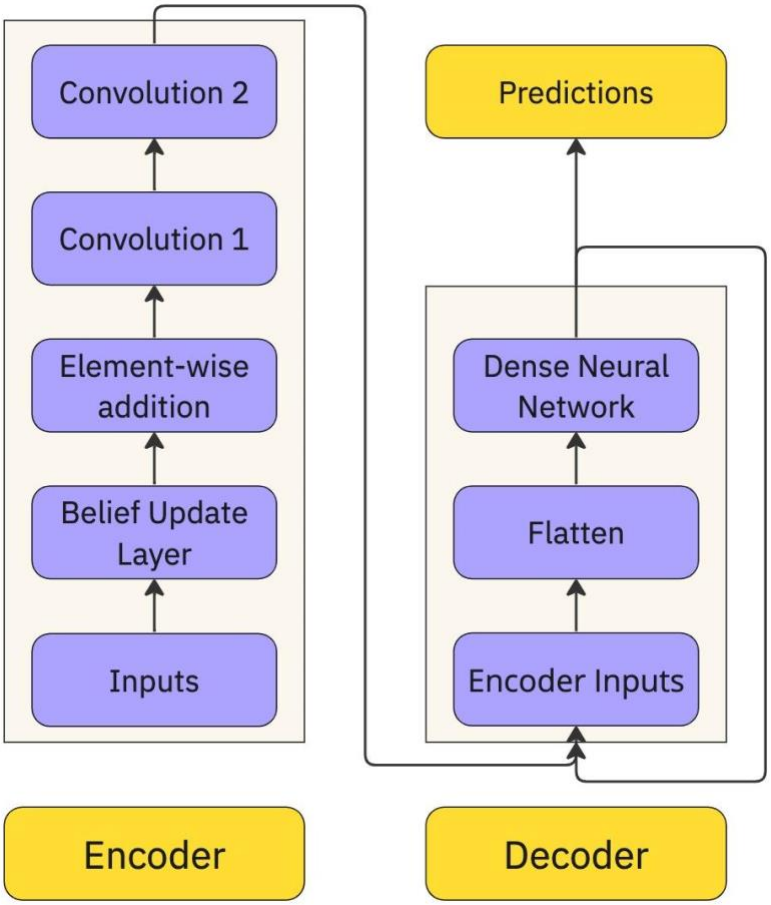


Thursday, December 7th, 2023:

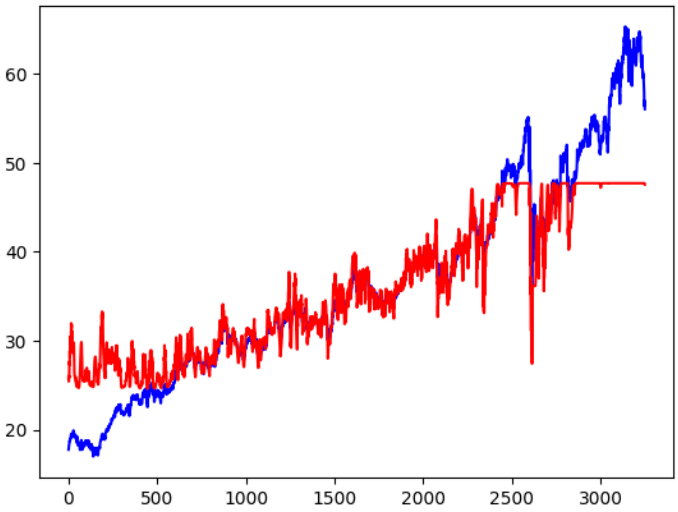
- Currently, I do not have any results, as my model is getting an error of: “no gradients provided for any variable: [...]”.
- Gradients are large groups of numbers which tell the model how to improve its predictions. So, my model was not training properly due to this error.
- This error has been occurring for about the past month. I believe that this is due to the fact that some of my calculations are being done in numpy while others are being done in tensorflow, which prevent tensorflow from calculating the gradients. So, I will try to convert all of my operations to tensorflow.
- Also, current training time is too long. So, I have reduced the input sequence length from 100 to 50. I believe that this will help make the model run at least four times as fast, since less computations are required for each input.

Monday, December 11th, 2023:

- I have fixed the “no gradients provided for any variable: [...]” error by converting all the operations in my model to tensorflow operations. Furthermore, I changed the activation functions in the dense neural network which generate the final predictions, as there was a 0/1 gradient problem (gradient became 0/1 which affected other gradients).
- So, I will now start training the model. This will be the final version of my model. Here is a diagram of my model’s architecture:

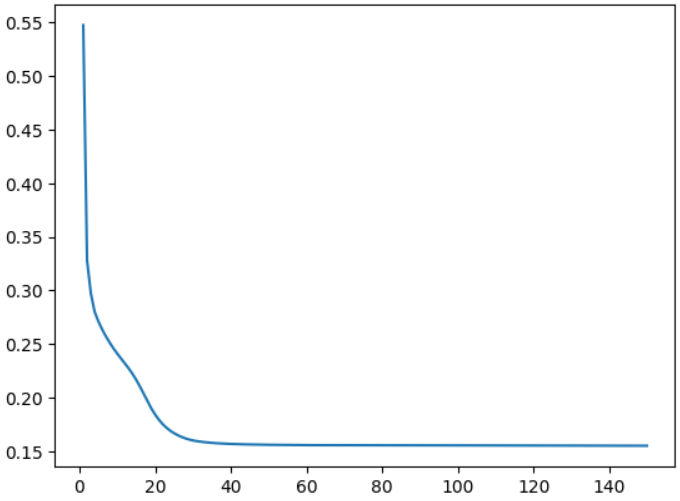


Saturday, December 16th, 2023:
LSTM results for the stock price data:



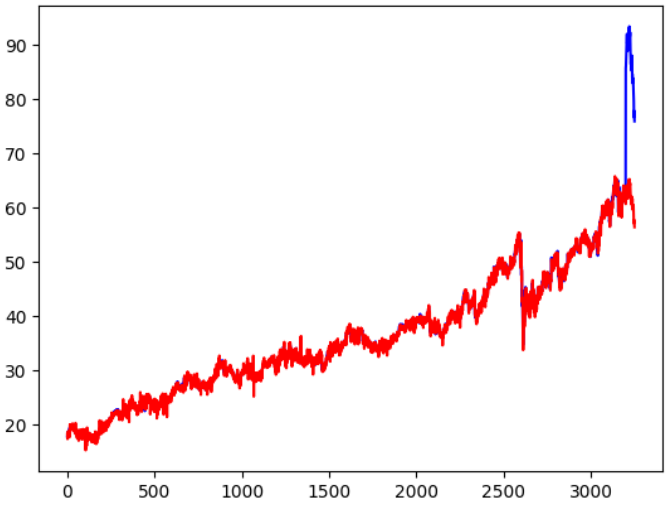
These were the LSTM's predictions for the testing data. The blue line is the actual stock prices and the red line is the LSTM's predictions.

The training losses of the LSTM are:



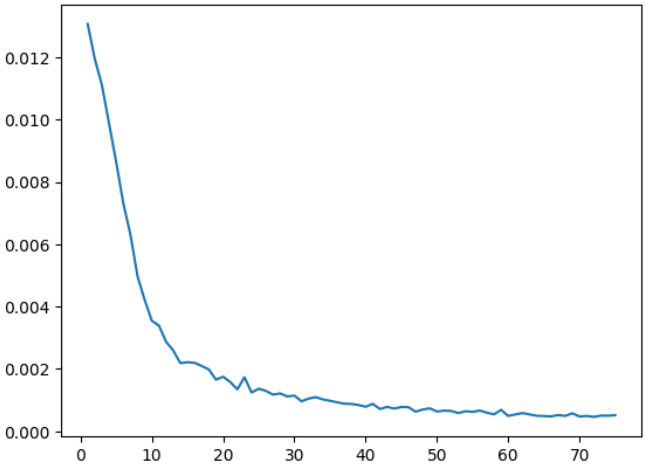
The LSTM quickly learns to predict how it currently predicts, but does not improve after around epoch 50. This suggests that the LSTM has low learning capacity. The LSTM's predictions also show that it is unable to capture complex relationships in the data.

BEDCA results for the stock price data:



These were the BEDCA's predictions for the testing data. The blue line is the actual stock prices and the red line is the BEDCA's predictions.

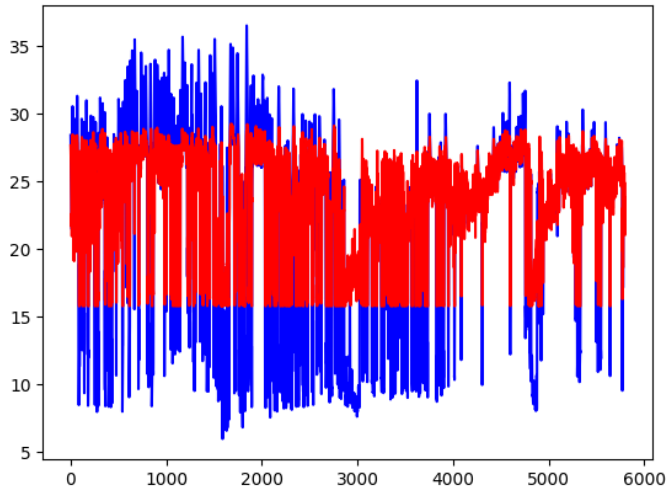
The training losses of the BEDCA were:



The BEDCA is still learning as the loss function is still decreasing. This and the exemplary above graph indicate that the BEDCA might be performing better than the LSTM.

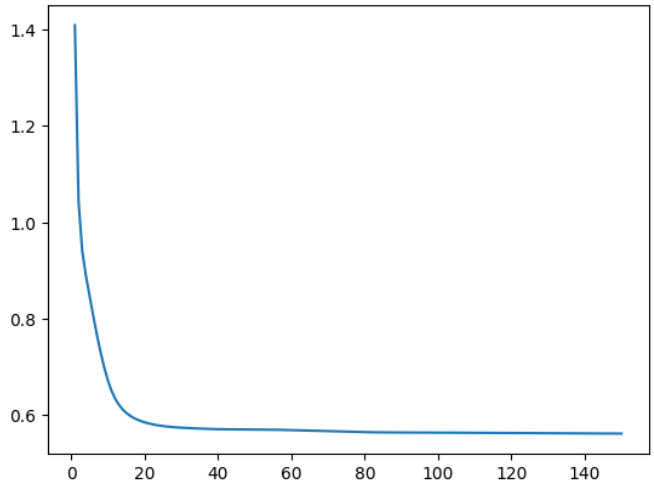
Friday, December 22nd, 2023:

LSTM results for the room temperature data:



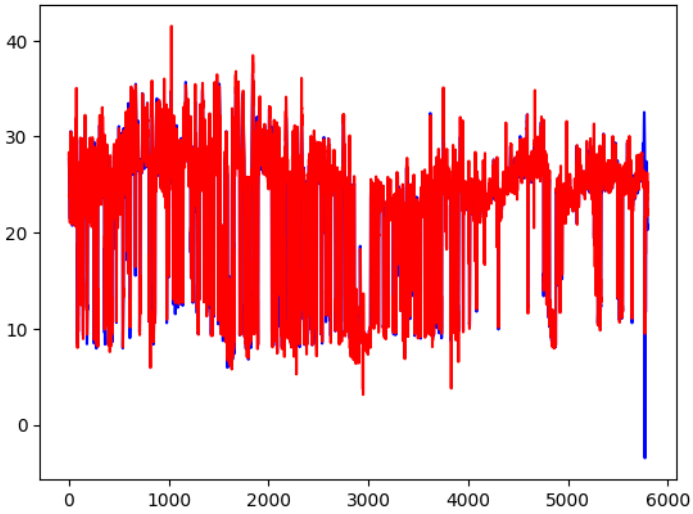
These were the LSTM's predictions for the testing data. The blue line is the actual room temperatures and the red line is the LSTM's predictions.

The training losses of the LSTM are:



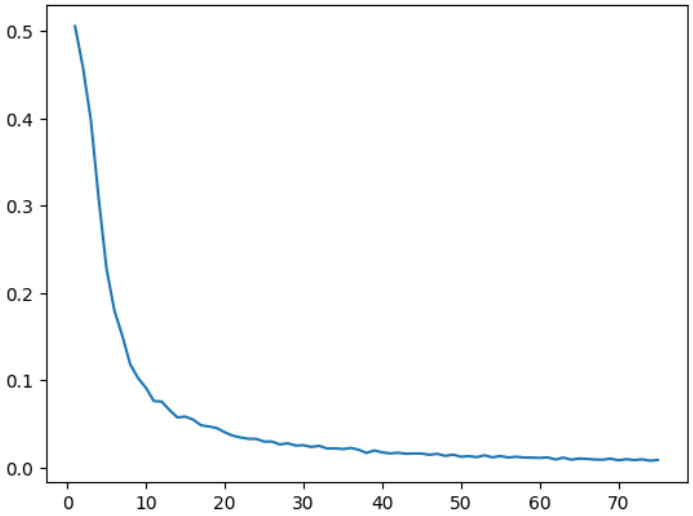
In this graph as well, the LSTM's loss quickly decreases and flattens out. This indicates that the LSTM learns quickly, but does not properly learn the relationships in the data.

BEDCA results for the room temperature data:



These were the BEDCA's predictions for the testing data. The blue line is the actual room temperatures and the red line is the BEDCA's predictions.

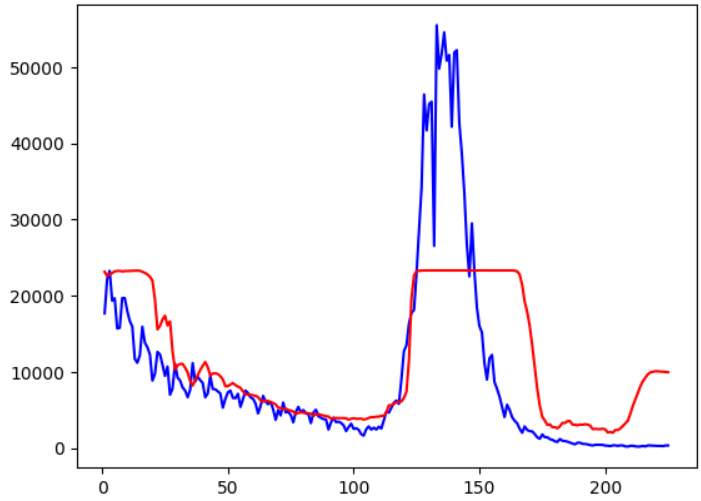
Here are the training losses for the BEDCA:



In this training loss graph as well, the BEDCA is evidently still learning. In this graph, the slope is even higher than the previous training loss graph.

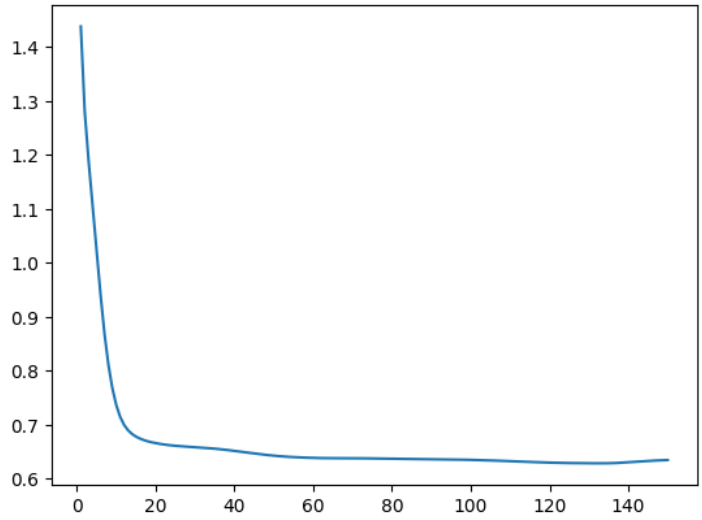
Saturday, January 6th, 2024:

LSTM results for the daily COVID cases data:



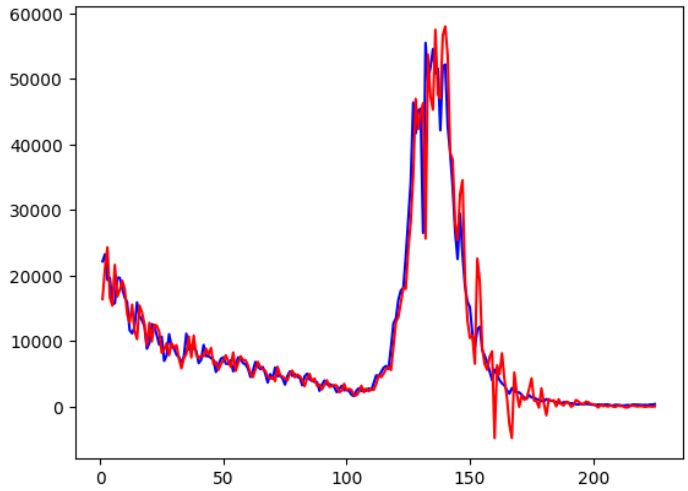
These were the LSTM's predictions for the testing data. The blue line is the actual number of COVID cases and the red line is the LSTM's predictions.

Here are the LSTM's training losses:



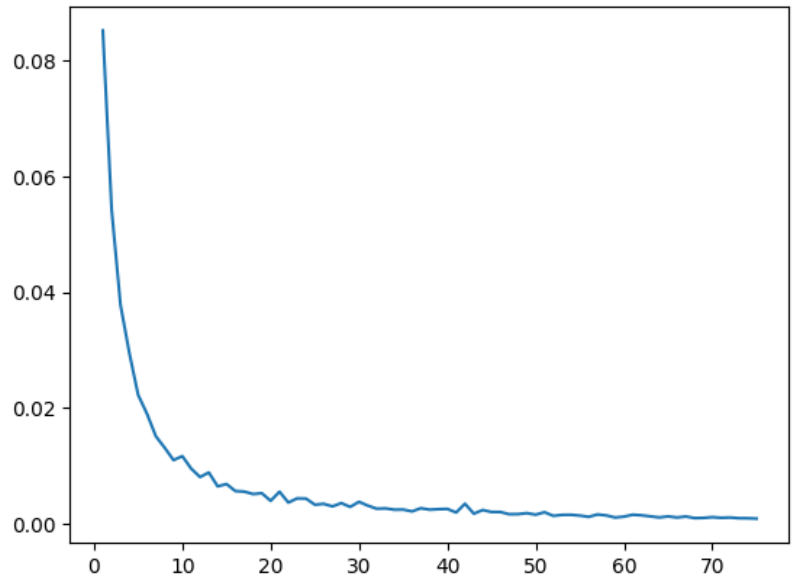
In this graph, it can be seen that the LSTM learned the relationship slower than in other datasets.

BEDCA results for the COVID cases data:



These were the BEDCA's predictions for the testing data. The blue line is the actual number of COVID cases and the red line is the BEDCA's predictions.

Here is the training losses for the BEDCA:

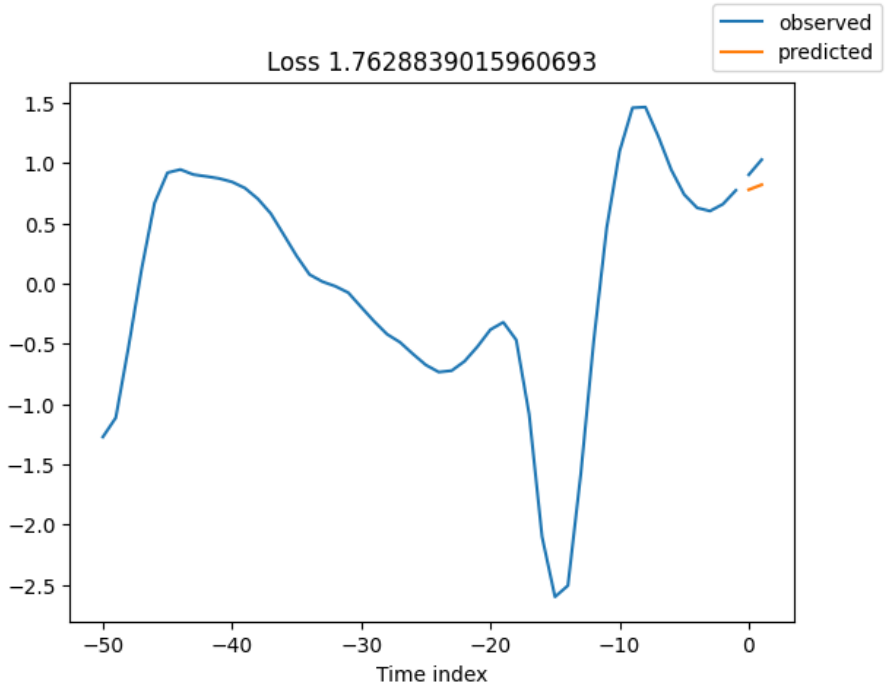


Sunday, January 7th, 2024:

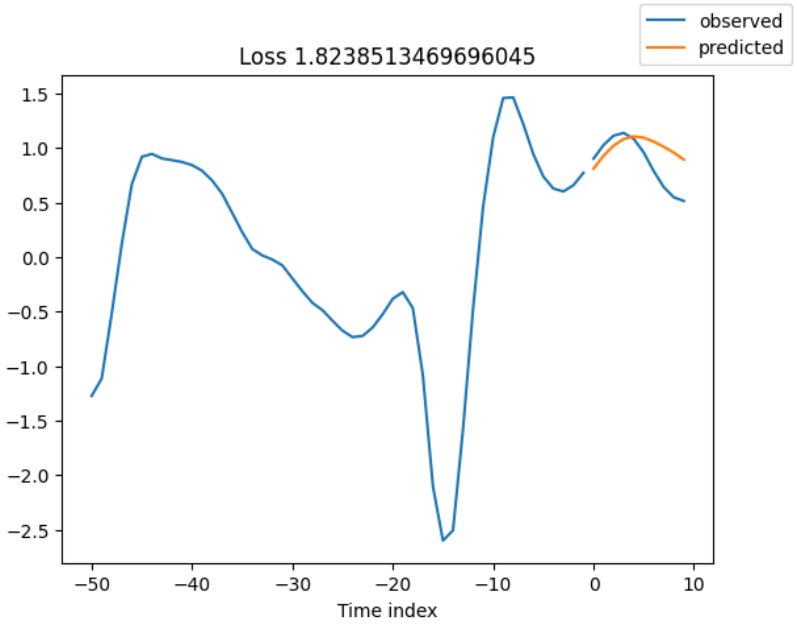
	LSTM MAE	Belief Encoder-Decoder MAE	% Reduction
Stock prices	2.86	0.74	74.13
Temperatures	2.54	1.49	41.34
COVID cases	5806.73	1799.50	69.01

	LSTM min/epoch	Belief Encoder-Decoder min/epoch	Ratio
Stock prices	0.42	1.28	3.05 : 1
Temperatures	0.38	1.24	3.26 : 1
COVID cases	0.15	0.48	3.2 : 1

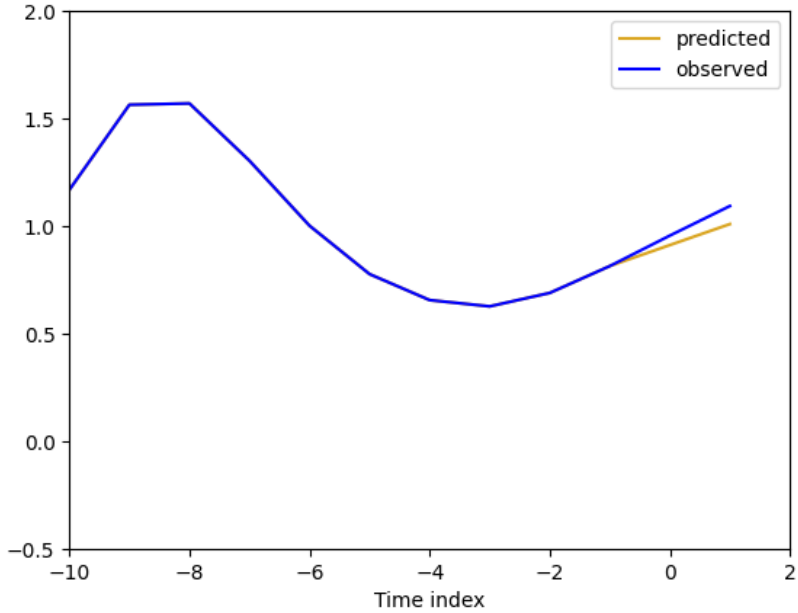
Saturday, February 17th, 2024:
I finished training the BEDCA and the N-BEATS models on the EEG data.
Graph of N-BEATS prediction of length 2:



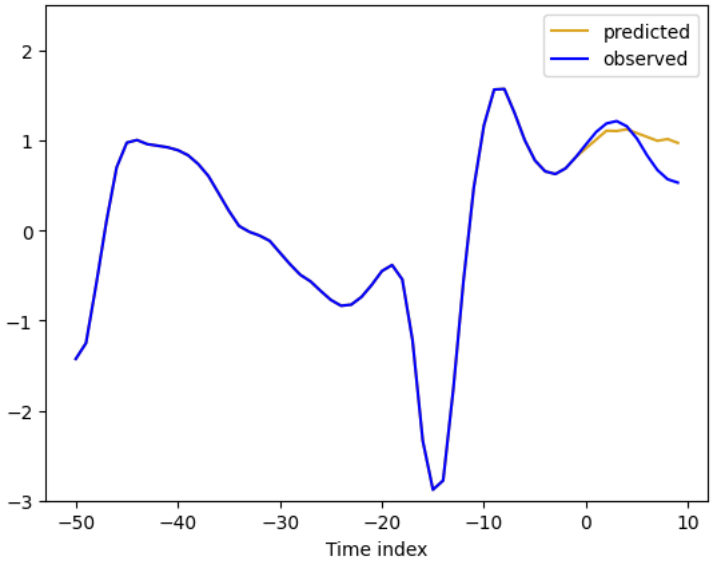
Graph of N-BEATS prediction of length 10:



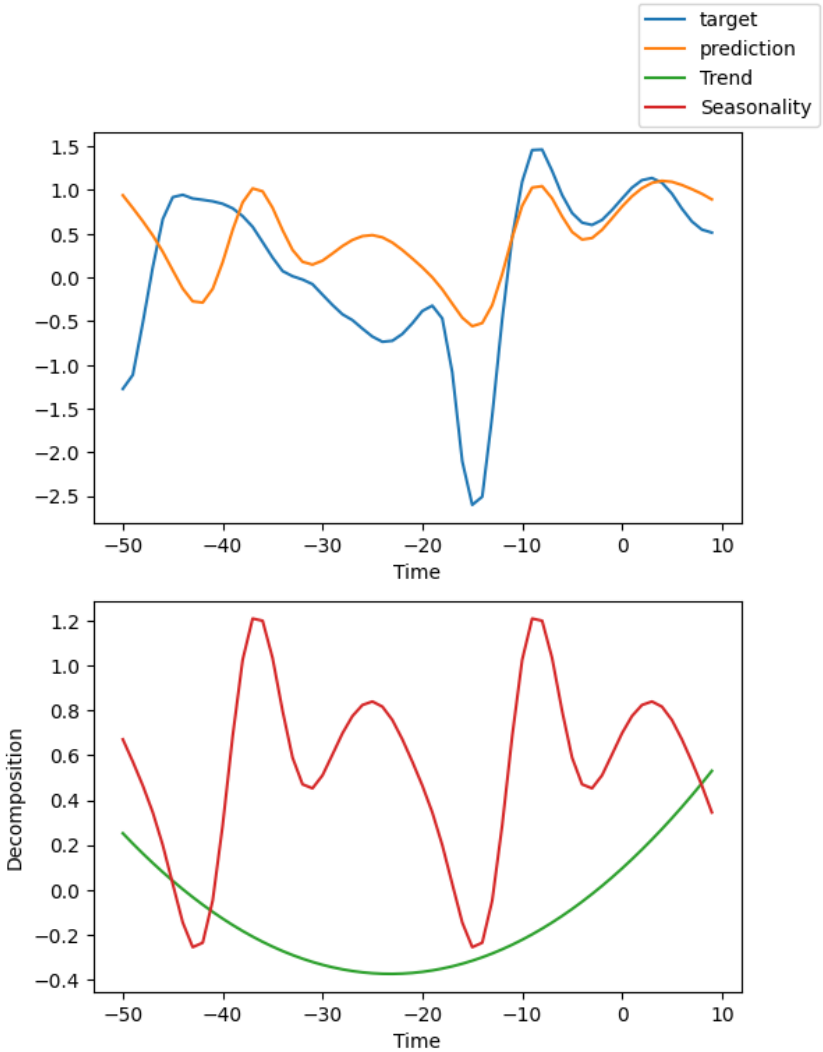
Graph of BEDCA prediction of length 2:



Graph of BEDCA prediction of length 10:



Decomposition of N-BEATS prediction of length 10 into trend and seasonality:



Discussion/Conclusion:

Friday, January 12th, 2024:

- From the results, it is quite evident that the BEDCA has a lower MAE for all three datasets. So, the BEDCA is almost at least twice as effective in its predictions.
- The 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 predictions images that the BEDCA's predictions have a lower variance compared to the LSTM's predictions. The BEDCA's predictions are able to take into account more variation in the data, giving the 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 do this with much more success.
- In all three of the BEDCA's training loss graphs, it can be seen that the loss is still decreasing. Thus, if trained for more epochs, the BEDCA would almost certainly have performed even better. This was not done due to difficulty in obtaining and keeping the L4 GPU active and the cost associated with this.

Wednesday, January 17th, 2024:

- The BEDCA was significantly more complex than the LSTM, but 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. I believe that this is because the LSTM performs more complicated calculations and updates than the BEDCA does.
- Comparing the BEDCA with a comparable Transformer (another extensively used model) would require that the Transformer have around 15 - 20 million parameters. Thus, by scaling laws, BEDCA should train around 5 - 6.7 times faster than the Transformer.

Friday, January 19th, 2024:

- Did I meet the goals I laid out at the start of my project?

1. **Achieve a MAE of less than 0.2σ on the testing data:** For the first and third datasets, this goal was met. However, this goal was marginally missed on the room temperature dataset. This goal could almost certainly have been met if the BEDCA was trained for more time.
2. **Reduce training time on relatively large datasets:** This goal was not met, since the BEDCA took longer to train than the LSTM on all three datasets.
3. **Create a model that does not require more than 5000 training examples to achieve the required predictive accuracy:** This goal was met, since a maximum of 1000 training examples were used to train the BEDCA, and the BEDCA had quite a low loss using only 1000 training examples.

4. **Create a generalizable model, which is a model that can learn to predict any type of time series data**: This goal was met, since the same model was used for all three datasets. However, this seems to be the least important goal since the LSTM was also a generalizable model.

My hypothesis was partially correct. I was able to achieve goals 3 and 4, almost achieved goal 1, but was unable to meet goal 2. I predicted that goal 4 would be hard to achieve, but I was able to meet it quite easily without considering any additional factors in my model (such as considering variant in COVID data). I predicted that I would achieve goal 1, but was unable to do so on the second dataset, highlighting the fact that the BEDCA is unable to predict well on cyclic data.

Sunday, February 18th, 2024:

For the EEG data:

- From the results, it can be seen that the BEDCA performs significantly better than the N-BEATS model for the forecast horizon of 2. However, the BEDCA has slightly lower Symmetric Mean Absolute Percentage Error (SMAPE) than the N-BEATS but slightly higher Mean Squared Error (MSE) for the forecast horizon of 10.
- This indicates that the error for the BEDCA compounds quickly. However, this could also be due to the fact that the N-BEATS model is specifically trained for a certain input length, thus focusing more on prediction of this length. On the other hand, the BEDCA does not focus on a particular prediction length.
- Looking at the N-BEATS prediction versus the BEDCA prediction for the forecast horizon of 10, it is evident that N-BEATS is able to faster adapt to the downtrend, while the BEDCA struggles with this, being much more accurate in the first few predictions, and less accurate in the last few.
- From the N-BEATS decomposition, the last 5 prediction time steps have a significant decrease due to seasonality, which is larger than the increase due to trend. Thus, it can be further inferred that BEDCA struggles with seasonality, but is able to capture the trend accurately.
- This agrees with the previous analysis of results from the first three datasets, where the BEDCA model struggled with the cyclic pattern in the room temperature data.

Applications/Future Considerations:

Tuesday, January 23rd, 2024:

- The most significant future consideration is decreasing training time:
    - Consider how to effectively backpropogate sparse weights (how to quickly update values which are very close to 0?). Most of my weights are close to 0, so I could try to adjust the backpropagation method to mainly consider non-zero weights.
    - I could also provide custom gradients.
    - Lastly, I could consolidate multiple operations into one larger operation

- Some future applications for the BEDCA are applying this model on healthcare data. Some possible avenues are:
    - Abnormal heart rate prediction. This could come from watch data.
    - Hospital resource pressure prediction. For example, hospital wait times could be predicted.

Sunday, February 18th, 2024:

- Future research could expand on the current EEG values forecasting. For example, the BEDCA could be improved to predict the occurrence of epileptic seizures.
- The BEDCA model could also be combined with another model to get the best of both models.

Acknowledgements:

Thank you to Daniel Plymire as well as my parents for their continued support throughout the project.

Bibliography:

Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., & Elger, C. E. (2001, November 20). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. Physical Review, 64(6). https://doi.org/10.1103/physreve.64.061907

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/

EEG (electroencephalogram) - Mayo Clinic. (2022, May 11). https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875#:~:text=An%20electroencephalogram%20(EEG)%20is%20a,lines%20on%20an%20EEG%20recording.

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

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019, May 24). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv.org. https://arxiv.org/abs/1905.10437

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/

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

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

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

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

View of Principle of Neural Network and its main types: Review — Journal of Advances in Applied & Computational Mathematics. (n.d.). https://avantipublisher.com/index.php/jaacm/article/view/851/502

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