Stock Market Prediction Using Recurrent Neural Networks

code available here

Kajol Sahu
kajolsahu.iitkgp@gmail.com
20CY20017

Vinayak Shrivastava
shrivastava.vinayak.iitkgp@gmail.com
20QE30011

1. Introduction

Overview

Stock market prediction is a crucial tool in the financial industry, enabling investors and analysts to make informed decisions by anticipating future price movements. This project utilises a Recurrent Neural Network (RNN), a type of deep learning model that excels at analysing time-series data to predict Google, Bitcoin, Binance, and Ethereum stock prices. By capturing the temporal patterns in stock prices, the RNN model aims to provide accurate predictions, offering insights that can enhance trading strategies and risk management. The project demonstrates the potential of machine learning in financial forecasting, bridging the gap between data-driven analysis and market trends.

Building on the methodologies used in traditional stock prediction, this project extends the application of Recurrent Neural Networks (RNNs) to the volatile and rapidly evolving world of cryptocurrencies, specifically Bitcoin, Ethereum, and Binance Coin (BNB). Unlike traditional stocks, cryptocurrencies operate in a decentralised market, often influenced by factors such as technological developments, regulatory news, and market sentiment. This introduces additional layers of complexity and unpredictability.

Objective

The primary objective of this project is to develop and implement a Recurrent Neural Network (RNN) model capable of predicting future stock prices based on historical data.

Dataset

The dataset used in this project consists of historical price data for **Google**, **Bitcoin**, **Binance**, and **Ethereum**, sourced from **Yahoo Finance**. This data includes key financial metrics such as opening prices, closing prices, high and low prices, and trading volumes, which are essential for training the RNN model to predict future cryptocurrency prices.

2. Recurrent Neural Networks (RNNs)

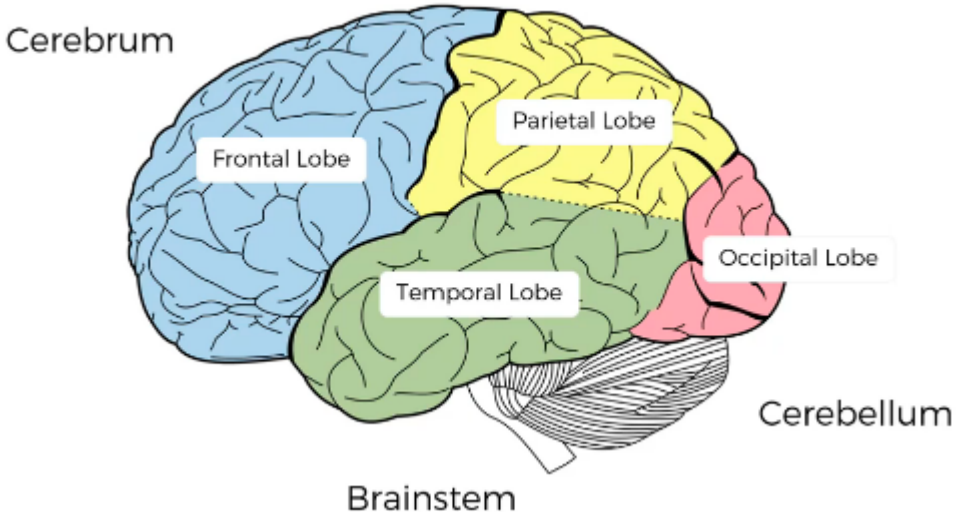
Concept

Recurrent Neural Networks (RNNs) are a type of neural network designed to recognise patterns in data sequences. Unlike traditional neural networks, RNNs have connections that

form cycles, allowing them to maintain a "memory" of previous inputs. RNNs are particularly well-suited for time-series data, such as stock prices, where the order of data points is crucial. By processing data sequentially and maintaining context, RNNs can capture temporal dependencies, making them practical for predicting future values based on past trends.

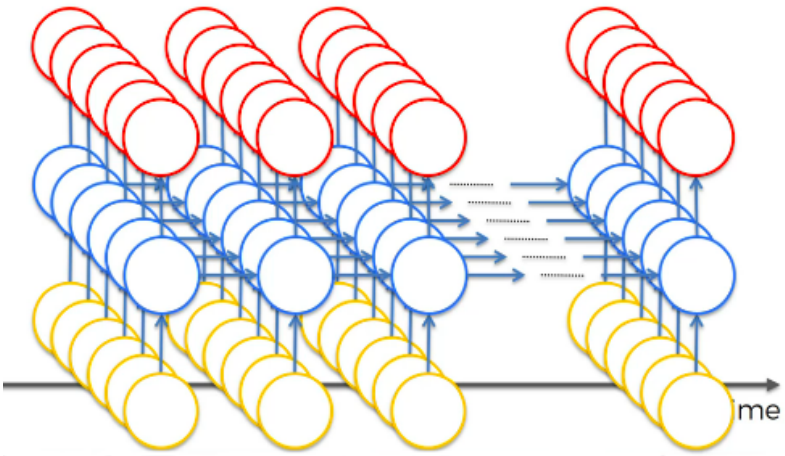
As we try to model Machine Learning to behave like brains, weights represent long-term memory in the Temporal Lobe. Recognition of patterns and images is done by the Occipital Lobe, which works similarly to Convolution Neural Networks. Recurrent Neural Networks are like short-term memory, remembering recent memories and creating context similar to the frontal lobe. The Parietal Lobe is responsible for special recognition like Boltzmann Machines. Recurrent Neural Networks connect neurons to themselves through time, creating a feedback loop that preserves short-term and long-term memory awareness.

The following diagram represents the old-school way to describe RNNs. It shows a Feedback Loop (temporal loop) structure that connects hidden layers to themselves and the output layer, giving them a short-term memory.



Compact Form Representation

Expanded Form Representation

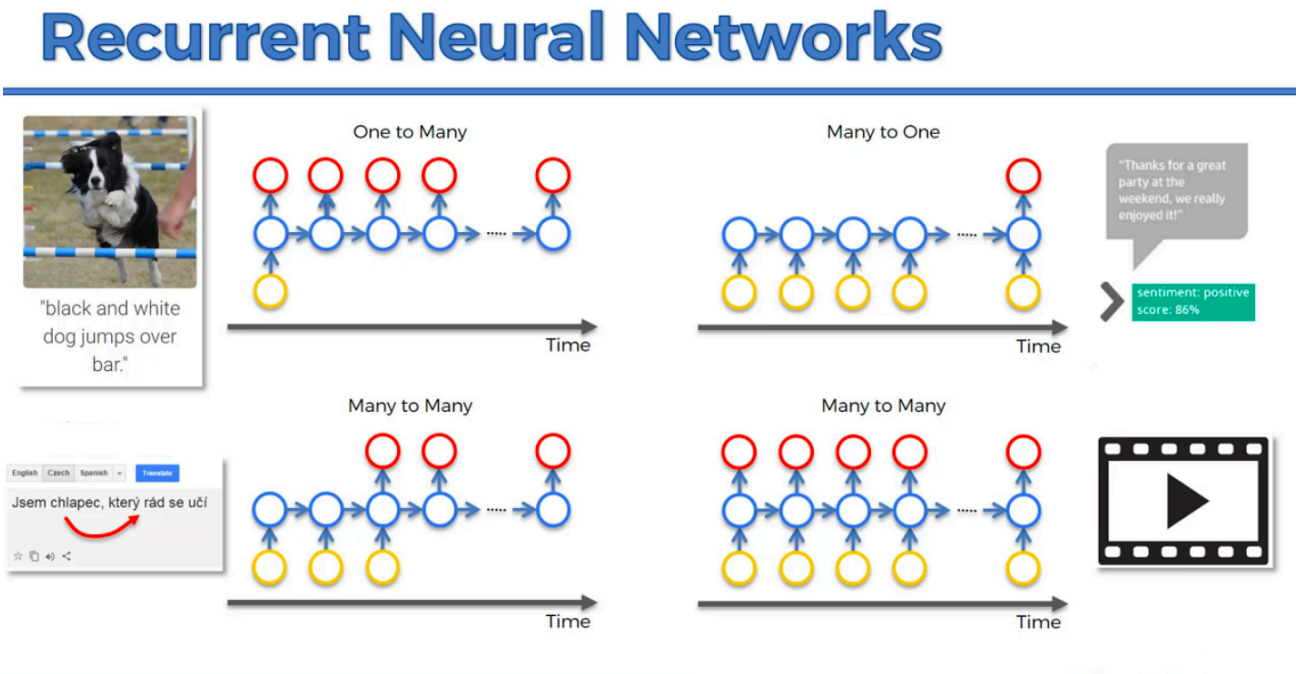


A more modern representation shows the following RNN types and uses examples:

One-To-Many: Computer description of an image. CNN is used to classify images, and then RNN is used to make sense of images and generate context.

Many-To-One: Sentiment Analysis of text (gauge the positivity or negativity of text)

Many-to-Many: Google translate of language whose vocabulary changes based on the subject's gender and a movie's subtitling.



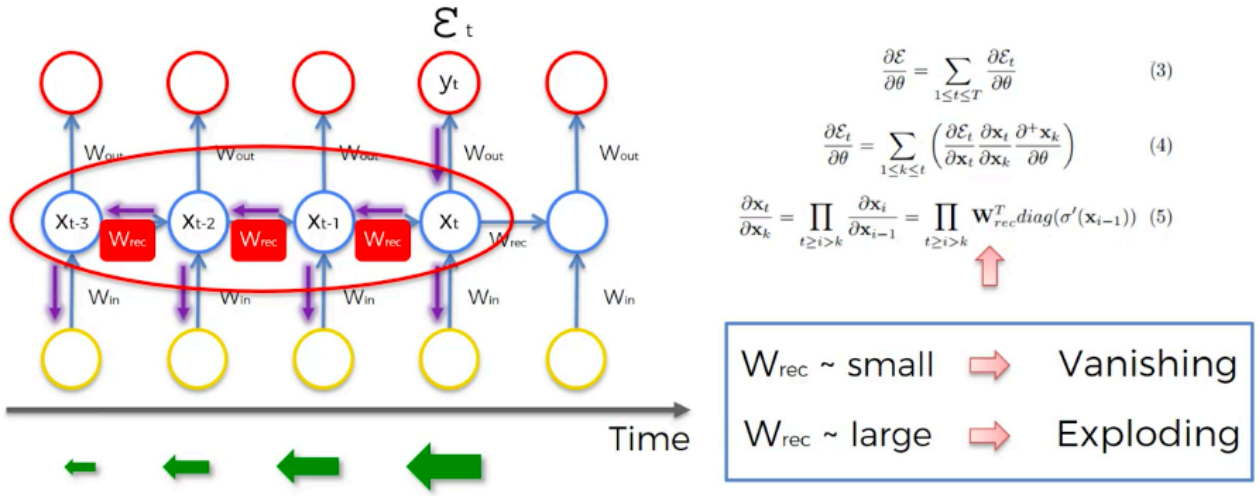
Applications

Beyond stock market prediction, RNNs have a wide range of applications. They are used in natural language processing (NLP) for tasks such as language translation, text generation, and sentiment analysis. In video analysis, RNNs help recognise actions or events over time, making them valuable for tasks like video captioning and surveillance. RNNs are also applied in speech recognition, music composition, and any other domain where sequential data plays a key role.

4. Model Development

RNN Gradient Problem

The gradient in an RNN is used to update the weights by looking back over a user-defined number of steps. A lower gradient makes adjusting neurons' weights (vanishing gradient) further back in time more challenging. This issue arises because earlier layers' outputs are used as inputs for subsequent layers, causing older neurons to train more slowly than the more recent ones, similar to a domino effect.



Expanding Gradient Solutions

1. Truncated Back-propagation
Stop back-propagation after a certain point (not optimal because it does not update all the weights). Better than doing nothing, which can produce an irrelevant network.

2. Penalties
The gradient can be penalised and artificially reduced.

3. Gradient Clipping
A maximum limit for the gradient which stops it from rising more.

Vanishing Gradient Solutions

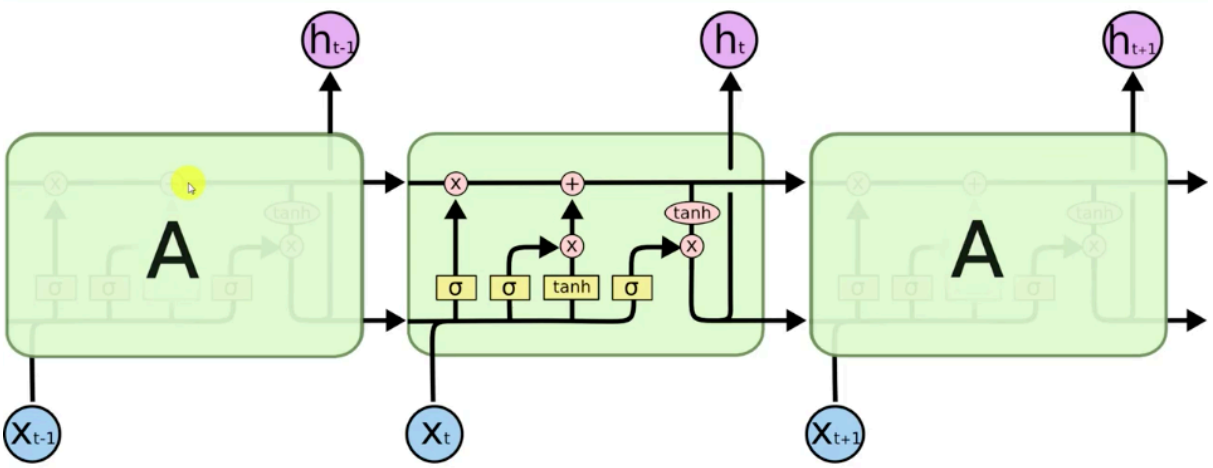
1. Weight Initialization
You can be smart about initialising weights to minimise the vanishing gradient problem.

2. Echo State Network
Designed to solve the vanishing gradient problem. It's a recurrent neural network with a sparsely connected hidden layer (typically 1% connectivity). The connectivity and weights of hidden neurons are fixed and randomly assigned.

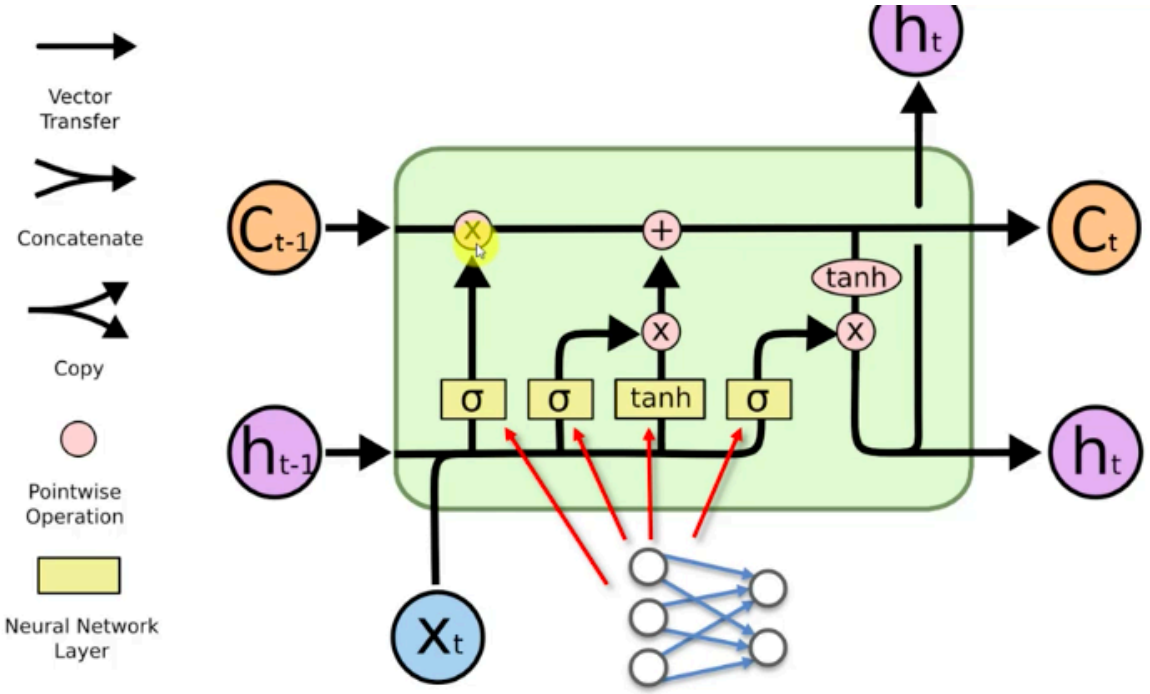
3. Long Short-Term Memory Networks (LSTM)
The most popular RNN structure to tackle this problem.

LSTM

When the weight of an RNN gradient 'W_rec' is less than 1, we get a Vanishing Gradient; when 'W_rec' is more than 1, we get an Expanding Gradient; thus, we can set 'W_rec = 1'.

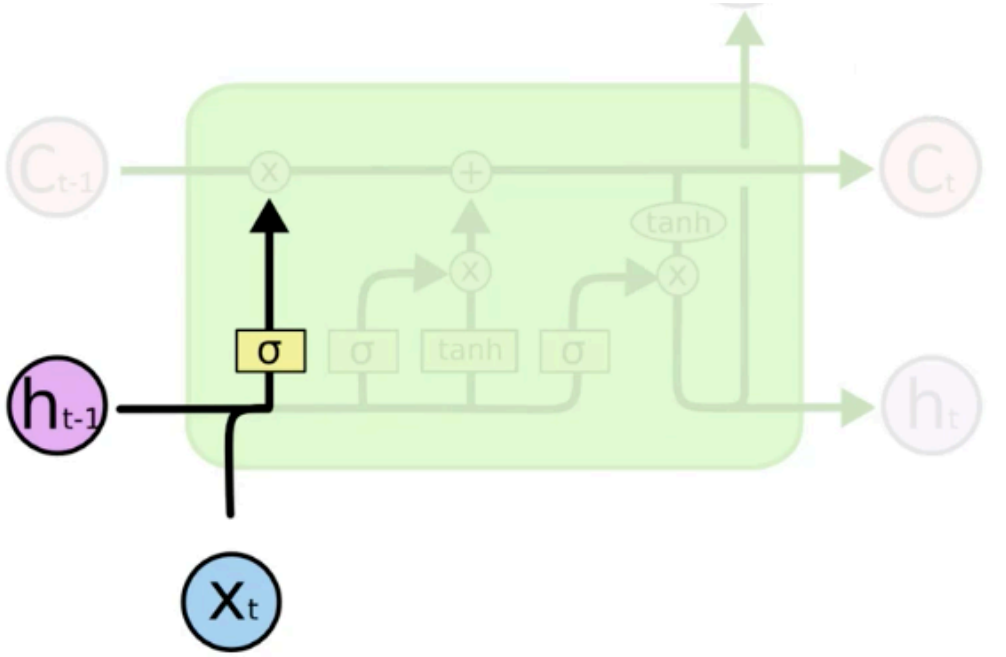


- Circles represent Layers (Vectors).
- 'C' represents Memory Cells Layers.
- 'h' represents Output Layers (Hidden States).
- 'X' represents Input Layers.
- Lines represent values being transferred.
- Concatenated lines represent pipelines running in parallel.
- Forks are when Data is copied.
- Pointwise Element-by-Element Operation (X) represents valves (from left to right: Forget Valve, Memory Valve, Output Valve).
- Valves can be open, closed or partially open as an Activation Function decides.
- Pointwise Element-by-Element Operation (+) represents a Tee pipe joint, allowing stuff through if the corresponding valve is activated.
- Pointwise Element-by-Element Operation (Tanh) Tangent function that outputs (values between -1 to 1).
- Sigma Layer Operation Sigmoid Activation Function (values from 0 to 1).



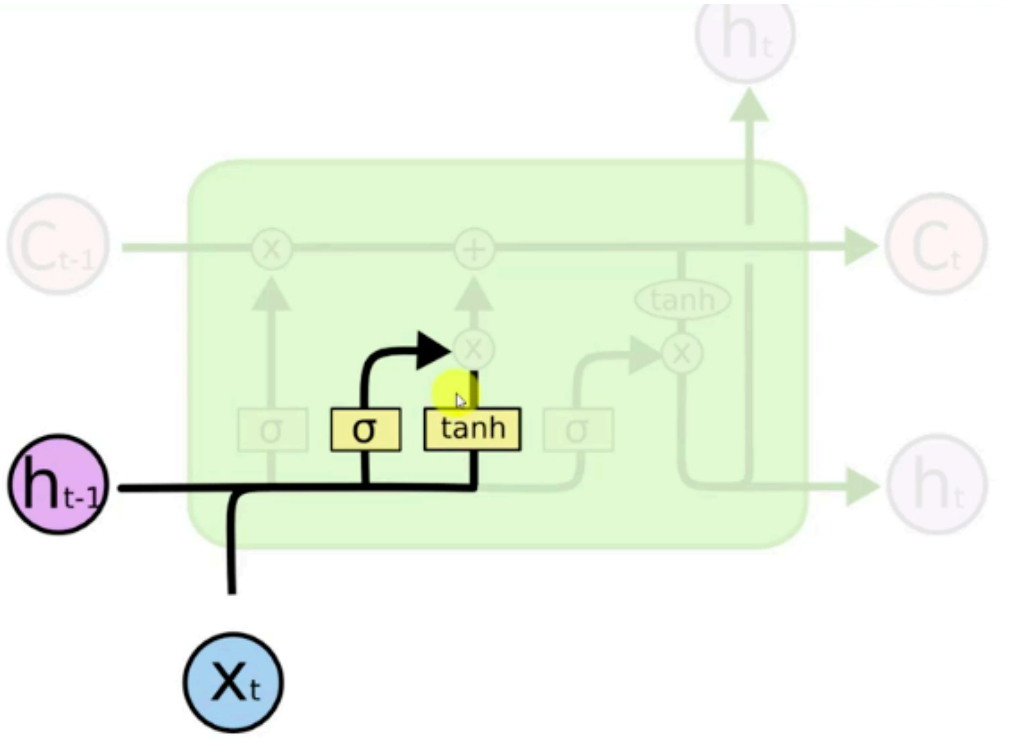
LSTM Step 1

The new value' X_t' and the value from the previous node 'h_t-1' decide whether the forget valve should be opened or closed (Sigmoid).



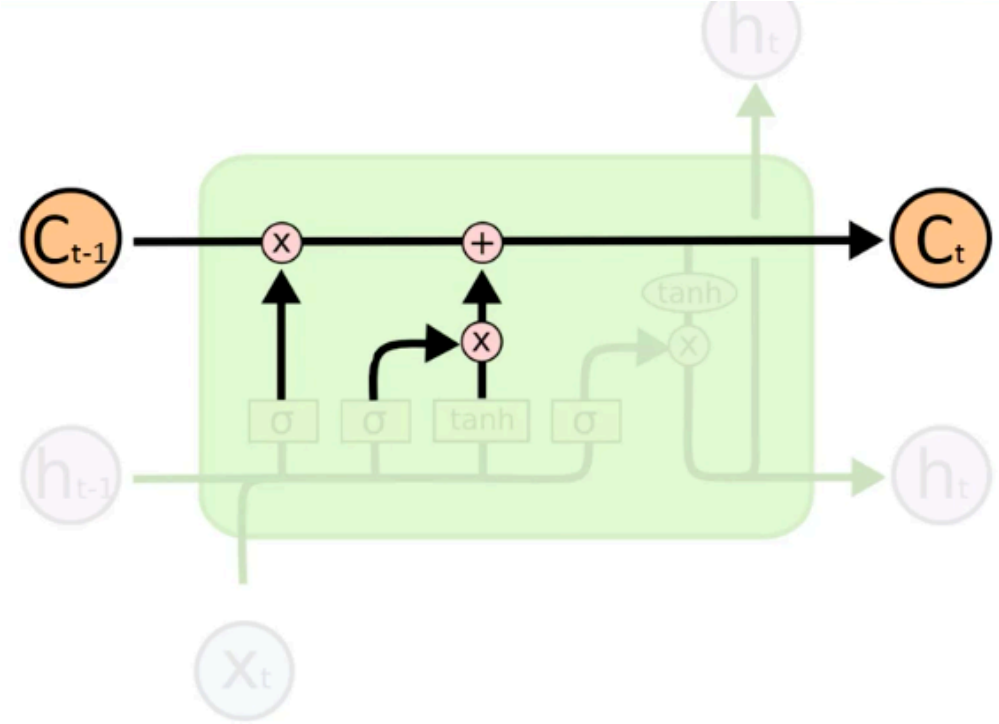
LSTM Step 2

New Value 'X_t' and value from Previous Node 'h_t-1'. They decide whether the memory valve should be opened or closed (Sigmoid). To what extent to let values through (Tanh from -1 to 1).



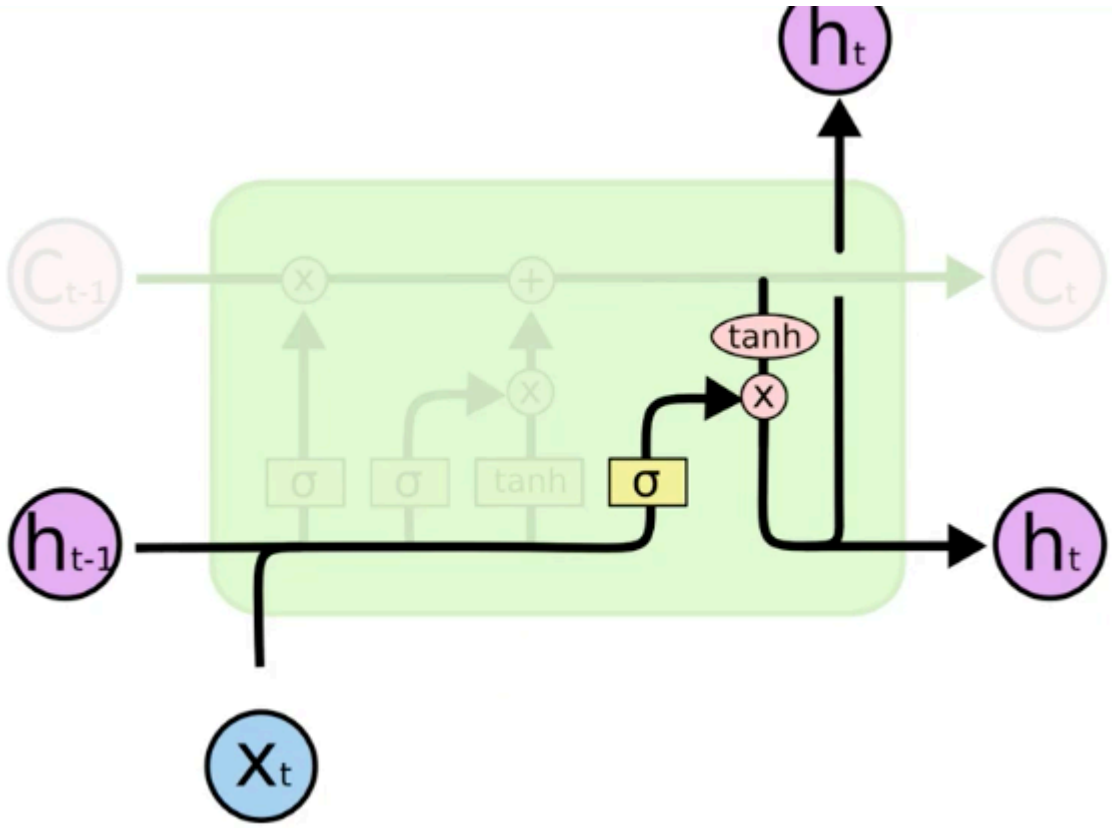
LSTM Step 3

Decide how much a memory cell 'C_t' should be updated from the previous memory cell 'C_t-1'. Forgetting and memory valves were used to decide this. You can update memory completely, not at all or only partially.



LSTM Step 4

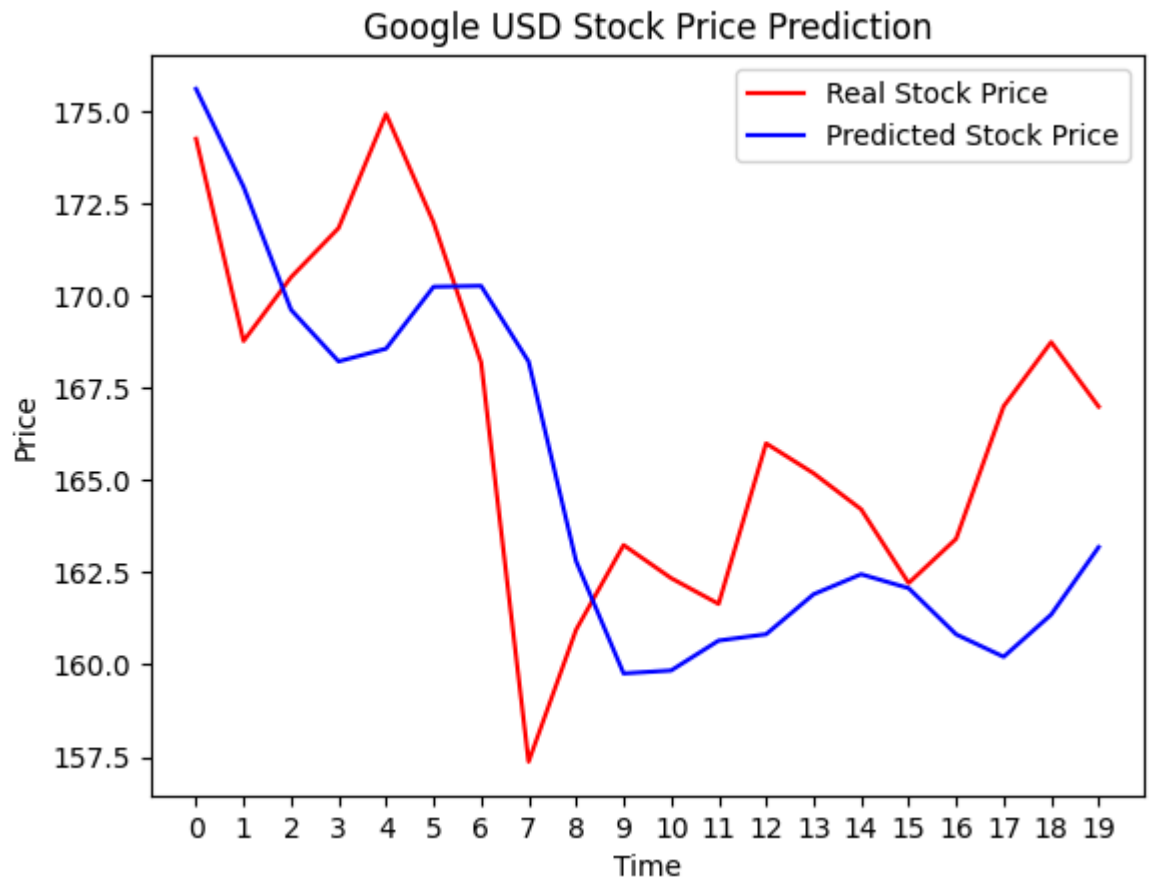
New value 'X_t' and value from previous node 'h_t-1' decide which part of the memory pipeline and to what extent they will be used as an Output 'h_t'.



5. Results

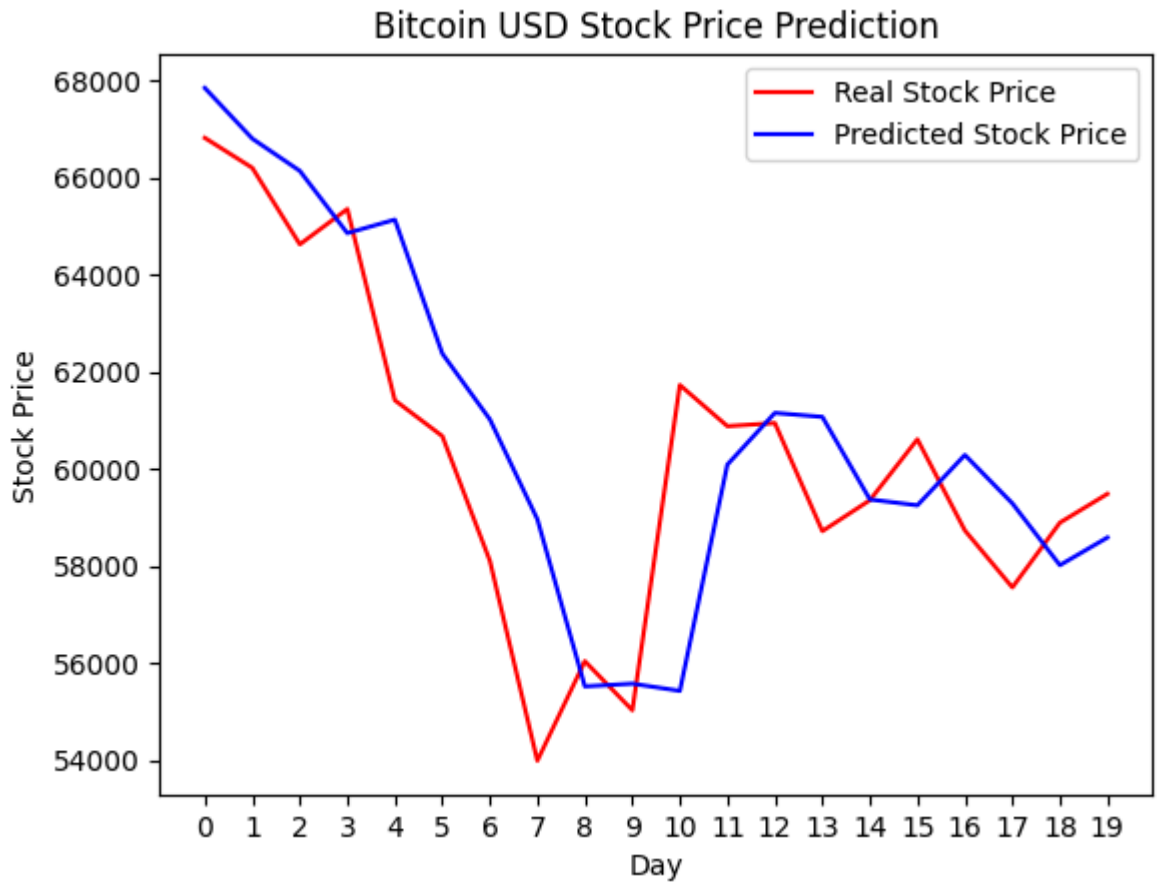
Google

R^2 Score	0.086
MAPE	0.02132, 97.86769%
MSE	0.0013



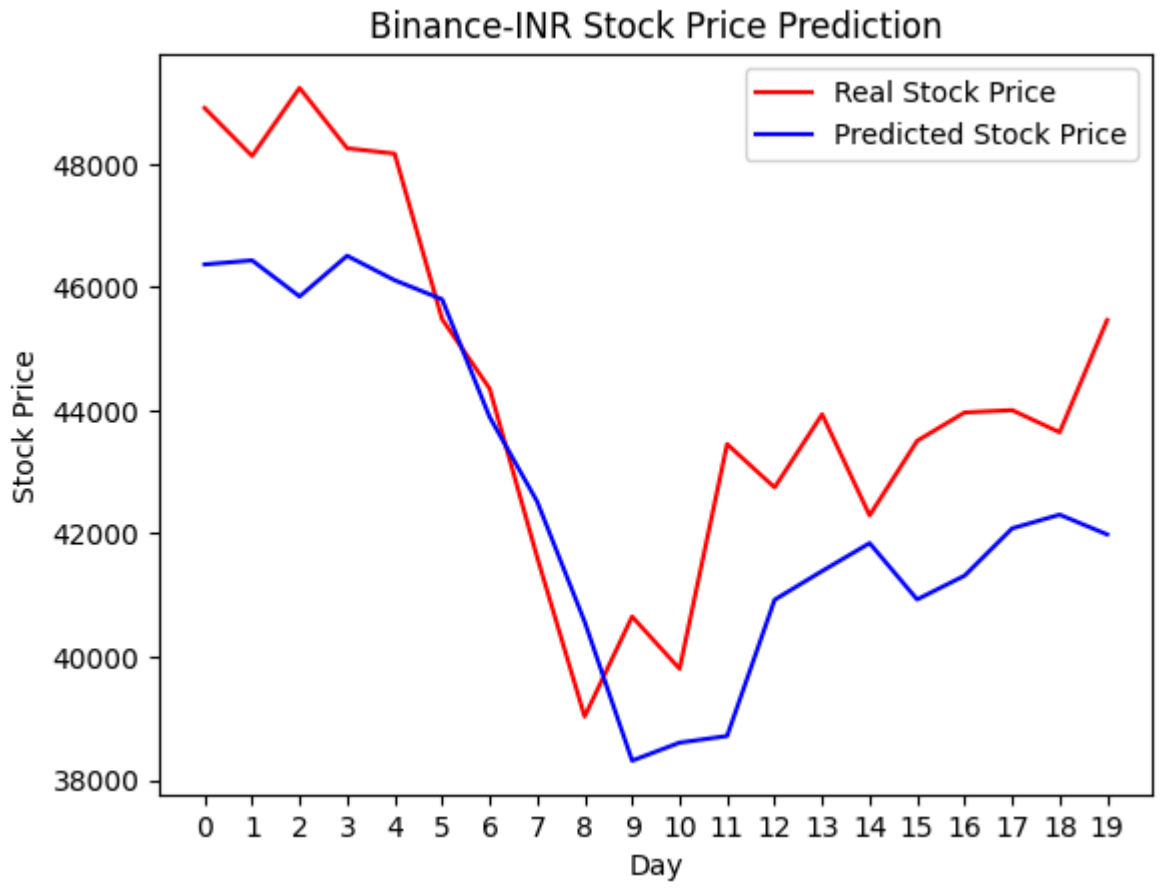
Bitcoin

R^2 Score	0.22
MAPE	0.042, 95.04%
MSE	0.0012



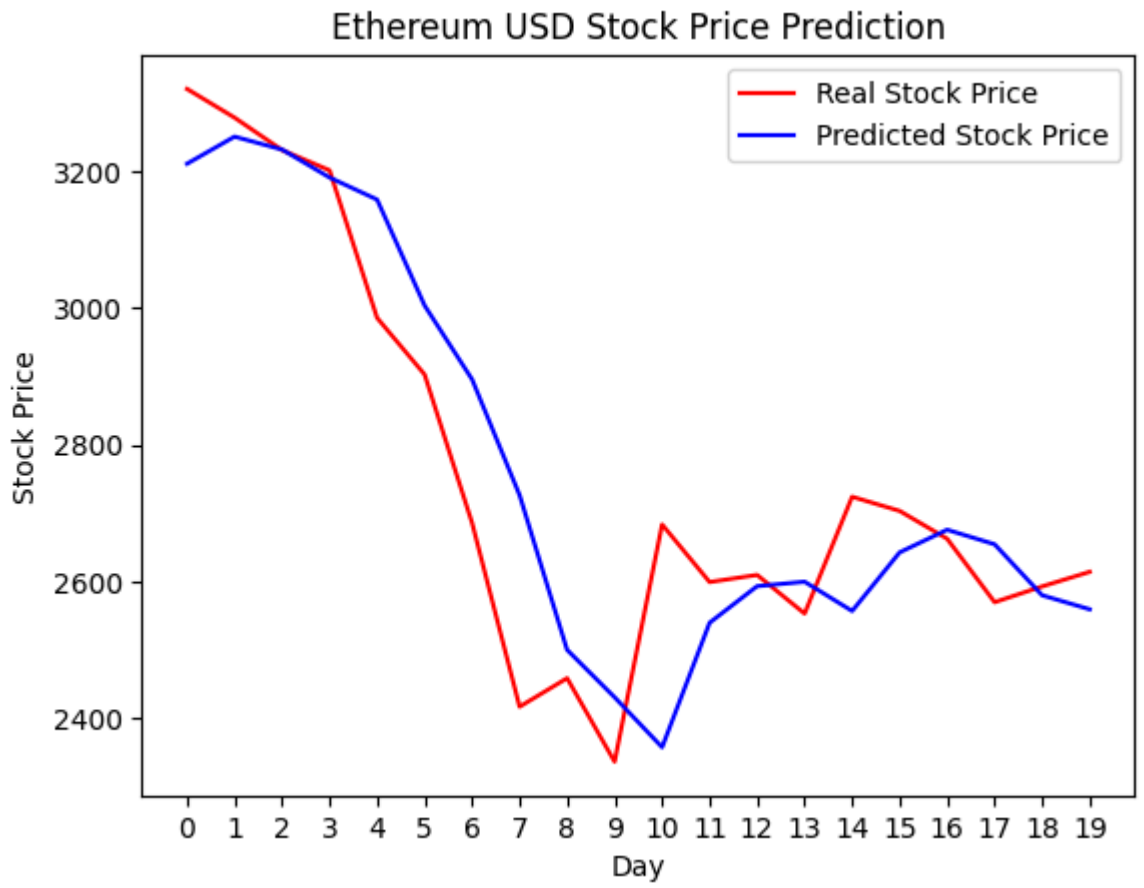
Binance

R^2 Score	0.72
MAPE	0.027, 97.29%
MSE	0.0011



Ethereum

R^2	**0.76**
MAPE	**0.038, 96.15%**
MSE	**0.0010**



7. Conclusion

Summary

This project successfully implemented a Recurrent Neural Network (RNN) to predict the stock for Google and the prices of cryptocurrencies, including Ethereum, Binance, and Bitcoin. The model demonstrated a solid ability to capture the overall trends in price movements, reflecting the sequential dependencies inherent in time-series data. However, while the model accurately followed broader trends, it faced challenges in predicting short-term price volatility and sharp fluctuations. The visualisations of predicted versus actual prices highlighted the model's strengths in trend prediction but also underscored the need for improvements in capturing rapid market changes.

Future Work

Future work could focus on several areas to enhance the accuracy and robustness of the predictions. First, incorporating additional features, such as sentiment analysis from news and social media, could provide more context to the model, potentially improving its ability to predict sharp price changes. Second, experimenting with more advanced architectures like Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRU) could help address the vanishing gradient problem and improve the model's performance on longer sequences. Finally, expanding the dataset to include other cryptocurrencies or financial instruments and testing the model in different market conditions could provide a more comprehensive evaluation of its predictive capabilities.

8. References

https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A Search Space Odyssey. *ArXiv*. https://doi.org/10.1109/TNNLS.2016.2582924

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). On the difficulty of training Recurrent Neural Networks. *ArXiv*. /abs/1211.5063

Karpathy, A., & Johnson, J. (2015). Visualizing and Understanding Recurrent Networks. *ArXiv*. /abs/1506.02078