A Transformer Architecture for Time-Series Data Applied to Stock-Market Closing-Price Prediction

by

Rohit Kuruvilla Sanjay

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Artificial Intelligence)
in the University of Michigan-Dearborn
2024

Rohit Kuruvilla Sanjay

rohitsan@umich.edu

Dedication

With profound emotion and deep gratitude, I dedicate my thesis to the divine guidance of God Almighty, whose boundless favor, grace, wisdom, and strength carried me through every phase of this journey. To my beloved parents, Mr., and Mrs. Sanjay, I extend my heartfelt thanks for their constant support, motivation, and inspirational guidance throughout the thesis process. I am also immensely grateful to my dear sisters, whose unwavering encouragement and steadfast presence kept me focused amidst the challenges.

My heartfelt appreciation goes to my esteemed advisor, Dr. Luis Ortiz, whose guidance, and dedication were pivotal in the success of this thesis. I also wish to acknowledge Dr. Jin Lu for igniting my passion for the chosen topic.

To the friends I've made at the University of Michigan Dearborn and to my cherished friends back home in Dubai, I offer my sincere gratitude. Your support, camaraderie, and encouragement have been invaluable, sustaining me through every obstacle and milestone of this remarkable journey.

Acknowledgments

This thesis would not have been possible without the support and guidance of my advisor Dr. Luis Ortiz; I am forever grateful for your help and contribution. I want to thank my family and friends for their constant support and inspiration for pushing me to do better. A special thanks to Dr. Jin Lu for instilling the desire to pursue a master’s thesis.

Table of Contents

List of Tables

List of Figures

List of Appendices

Abstract

The transformer architecture revolutionized natural language processing (NLP) tasks and gave birth to powerful models such as the generative pre-trained (GPT) model, bidirectional encoder representation from the transformer (BERT), text-to-text transformer. This thesis dives into this unexplored landscape, investigating the potential of transformer-based models for accurately predicting future closing prices of stocks. We propose a novel architecture, derived from the original transformer, specifically crafted for this task. Our approach involves not only building and optimizing this model but also tuning its hyperparameters for each of the four major stock market sectors: technology, finance, pharmaceutical, and FMCG (Fast Moving Consumer Goods). By carefully tailoring the model to each sector's unique characteristics, we aim to maximize its effectiveness and capture nuanced market dynamics. Finally, we put our model to the test, evaluating its performance on unseen data against established time series models such as the long-short-term memory (LSTM) network. This comparative analysis will not only reveal the efficacy of our transformer-based approach but also highlight its potential advantages in terms of accuracy and interpretability compared to traditional methods. Through this exploration, we hope to illuminate a promising new avenue for stock price prediction, offering valuable insights to researchers and investors alike.

Chapter 1 Time Series Data and Stock Market Data

This opening chapter explains what time series data is and its important characteristics. It sets the foundation for understanding how time-series data works. The second objective of this section is to focus on stock market data and the features that enable the prediction of the closing price of individual stocks.

1.1 Time Series Data

Time series data is a feature measurement at a given timestamp. It is a key component in data analysis in many industries. Time series data is useful as it can answer questions about trends, patterns, and correlations over time. The ability to see how a feature changes over time is a powerful insight into many business operations and equipment. The following are the characteristics of time series:

- **Trend:** A trend is a gradual shift in a series over time, either up or down. It can be local or global, and a series may show both or neither. [1]
- **Seasonal Cycle:** A seasonal cycle is a repetitive predictable pattern formed in a period inside the time series data. Seasonality cycles are tied to intervals of your series, for example, monthly data typically cycles over quarters and years. [1]
- **Non-Seasonal Cycle:** A non-seasonal cycle is a repetitive and unpredictable period in the time series data. Some time series however show cyclic behavior, but the periodicity of the cycle varies over time making it difficult to predict when a high or low will occur. [1]

- **Pulse and Steps:** Pulse refers to temporary shifts in the data, while step denotes permanent changes in the data. It's crucial to find plausible explanations when observing these changes. [1]
- **Outliers:** Shifts in the level of a time series that cannot be explained are referred to as outliers. These observations are inconsistent with the remainder of the series and can dramatically influence the analysis and, consequently, affect the forecasting ability of the time series model. [1]

Figure 1-1 (next page) showcases the S&P500 closing price decomposition into trend, seasonal component, residual component. The x-axis represents the year while the y-axis for trend graph represents closing price value, for seasonal component represent movement of stock, for residual component shows randomness values in the stock.

1.2 Stock Market Data

In the financial market, market data like price, stock volume, and other related data are financial instruments reported by a trading venue such as a stock exchange. Market data helps investors and traders understand the market and make informed decisions on whether to buy or sell a financial instrument like stock. Some of the main examples of financial instruments are stocks, equities, fixed-income products, derivatives, and currencies. Each data point that is collected concerning stock market data is time-dependent, this implies that at each second there is a new data point reflecting a particular stock feature and each data point is a result of all the previous stock market data points, the economic condition and market rhymes. Trend analysis enables understanding of the long-term movements, and differentiation between the upward slope signifying a bullish market and downward movement as a bearish market. While stock market data is a more focused

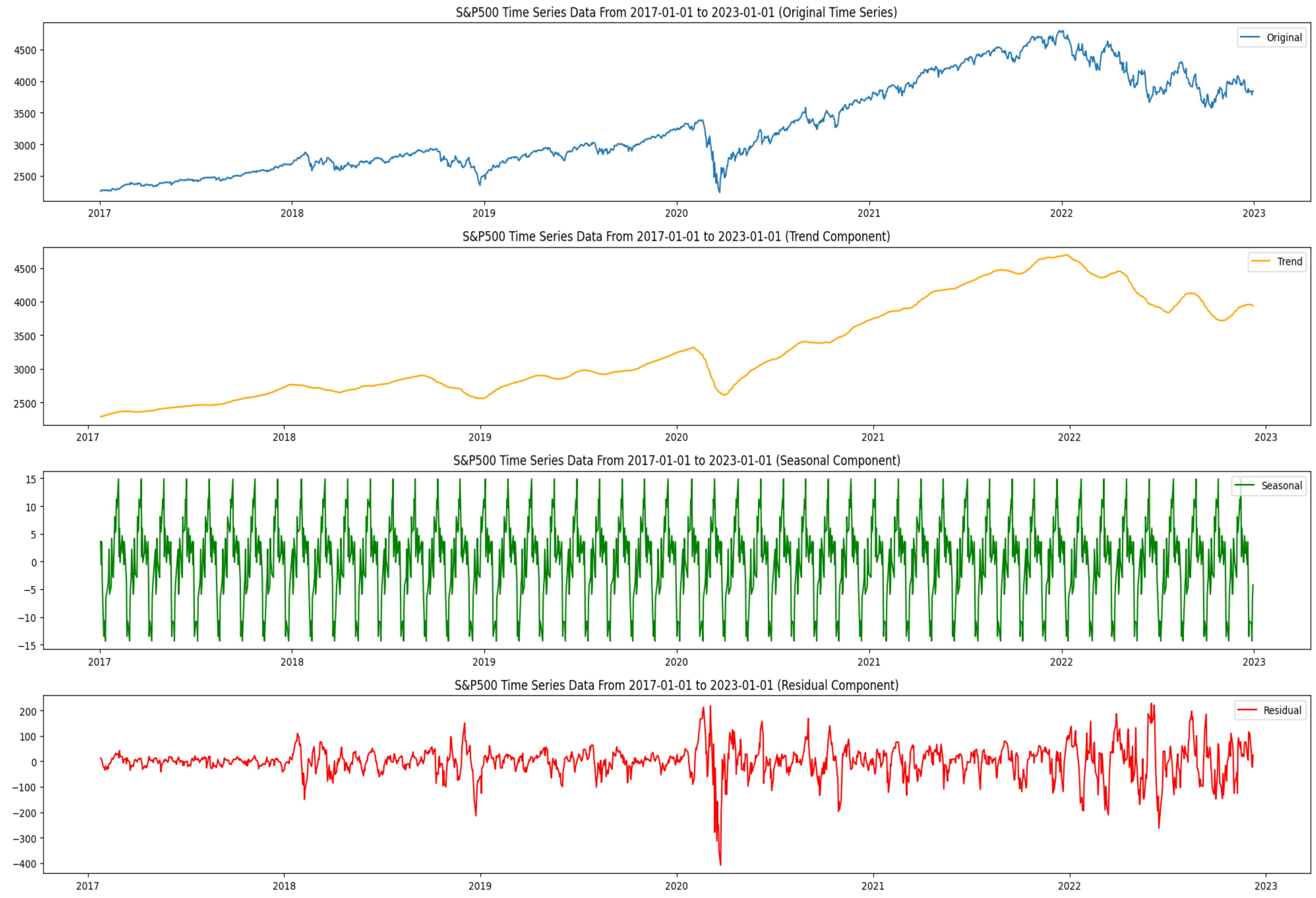
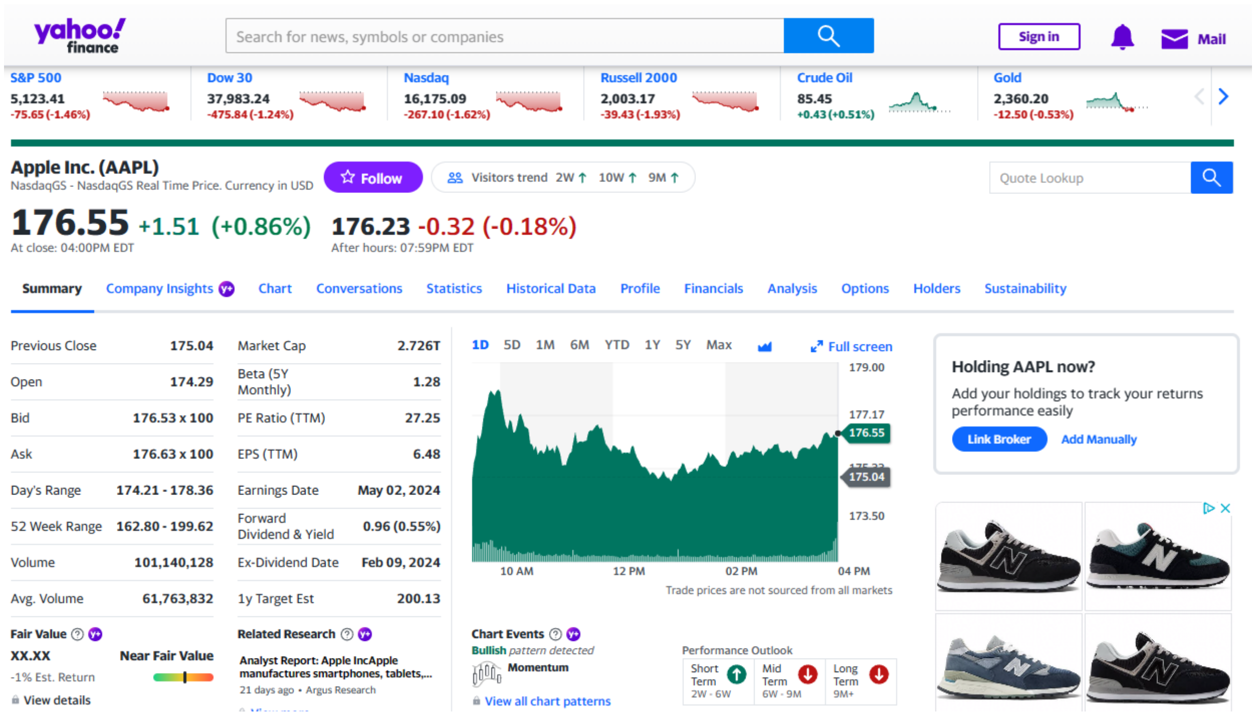


Figure 1-1: Time Series Decomposition using StatsModel Python Package

version of the market data in general, it is important to understand that the stock market is a victim of similar uncertainty and unpredictability. [2]

Each country typically has its stock market exchange, responsible for regulating, monitoring, and evaluating the traded stocks. Leading examples include the NYSE and NASDAQ in the US and the BSE and NSE in India. These exchanges hold a wealth of crucial information about the companies listed, including trading data, company profiles, regulatory filings, and market indices. Below are some of the information that can be found on stock while trying to model, predict, forecast, or decide on how to handle stocks:

- **Ticker Symbol:** A short, unique set of letters or letters and numbers (generally 1-4 characters) used to identify a specific company's stock on an exchange.
- **Opening Price:** The first price at which a particular stock trades on an exchange on a given trading day.
- **Closing Price:** The last price at which a particular stock trades on an exchange on a given trading day.
- **High Price:** The highest price at which a particular stock trades on an exchange on a given trading day.
- **Low Price:** The lowest price at which a particular stock trades on an exchange on a given trading day.
- **Volume:** The total number of shares of a particular stock that have been traded on an exchange on a given day.
- **Bid Price:** The bid price is the highest price that a buyer is willing to pay for a share of a stock at any given time.



S&P 500	Dow 30	Nasdaq	Russell 2000	Crude Oil	Gold
5,123.41	37,983.24	16,175.09	2,003.17	85.45	2,360.20
-75.65 (-1.46%)	-475.84 (-1.24%)	-267.10 (-1.62%)	-39.43 (-1.93%)	+0.43 (+0.51%)	-12.50 (-0.53%)

Apple Inc. (AAPL)
NasdaqGS - NasdaqGS Real Time Price. Currency in USD

176.55 +1.51 (+0.86%) **176.23** -0.32 (-0.18%)
At close: 04:00PM EDT After hours: 07:59PM EDT

Previous Close	175.04	Market Cap	2.726T
Open	174.29	Beta (5Y Monthly)	1.28
Bid	176.53 x 100	PE Ratio (TTM)	27.25
Ask	176.63 x 100	EPS (TTM)	6.48
Day's Range	174.21 - 178.36	Earnings Date	May 02, 2024
52 Week Range	162.80 - 199.62	Forward Dividend & Yield	0.96 (0.55%)
Volume	101,140,128	Ex-Dividend Date	Feb 09, 2024
Avg. Volume	61,763,832	1y Target Est	200.13

Fair Value
XX.XX **Near Fair Value**
-1% Est. Return
View details

Related Research
Analyst Report: Apple IncApple manufactures smartphones, tablets,...
21 days ago • Argus Research

Trade prices are not sourced from all markets

Chart Events
Bullish *pattern detected*
Momentum
View all chart patterns

Performance Outlook

Short Term 2W - 6W	Mid Term 6W - 9M	Long Term 9M+
↑	↓	↓

Holding AAPL now?
Add your holdings to track your returns performance easily

Figure 1-2: Yahoo Finance Dashboard (Image taken from [27])

- **Ask Price:** Conversely, the ask price is the lowest price that a seller is willing to accept for a share of a stock at any given time.
- **Spread:** The spread is the difference between the bid price and the ask price.

Figure 1-2 is a screenshot of the yahoo finance dashboard with real time Apple stock value movement. The left side of the dashboard indicates key information such as opening price, bid price, ask price, 52 weeks average and so on while the center is a real time graph plotting the movement of the Apple stock every second.

Chapter 2 Stock Market Prediction Models

The stock market prediction model attempts to determine the future value of a company stock or other financial instrument through different prediction methodologies. Furthermore, this chapter depicts different approaches and techniques to predict stock market closing prices.

2.1 Types of Prediction Methods:

The ability of a model or analysis to be able to predict future prices dictates the future decisions made by investors and traders on whether to buy or sell a particular stock. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. The three main categories of these methods are fundamental analysis, technical analysis, and machine learning models. [3]

- **Fundamental Analysis:** This strategy analyzes past performance and determines the true value of the stock, comparing it with the market value to realize a potential undervaluation for long-term investment. [3]
- **Technical Analysis:** This strategy primarily depends on analyzing past prices and trends to anticipate future stock price movements, making it predominantly used for short-term investing. [3]
- **Machine Learning Models:** Models such as regression, CNN, RNN, LSTM, and GRUs have been able to leverage deep learning to understand stock market data and predict future movement. [3]

2.2 ARIMA Models

Autoregressive Integrated Moving Averages (ARIMA) model are used for time series forecasting and predictions. Auto-Regressive models or AR models use past stock price data to predict future prices while Moving Averages or MA models use past forecasting errors to predict future prices. A key component required to apply the ARIMA model on time series data is to have stationary data. Data can be called stationary only if the mean and variance are consistent throughout the dataset. Most of the real-world time series data are not stationary and hence need to be transformed for them to be compatible with these kinds of models. The transformation process is called differencing where the d^{th} difference is taken from the series until the data is stationary. An ARIMA model can be expressed using the following notation, ARIMA(p,d,q), where p stands for autoregressive order, d stands for the differencing degree and q stands for moving averages order [4].

2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) (refer figure 2-1) are one of the most prominent models in deep learning especially for computer vision tasks. Their ability to capture spatial features and extract information has accounted for high accuracy and performance in handling time series data tasks. To make CNN compatible with time series data, it is crucial to transform the data into batches with a specific window size that contains past information. This helps the model understand the temporal context for predicting future values. CNN uses 1D filters across each window to extract relevant features, these features are then passed into a pooling layer that helps

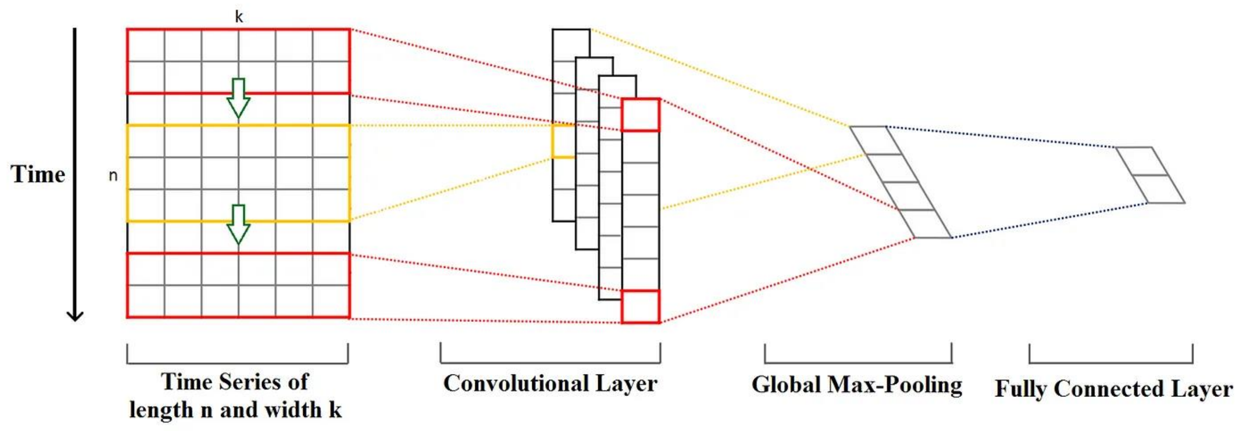


Figure 2-1: CNN's Applied on Time Series Data (Image taken from [28])

aggregate the information extracted from the first step. Finally, the pooled information is passed into a linear network that processes this information and returns a future value. [5]

2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (refer figure 2-2) are made up of many artificial neurons that work together to perform tasks that are challenging and complex. The arrangement of the neurons in RNN is as input, hidden, and output layers. The RNN first passes the data into the input layer which usually consists of a feature extraction layer, the extracted information is passed into the hidden layer which consists of a series of linear layers and activation functions, and finally, the output is predicted using the output layer. A key distinguishing factor of the RNNs is that the model has a self-looping where the hidden layer can remember and use previous inputs for future predictions in its memory components. Just like a simple ANN (Artificial Neural Network) which uses backpropagation to adjust its weights, for RNN, the model uses a similar approach called back-propagation through time or BPTT. BPTT rolls back the output to the previous time step and recalculates the error rate. This way, it can identify which hidden state in the sequence is causing a significant error and readjust the weight to reduce the error margin. [6]

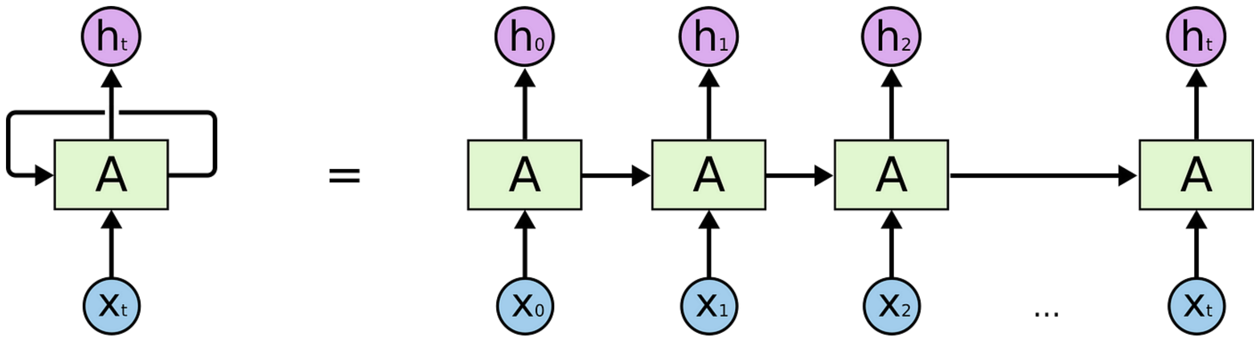


Figure 2-2: RNN's Architecture (Image taken from [7])

2.5 Long Short-Term Memory Models

Long short-term memory models or LSTM models (refer figure 2-3) are part of the RNN's family of deep learning models. LSTM's tackles vanishing gradient problem by introducing gated units that selectively retain or discard information depending on the task. There are 3 gates and 1 memory cell in the LSTM architecture, forget gate, input gate, output gate and candidate memory cell respectively. The input gate controls the follow of information within the model. It consists of a sigmoid activation function followed by a pointwise multiplication. The forget gate controls the flow of information in the candidate memory cell, this is done by using sigmoid activation function and pointwise multiplication. The output gate controls the flow of information leaving the model at each time step, this is done by sigmoid activation function and pointwise multiplication. The working of these gates together makes LSTM well suited for time series tasks as it can learn and remember complex temporal patterns and make future predictions.

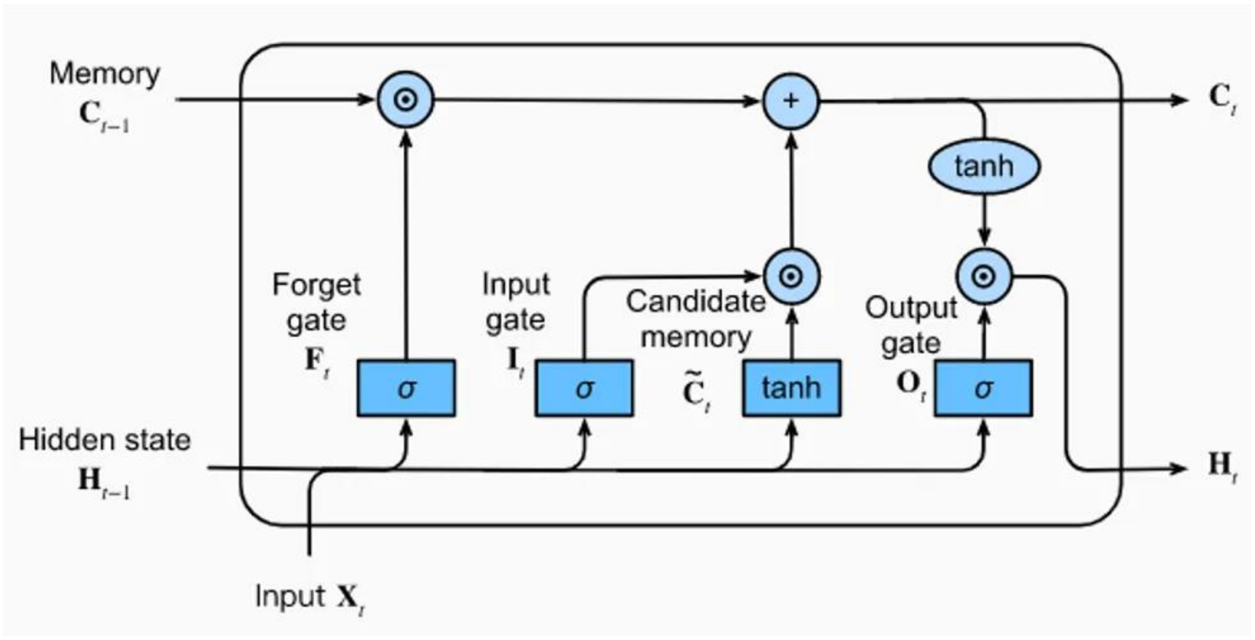

Memory $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$

Figure 2-3: LSTM Architecture (Image taken from [8])

2.6 Relevant Work

Several studies have explored various methodologies for predicting stock market closing prices, aiming to navigate the inherent uncertainties of financial time series data. Samarawickrama and Fernando (2023) investigated RNNs, LSTMs, and gated recurrent units (GRU), for predicting the closing price of three Sri Lankan companies. Despite its promise, their study found a multilayer perceptron (MLP) outperformed all RNN models. Interestingly, GRUs exhibited higher errors compared to other RNN architectures [9]. In contrast, another study by Putu Candra and Ni Luh Wiwik (2023) compared the performance of support vector machines (SVMs) and linear regression (LR) in predicting daily stock prices for an Indonesian company from 2014 to 2023. The analysis based on various training and testing data splits, concluded that LR outperformed SVMs where the focus remained on publicly traded companies in the S&P500 [10]. Ravikumar and Saraf (2023) explored predicting closing prices using various regression and classification algorithms. After extracting additional features like volatility and momentum from historical data, their study found polynomial regression achieving the highest accuracy (91.45%), followed by linear regression (82%) and SVM (87%) [11]. These diverse studies highlight the ongoing exploration of different approaches for stock price prediction. While RNNs hold promise, MLPs and traditional statistical models like LR may offer competitive performance depending on the chosen dataset and features.

Transformer-based architectures have emerged as promising tools for capturing the complex dynamics of financial time series. Mirjebreili and Solouki proposed a multi-task transformer trained on Iranian stock market data. Their model employed quintuple barrier labeling and a custom purge k-fold validation approach to address data imbalance and time-dependency issues. While achieving moderate accuracy, their work demonstrates the potential of transformers for multi-class stock price prediction [12]. Costa and Machado compared transformers to traditional ARIMA and LSTM models for predicting Ibovespa index stock prices. Their transformer architecture outperformed both baselines in 60% of the tests, showcasing its ability to capture long-range dependencies within the data. This study emphasizes the potential of transformers to surpass established methods in specific contexts [13]. Yawei Li and Xingua Liu incorporated transformers and attention networks to predict stock movements based on tweets and historical prices. Their TEANet model leveraged text embeddings and temporal attention to integrate sentiment analysis with price data. This approach achieved improved accuracy compared to other models, highlighting the value of incorporating additional information sources like social media sentiment [14]. Nadeem Malibari and Iyad Katib were the first to apply a transformer architecture for predicting future prices on the Saudi Stock Exchange. Their model, inspired by the vision transformer, achieved an impressive 90% accuracy, demonstrating the effectiveness of transformers in specific emerging markets. This study underscores the need for further research into tailoring transformer models to different market characteristics [15]. Tashreef Muhammad and Anika Bintee explored a transformer-based deep learning model for the Bangladesh Stock Market. Their study trained separate models for eight companies, achieving varying levels of success. This highlights the potential but also the challenges of applying transformers to individual stocks, where data availability and model complexity can play a role [16]. Harsimrat Karley investigated transformers combined with sentiment analysis for predicting normalized opening prices. Their model uses technical indicators and news headlines to capture both quantitative and qualitative market signals. This work highlights the potential of transformers to integrate diverse data sources for improved prediction [17]. Chaojie Wang and Shuqi Zhang implemented an encoder-decoder transformer to predict closing prices for three major international indices. Their model achieved competitive performance compared to established benchmarks. [18]. A comparative paper was published by Ailing and Muxi discussing the validity of transformers with respect to time series data. To establish their claim, a set of simple one-layer linear models are compared with different transformers across nine different datasets. The paper showcases that all the linear models outperform the transformer models by large margins and this line of research needs to be revisited [19].

In conclusion, these studies reveal the ongoing exploration of transformers for stock price prediction. While transformers demonstrate promise, their performance can vary depending on the specific market, data characteristics, and chosen architecture. Further research is needed to fully understand their potential and limitations in this complex domain. Additionally, combining transformers with other approaches like sentiment analysis or technical indicators holds promise for further enhancing prediction accuracy.

Chapter 3 The Transformer Architecture

A transformer is a type of neural network architecture that consists of an encoder and a decoder working together with attention layers to extract relevant information and output sequence of results [20]. Transformers has seen tremendous success with the realm of natural language processing tasks especially with respect to machine translation and sentence generation. Below subchapters dive deep into the different layers and blocks inside a transformer.

3.1.1 Tokenization

Tokenization plays a crucial role in transforming discrete tokens into meaningful representations. These vectors act as a bridge between the surface features of language and the underlying semantic complexities. Assuming that a huge corpus of text is taken into consideration, this corpus is divided into chunks of smaller texts of fixed size and fed into a pretrained or learned model that can convert the text into a series of tokens. Some of the tokenization techniques are, word tokenization where text is broken down into individual words. Another tokenization method is character tokenization breaks down a word into individual characters, this method is effective in tackling problems where language does not have clear word boundaries or tasks that require granular analysis. Finally, sub-word tokenization is a method that finds the grey region between the two previously mentioned methods. This method breaks down text into units that might be larger than a single character but smaller than a full word [21].

3.1.2 Word Embedding

The embedding layer occupies a crucial role in converting discrete word tokens into continuous vector representations suitable for further processing within the transformer. After tokenization, the tokens need to be represented in a higher dimensional form. There are two ways to achieve this, pretrained or learned embeddings. Essentially for the pre-trained embeddings, it performs a

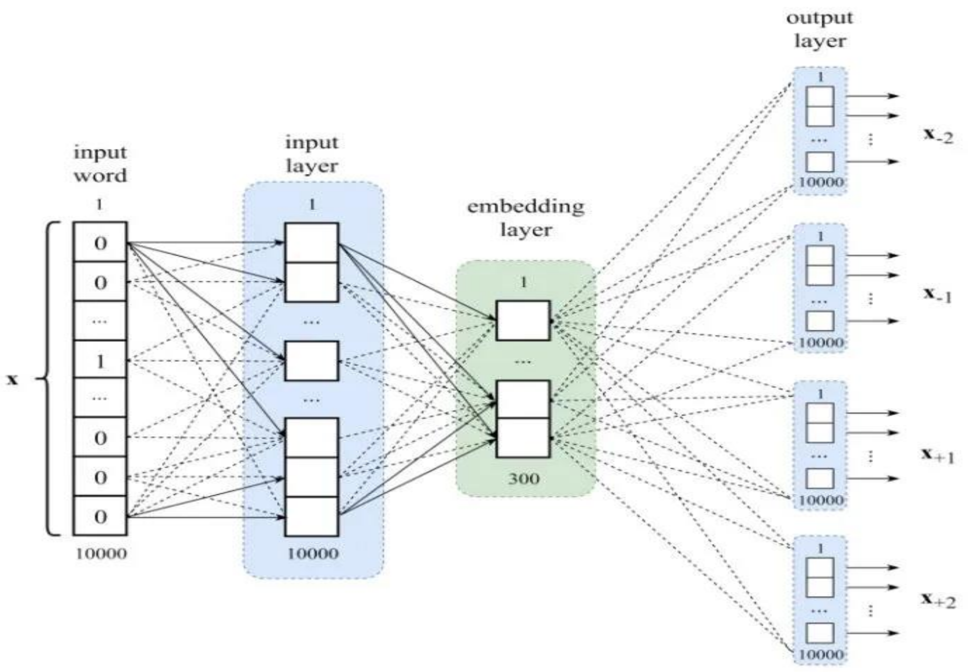


Figure 3-1: Word Embedding layer (Image taken from [29])

lookup operation on a pre-trained weight matrix (the embedding matrix) where each row denotes a unique word in the vocabulary and each column corresponds to a specific dimension in the embedding space. This process maps the integer indices representing words to dense vectors of fixed dimensionality, capturing semantic and syntactic relationships between words in a numerical format. Learned embeddings use neural networks that adopt a data-driven approach, directly learning word representations from the training data itself. These layers typically utilize embedding matrices, learnable weight matrices where each row encodes a unique word's vector representation. Through backpropagation and gradient descent, the model refines these vectors, optimizing their ability to capture word relationships between each other [22] .

3.1.3 Positional Encoder

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Figure 3-2: Positional Encoding Formula (Image taken from [20])

Natural language processing tasks mainly require sequence processing, understanding the relative positioning of words is crucial to create more meaningful sentences. In transformers, these encodings are done using learned vectors crafted through sine and cosine functions, which function as multi-dimensional embeddings within each element's representation. This encoding scheme relies on two key properties of sine and cosine waves [20]:

- Frequency-Based Distinction: Trigonometric functions oscillate uniquely which enhances the sensitivity of the model to local and global relationships.
- Dimensional Orthogonality: Sine and cosine in linear combinations, exhibit orthogonality. This makes sure the information encoded is independent and no information leak takes place during training.

Figure 3-2 is taken out of the 2017 paper "Attention is all you need", where pos are the position, i is the ith dimension of the embedded vector, and d_{model} represents the model dimension of the transformer. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to 10000 · 2π.

3.1.4 Attention Layer

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio at the ICLR 2015 conference showcased the first working attention mechanism [23]. The attention mechanism consists of three main matrices:

$$Attention(Q, K, V) = Softmax(\frac{Q.K^T}{\sqrt{d_k}})V$$

Figure 3-3: Equation of Attention Layer (Image taken from [20])

- Queries (Q): These vectors represent the network's current target, often corresponding to a specific word or element.
- Keys (K): Each position in the sequence possesses a unique identifier encoded in the key vector.
- Values (V): These values hold raw information that might be seen during querying.

Figure 3-3 showcases the scaled dot-product computation formula for calculating attention score by using the Query, Key, and Value, the model uses this matrix to find a relationship between the Key-Value pair and then uses the query to search for a similar vector. In the context of similarity computation, the neural network utilizes a dot-product operation to measure the alignment between the query vector and each key vector within the sequence. This process captures the degree of association or similarity between the current focus and individual elements in the sequence. To address potential instability during training, a scaled relevance scoring mechanism is employed. The raw dot product is scaled by the square root of the key vector's dimensionality, thereby mitigating the impact of large dimensions, and ensuring manageable gradients for stable training. Once the scaled similarly scores are returned, the SoftMax function is applied to convert the values into attention weights. This ensures that the attention weights sum up to 1 and represents a probability distribution over the key vectors. Finally, these attention weights are multiplied with the value matrix which allows the attention mechanism to selectively aggregate information based

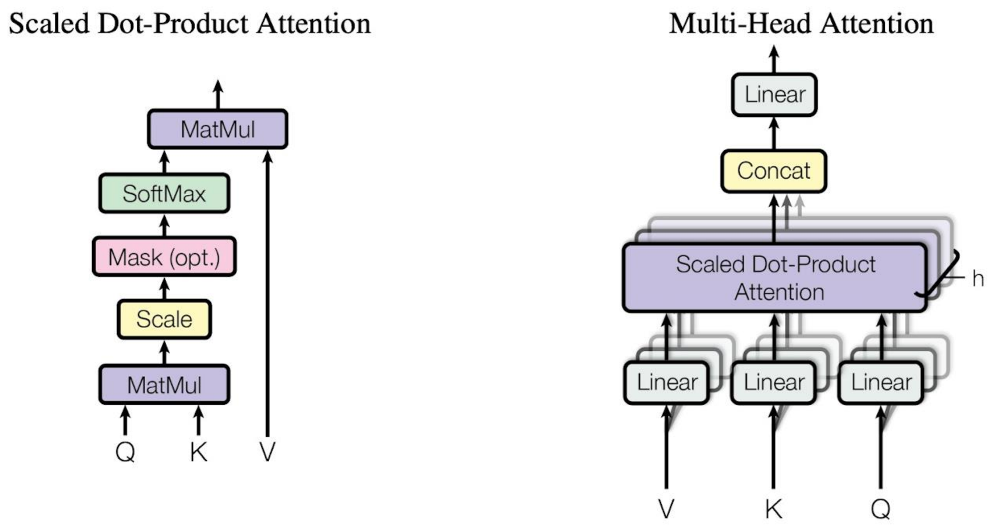


Figure 3-4: Multi-Head Attention Mechanism (Image taken from [20])

on relevance, producing contextual representation that captures the most important features of the given input [20].

Multihead Attention (refer figure 3-4) in a transformer model allows capturing different relationships between words simultaneously by performing multiple attention operations in parallel. This is done by dividing the query, key, and value matrices into multiple heads and computing individual attention scores for each head, later concatenated, and linearly transformed for output. This mechanism helps diverse patterns and dependencies within an input sequence which leads to improved performance.

3.1.5 Feed Forward Layer

In a Transformer architecture, the feed-forward network (FFN) adds two vital functionalities, non-linear transformation, and feature enrichment. This is achieved by using a series of non-linear activation functions and linear layers. Non-linear activation functions (e.g., ReLU), inject non-linearity into the network, allowing it to capture intricate patterns and interactions beyond the capabilities of attention alone, similarly, feature enrichment is also achieved by transforming the attention-derived representation through non-linear activation functions and linear layers. FFNs can lead to the creation of new features, the modulation of existing ones, and ultimately a richer understanding of the sequence.

3.1.6 Output Layer

The output layer consists of mainly 2 layers, a linear layer and a SoftMax layer. The linear network, at its core, is a single layer of interconnected neurons. The final hidden state vector generated by the decoder, which encapsulates the processed input sequence, and project it onto a probability distribution for each possible output token. This distribution represents the model's confidence in each potential output, allowing it to account for uncertainty in situations where clear predictions are not always possible. The linear layer will project the final output vector in the dimension of the vocabulary size used for the model. The output of the linear layer is then fed into a SoftMax layer to extract the raw probability distribution of the output vector. This distribution conveys the model's confidence in each potential output, not just as a number but as a percentage ranging from 0 to 1. This information is then utilized in various ways depending on the specific task [20]. Figure 3-5 is a snippet of the original transformer block diagram with encoder network on the left and decoder network on the right.

3.2 Types of Transformers

Transformers are known for their heavy use of computational resources and slow training times. As a result, this has led to the introduction of encoder only models or decoder only models. This section an explanation of the advantages and disadvantages of each of these models are provided.

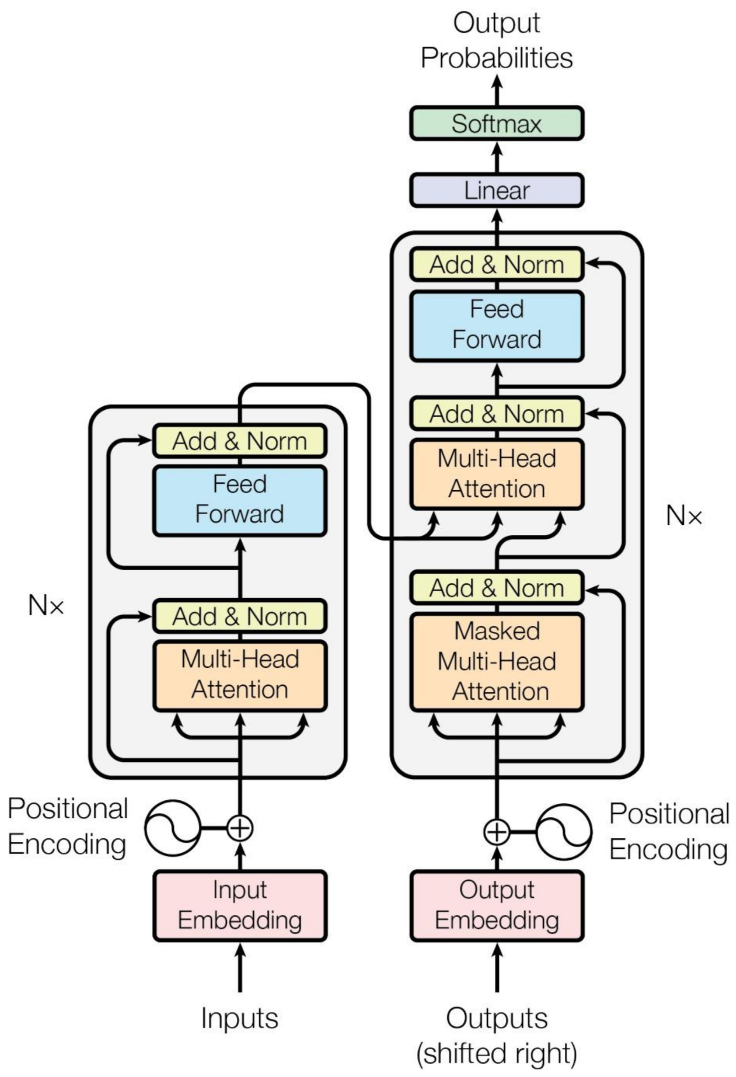


Figure 3-5: Block Diagram of the Original Transformer (Image taken from [20])

3.2.1 Encoder Only Models

Transformer encoder only models (refer figure 3-6) are a specialized form of the transformer where the decoder part of the transformer does not exist. This modification is done to offer advantages over the original architecture. One prominent advantage is computational efficiency. By removing the decoder, the computational load decreases and makes them better for use cases where resources for training are limited. Another advantage of encoder only models is that due to the absence of the decoder, there is more transparency in the interpretability of the model and the processing of

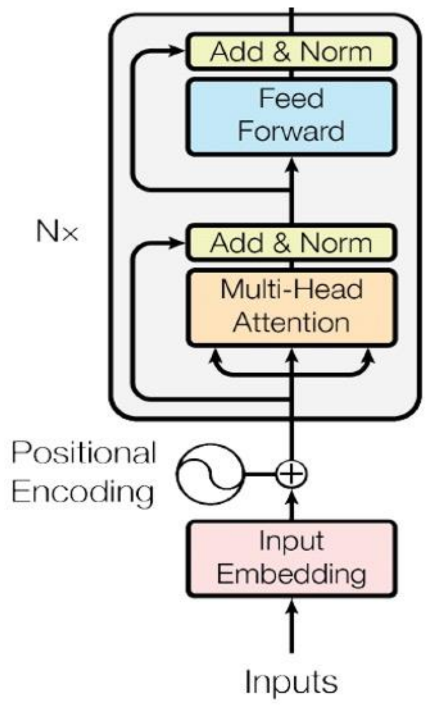


Figure 3-6: Block Diagram for Encoder Only Models (Image taken from [20])

the data. The model performs extremely well for feature extraction tasks and to understand incoming data. Despite such advantages, they have their own set of limitations. One notable drawback is the inability of the model to produce sequences as the decoder part is held responses to for such generation in the traditional tasks. One of the most successful encoder only models is the BERT model released by Google [24].

3.2.2 Decoder Only Models

Decoder only models (refer figure 3-7) are a specialized version of the transformer such that the encoder part of the original transformer is missing from the architecture. This approach offers different advantages over the original architecture such as reduced computational complexity and cost. By excluding the encoder, the model becomes lightweight and has faster inference during the production of the model in real-time applications. However, due to the missing presence of the encoder, the decoder lacks in understanding or extracting features of the data and needs assistance

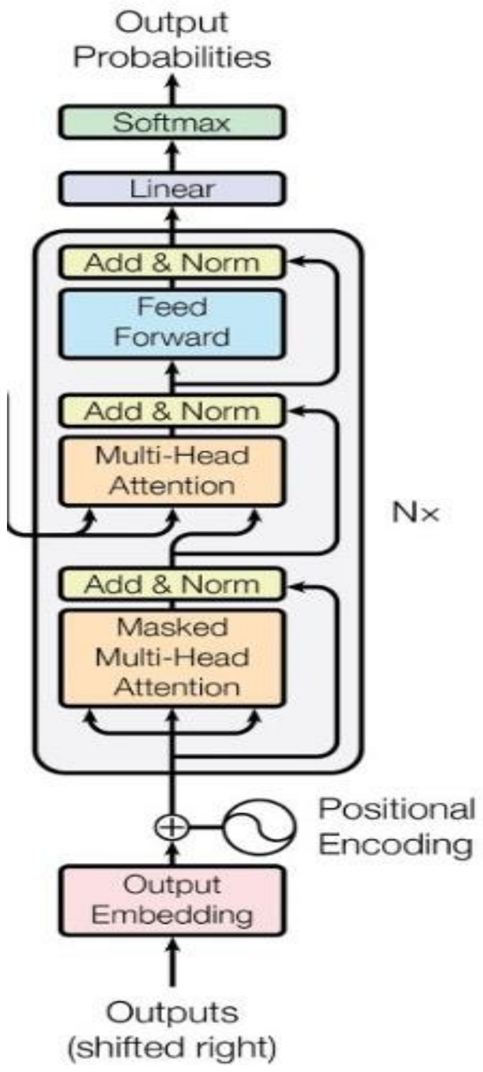


Figure 3-7: Block Diagram for Decoder Only Models (Image taken from [20])

to process such data. Decoder only models are most used where the primary objective is sequence generation such as text generation, image captioning, etc. One of the most successful decoder only models is are GPT model created by OpenAI [25].

3.3 Proposed Time Series Transformer

The proposed time series transformer takes inspiration from encoder only models. The encoder block is coupled with a projection block, and positional encoding block at the start to represent the

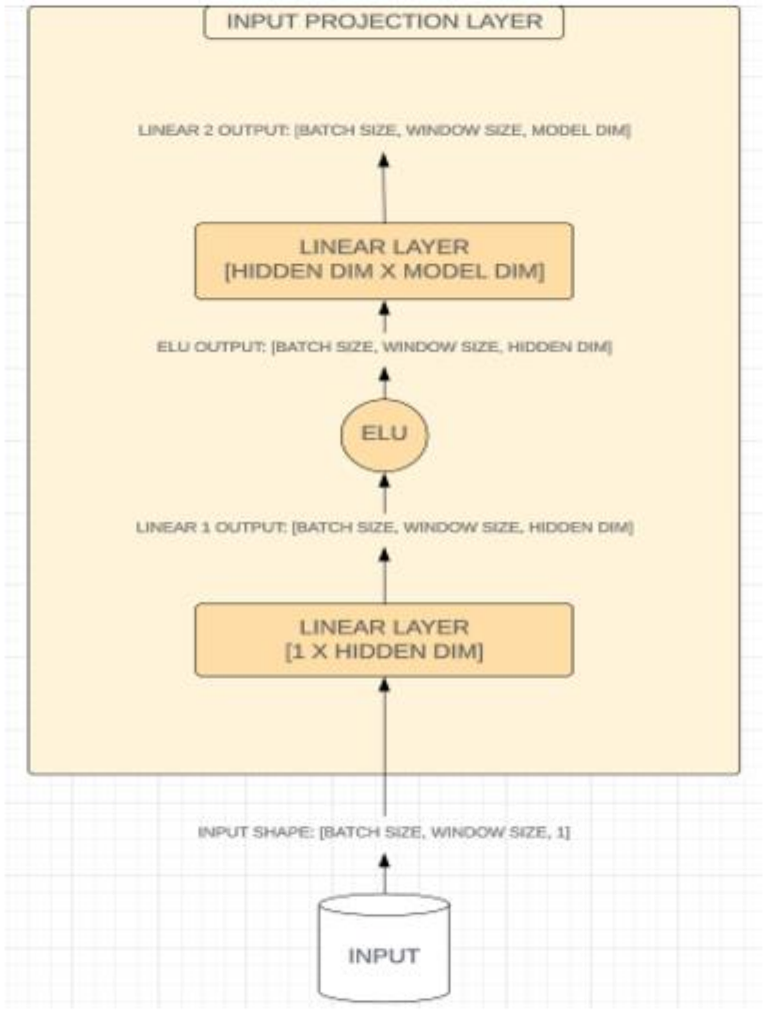

INPUT PROJECTION LAYER
LINEAR 2 OUTPUT: [BATCH SIZE, WINDOW SIZE, MODEL DIM]
LINEAR LAYER
[HIDDEN DIM X MODEL DIM]
ELU OUTPUT: [BATCH SIZE, WINDOW SIZE, HIDDEN DIM]
ELU
LINEAR 1 OUTPUT: [BATCH SIZE, WINDOW SIZE, HIDDEN DIM]
LINEAR LAYER
[1 X HIDDEN DIM]
INPUT SHAPE: [BATCH SIZE, WINDOW SIZE, 1]
INPUT

Figure 3-8: Block Diagram of Input Projection Block

data in higher dimensions. Secondly, the output of the encoder block is fed into an output block which is expected to predict future closing price of different stocks.

3.3.1 Input Projection Block

The input project block is the first layer in the time series transformer model. The layer consists of 2 linear layers and a single activation function called exponential linear unit (ELU). The combination of linear layers and ELU activation function helps project the data to the required dimension while learning data relationships and complexities. ELU activation function tackles the problem of dying ReLU by allowing negative values, this helps prevent neurons from

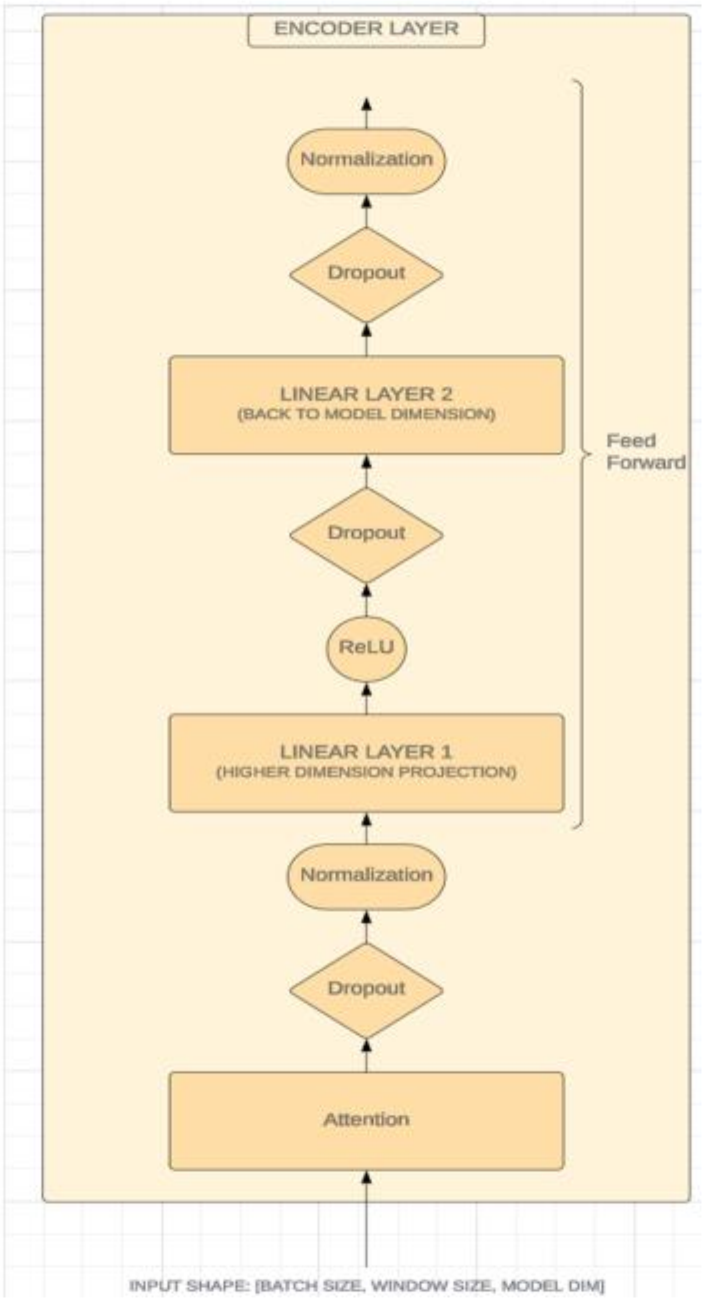


Figure 3-9: Block Diagram for Encoder Block

becoming inactive during training [26] . In figure 3-8 (previous page), the data is first introduced into the block with a shape of batch size x window size x 1, passed onto an activation function (ELU), projected into a hidden dimension, and finally converted to a shape of batch size x window size x model dimension.

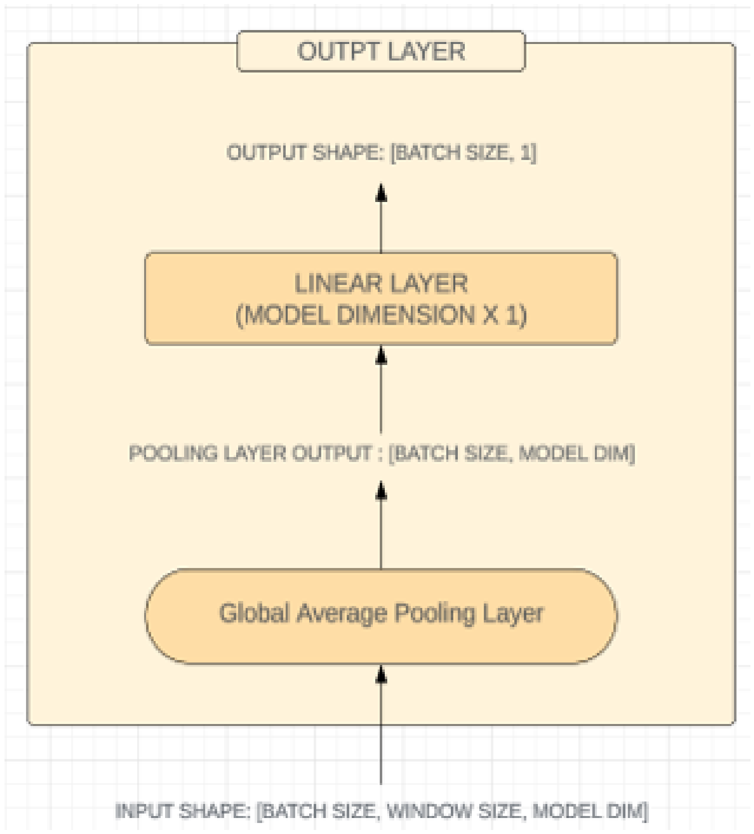

OUTPT LAYER
OUTPUT SHAPE: [BATCH SIZE, 1]
LINEAR LAYER
(MODEL DIMENSION X 1)
POOLING LAYER OUTPUT : [BATCH SIZE, MODEL DIM]
Global Average Pooling Layer
INPUT SHAPE: [BATCH SIZE, WINDOW SIZE, MODEL DIM]

Figure 3-10: Block Diagram for Output Block

3.3.2 Encoder Block

The encoder block consists of an attention layer and a feed-forward network. The data that is projected using the input projection block and positionally encoded will be fed into the attention layer first. The attention layer applies the scaled dot product mechanism and returns the attention scores for each input. This information is passed forward into the feed-forward network, where non-linear transformation and feature enrichment take place. With the introduction of residual connections and layer normalization, these techniques help address gradient vanishing/exploding problems and improve the rain stability of deep networks. In figure 3-9 (previous page), the input projection block passes the data of shape batch size x window size x model dimension into the attention layer and feed forward layer. Finally, the output data is of shape batch size x window size x model dimension.

3.3.3 Output Block

The output layer is the final block in the time series transformer consisting of an average pooling layer and a linear layer. The global average pooling layer calculates the average value of each feature map across all spatial dimensions (width and height) of the attention scores. This reduces the feature maps to a single vector for each channel, capturing the overall significance of that feature in the matrix. In figure 3-10 (Previous Page), the output layer receives data of shape batch size x window size x model dimension. This data is then pooled together and passed into a linear layer that predicts T+1 future price of the stock. The final output shape of this block is batch size x 1.

3.4 Time Series Transformer

The time series transformer (figure 3-11) took inspiration from the encoder only models and consists of only four blocks, input projection block, positional encoding block, encoder block, and output block. The input projection block acts like the word embedding layer and projects the information in a higher dimension which helps the model extract key relationships and information about the input data. This is followed by a positional encoding block that uses the sine-cosine value to encode the data with positional information. The encoder takes this data and passes the data through an attention layer and feed-forward network. The output of the encoder block consists of data that highlights key information and relationships about the input. Finally, the output block takes the data and aggregates the information using an average pooling layer and uses a linear layer to understand the information after aggregation to predict future T+1 closing prices for a particular stock.

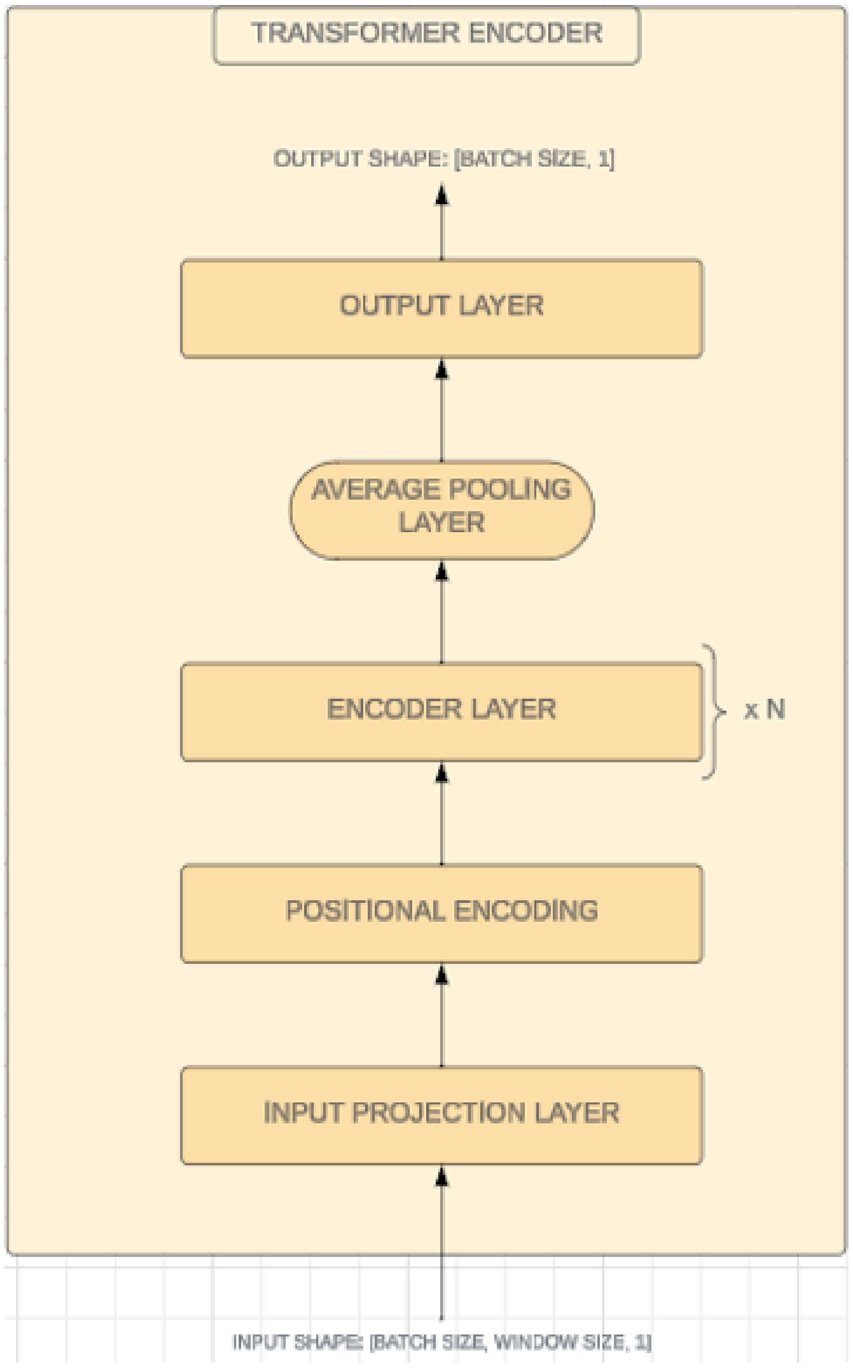


Figure 3-11: Block Diagram of Time Series Transformer

Chapter 4 Model Training and Testing

This chapter explores key steps in building the time series transformer model. First, the collection of data takes place followed by transforming the data to be ready for time-series transformer. This is followed by introducing the model configuration and using those exact parameters to train the transformer model for each stock in each sector. Lastly, training and testing of the time series transformer along with relevant graphs showing the output.

4.1 Data Acquisition

The Yahoo Finance API is a free available tool that can be leveraged to extract real-time stock market prices of most listed stocks across the globe. Yahoo Finance returns information on different periods, for example hourly, daily, weekly, monthly, and yearly. Depending on the task, Yahoo Finance returns information such as closing price, opening price, high price, low price, and volume. It is important to make sure that the information of the stock is reliable and consistent as missing values or incorrect values could hinder the learning or accuracy of the model.

4.2 Data Transformation

The data that is extracted using Yahoo Finance needs to be transformed so that it is compatible with the time series transformer. This is achieved using the sliding window technique. For our time series transformer, it is important to have input data and target data called X and Y respectively. The data extracted is in the form data shape x 1 and needs to be transformed into X shape batch size x window size x 1 and Y shape batch size x 1. This is achieved by using an iterative sliding window mechanism on the data. In the first iteration, data points from 0 to T+1 are sliced and split into X and Y datasets. The split takes place such that 0 to T data points are copied to X and (T+1) $^{\text{the}}$ data point is copied to Y. In the second iteration, the sliding window shifts n data points ahead (for this project n =1) and performs similar steps as above. Each time

```
Algorithm 1 Create Data Batches
function CreateDataBatches($data, window\_x, window\_y, window\_shift$)
    $X, Y \leftarrow [], []$
    for $i \leftarrow 0$ to $len(data) - window\_x - window\_y + 1$ step $window\_shift$ do
        $x\_window \leftarrow data[i : i + window\_x]$
        $y\_window \leftarrow data[i + window\_x : i + window\_x + window\_y]$
        $X$.append($x\_window$)
        $Y$.append($y\_window$)
    end for
    $X \leftarrow$ torch.stack($X$)
    $Y \leftarrow$ torch.stack($Y$).squeeze(2)
    return $X, Y$
end function
```

Figure 4-1: Pseudocode for Sliding Window Technique

the X and Y-sliced data is extracted, it is then stacked over each other to form a batch. This iterative process lets the final output of X and Y have a shape Batch size x Window Size x 1 and Batch size x 1 respectively. This means that for each batch size, there is a window size W with data points T that match a singular closing data point in Y for T+1. Figure 4-1 is the pseudocode for the iterative sliding window technique applied to the data in our thesis.

4.3 Model Configuration

The model configuration is an important aspect of training as it gives you leverage and control over how well the model performs in that task. In this thesis, all the model parameters are fixed except for model dimension and window size which were selected using cross-validation. These values change depending on the sector that the stock is found in (more information can be found in Appendix A). It is important to know that these parameters help reduce the loss of the model and increase the accuracy of predictions. The transformer model has many configurations possible due to its large variety of model parameters. They are: -

1. d_model: -This parameter defines the model dimension. This parameter changes depending on the sector the data is from.

2. nheads: - This parameter specifies the number of heads in the attention layer. The number of heads is fixed to 8.
3. num_layers: - This parameter specifies the number of encoders in the transformer. architecture. The number of encoder layers is fixed to 2.
4. Max_seq_length: - This parameter belongs in the positional encoder layer and specifies the maximum length of the input data that the model should be able to encode. The sequence length that can be positionally encoded is set to 500.
5. pl_input_dim: - This parameter belongs in the input projection layer that helps specify the last dimension in the input sequence. The dimension is fixed to 1 for training and testing.
6. pl_hidden_dim: - This parameter controls the hidden dimension in the input projection layer. The dimension is fixed to 128 for training and testing.
7. pl_output_dim: - This parameter controls the output dimension in the input projection layer which usually matches the model dimension of the transformer.
8. dim_feedforward: - This parameter controls the dimension of the feed-forward network that is found in the encoder layer. The dimension is fixed to 2048 for training and testing.
9. output_dim: - This parameter controls the final output dimension in the output layer. Usually, this is set to 1 (Closing Price). The dimension is fixed to 1 for training and testing.
10. dropout: - This parameter controls the added dropout layer in all the layers to make sure the model does not over-train. The dropout layer is set to 0.01.
11. learning_rate: - This parameter controls the tuning in the optimization algorithm used in the model by defining the size of the steps taken to reach the lowest loss. The learning rate is set to 0.0001.

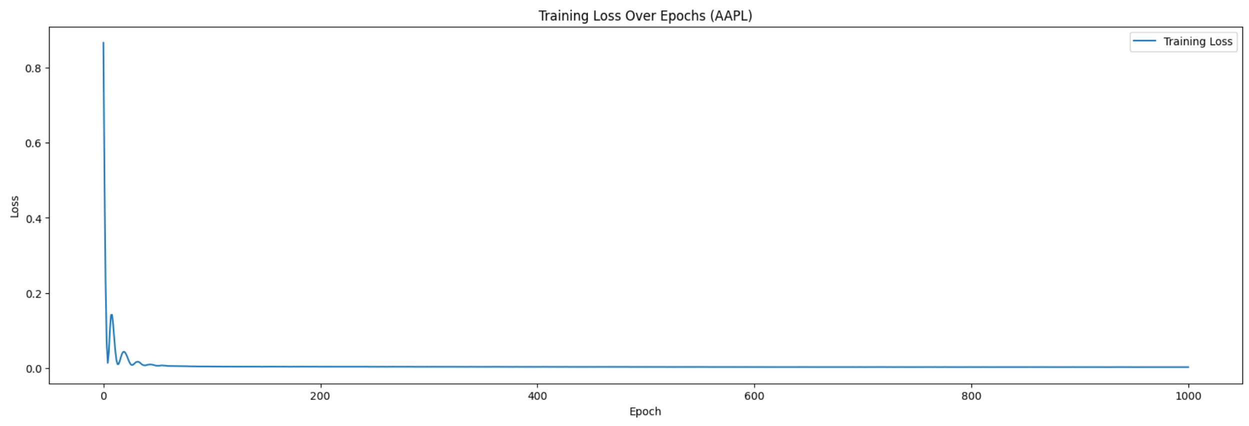


Figure 4-2: Apple Training Graph using Time Series Transformer

12. Adam Optimizer: - It is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. Adam is considered a combination of RMSprop and Stochastic Gradient Descent with momentum.

4.4 Model Training

Model training is an iterative process of feeding data to a machine learning algorithm to help it learn and improve its ability to perform a specific task, in this case predicting the future closing price values of individual stocks. Each of the stock market sectors has their specific window sizes and model dimensions which maximizes the model accuracy and reduces loss. I used Google Collab for model training using an T4 TPU which has 16GB of DDR4 RAM (Nvidia Tesla GPU), 201 GB of SSD Storage, and 12.7GB of System RAM.

The training data remains the same as the data chosen during the cross-validation tests. This ranges from 1st January 2017 to 2nd February 2023 and returns 1530 data points for the model to train with each individual stock. The training data is then batched and resized using sliding window technique.

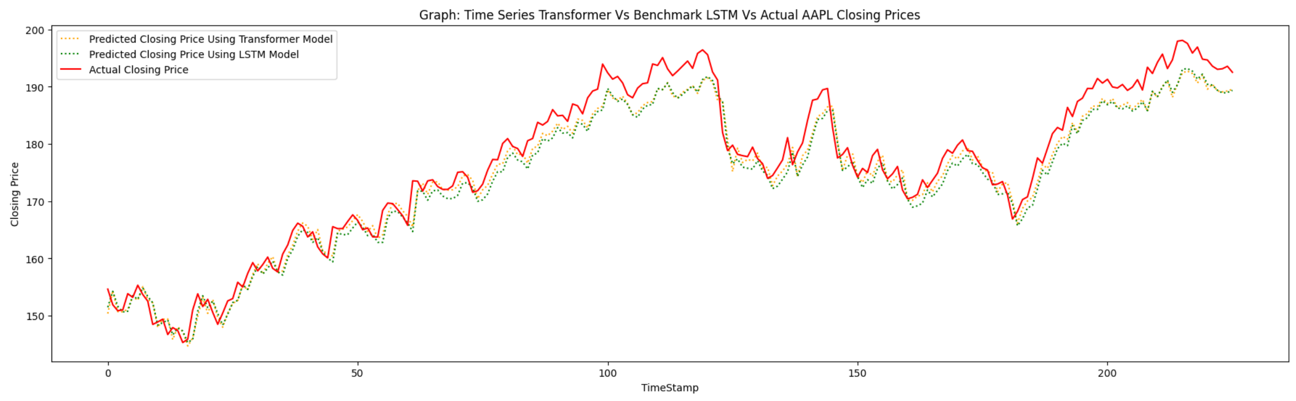


Figure 4-3: Time Series Transformer and LSTM model Predicting the Apple Stock Closing Price

Figure 4-2 (refer previous page) is an example of the time series transformer MSE loss graph for Apple Stock. The x-axis showcases the number of epochs the model ran for while the y-axis depicts the loss value at each epoch. It is seen that after epoch 50 the MSE values converge to zero (other stock training graph can be found Appendix B).

4.5 Model Testing

After training the model in a particular stock using the training dataset, it is crucial to see how well the model has learned the data, understood dependencies, and found relationships such that it can achieve similar results with unseen data. For testing, the data period starts from 1st February 2023 to 31st December 2023. This includes 230 data points for the model to predict. Each of the individual data points is also predicted using a baseline LSTM model which gives a visual understanding of how well the time series transformer predicts the data as compared to an LSTM model and vice versa. This process is repeated for all 20 different stocks that were extracted from 4 different sectors. Figure 4-3 plots the closing price values that are predicted by the time series transformer (dotted yellow), LSTM Model (dotted green) and actual closing price of Apple stock (red). The x-axis represents each timestamp, and the y-axis represents the closing price of the stock (other prediction comparison graphs can be found on Appendix C).

Chapter 5 Results and Discussions

In short, this thesis envisages a choice of an LSTM model as the baseline model. Five different stocks from four different sectors have been used. This gives the model a total of 20 different stocks to train and predict individually. The training data consists of closing price values of each stock from 1st January 2017 to 2nd February 2023 while the testing data had values from 2nd February 2023 to 31st December 2023.

5.1.1 Results Table

Tables 1-4 provide a comparative analysis of RMSE values achieved during the testing of time series transformers and LSTM models for various stocks, alongside recorded training times. Additionally, green highlighted cells denote stocks where significance has been established at a 95% confidence level (see Appendix C for details) and Bold indicating lower RMSE values.

Ticker	**Transformer (RMSE)**	**LSTM (RMSE)**	**Training Time Transformer (Seconds)**	**Training Time LSTM (Seconds)**
Technology Sector				
AAPL	**2.73179**	3.05637	71.63179	11.83203
GOOGL	2.26627	**2.22430**	74.94701	8.95759
IBM	1.38770	**1.35420**	72.5052	8.94904
AMZN	**2.47154**	2.51771	73.80354	10.23931
HPQ	**0.45323**	0.47398	75.09602	9.98877

Table 1: RMSE & Training Time Values for Time Series Transformer and LSTM Model for the Technology Sector

Ticker	**Transformer (RMSE)**	**LSTM (RMSE)**	**Training Time Transformer (Seconds)**	**Training Time LSTM (Seconds)**
Finance Sector				
JPM	**1.85619**	1.87050	13.36561	9.89305
BAC	**0.50232**	0.50837	12.88519	9.83190
COF	**1.95182**	1.97675	13.06328	9.69739
WF	**0.73531**	0.75214	13.06232	9.56277
AXP	**2.39368**	2.45006	13.06951	9.47508

Table 2: RMSE & Training Time Values for Time Series Transformer and LSTM Model for the Financial Sector.

Ticker	**Transformer (RMSE)**	**LSTM (RMSE)**	**Training Time Transformer (Seconds)**	**Training Time LSTM (Seconds)**
Pharmaceutical Sector				
JNJ	1.709663	**1.63408**	17.91004	8.89433
PFE	**0.52091**	0.52505	17.12886	11.09893
MRK	1.51717	**1.35821**	17.19714	10.80851
ABT	**1.32592**	1.37367	17.60171	8.92115
AMGN	3.42139	**3.3555**	17.65434	8.86070

Table 3: RMSE & Training Time Values for Time Series Transformer and LSTM Model for the Pharmaceutical Sector.

Ticker	**Transformer (RMSE)**	**LSTM (RMSE)**	**Training Time Transformer (Seconds)**	**Training Time LSTM (Seconds)**
FMCG Sector				
PG	1.38487	**1.36692**	17.28249	9.68173
PEP	**1.74527**	2.05813	17.35069	9.64310
NSRGF	**1.24706**	1.25348	17.62932	9.61233
UL	0.46956	**0.45560**	17.49177	9.62335
CL	0.71232	**0.71074**	19.49887	9.63754

Table 4: RMSE & Training Time Values for Time Series Transformer and LSTM Model for the FMCG Sector.

5.2 Results

One of the key objectives of the thesis is to design a transformer that can be used for stock market prediction and performance analysis. Creating a time series transformer model that took inspiration from the transformer-encoder only model and replacing certain blocks such word embedding layer with an input project layer and adding an output layer at the end of the encoder layer gave way to the time series transformer to predict time series data. The model learns and successfully predicts 20 different stocks closing price by fine-tuning hyperparameters such as window size and model dimension.

The time series transformer and baseline LSTM are compared against each other yielding interesting results. According to the RMSE comparison, time series transformer achieved lower values than the LSTM model for 12 out of 20 stocks. By employing a 95% confidence interval on both the models, the ranges overlap with one another. However, using paired z-test, 7 out of the 20 stocks were able to establish statistical significance in performance at the 95% confidence level.

With respect to training time, LSTM have better training speeds as compared to time series transformers. In the technology stock transformers took 3 times as much time to train as compared to LSTM models, similarly for other sectors, time series transformers took 1.2 times as much time as to train a LSTM Model.

From the above results, time series transformers can perform similarly to LSTM and given the right parameters even outperforming them in certain areas. Due to the training time taken and various model parameters, the trade-off between time series transformer and LSTM is important to note for further evaluation. This contradicts the paper published by Ailing and Muxi [19] which debates the validity and line of research for transformer in time series data (refer section 2.6).

5.3 Future Scope

A key objective is to showcase the ability of a transformer to perform time series tasks, specifically closing price prediction for stocks. To extend our research, we would first compare our model with some more industry standard models such as polynomial regression models, Multilayer perceptron, and RNN’s. This may provide additional information on the performance of the time series transformer. We also suggest incorporating the model to accept features such as volume of stock traded, daily stock volatility and even stock sentiment measures to help introduce more information within the model for more accurate predictions. Finally, as transformer models are known for their ability to generalize on large data, creating a large stock market model would help predict long range future stock closing price values for different stocks using a single model.

Appendices

Appendix A: Hyperparameter Tunning

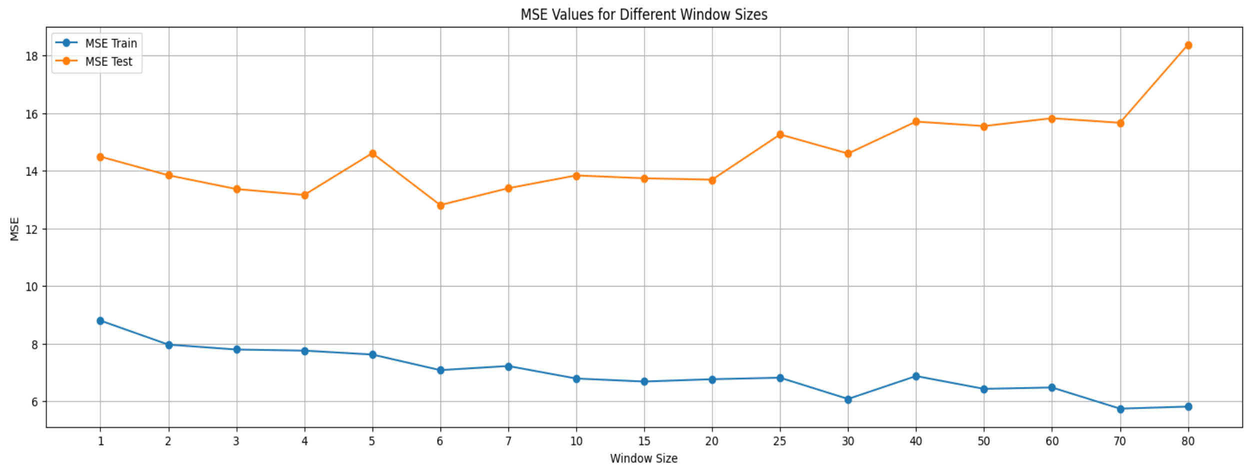
A.1: Window Size Selection

The window size is an important hyperparameter, it determines the number of past data points to be fed into the time series transformer for it to be able to predict future closing prices for a stock. Hence choosing the right window size is crucial as a narrow window size might miss out on important information while a larger window size might introduce irrelevant information that could potentially ruin predictions. It is also important to note that increasing the window size will also increase the model complexity and training times.

In this thesis, we used 17 different window sizes from 1 to 80 to help us understand how well the training and testing of the model. With the help of K-fold cross-validation, we split the original training dataset into 3 folds and for each iteration, 2 of the folds are concatenated together to form a new training dataset, and the remaining 1-fold is used for testing. The RMSE and MSE values achieved from each iteration are averaged out and then plotted on a graph for representation. It can be seen from the below image that as the window size increases, the model keeps achieving lower RMSE and MSE values but for the test RMSE and MSE, window size 6 achieved the lowest RMSE test value.

Window sizes selection: [1, 2, 3, 4, 5, 6, 7, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80].

Below figure A-1 showcases the RMSE and MSE values obtained from cross-validation of the training data and testing data. The x-axis in the graph represents the window size of each model and the y-axis represents the MSE/RMSE Loss for 3-fold cross-validation.



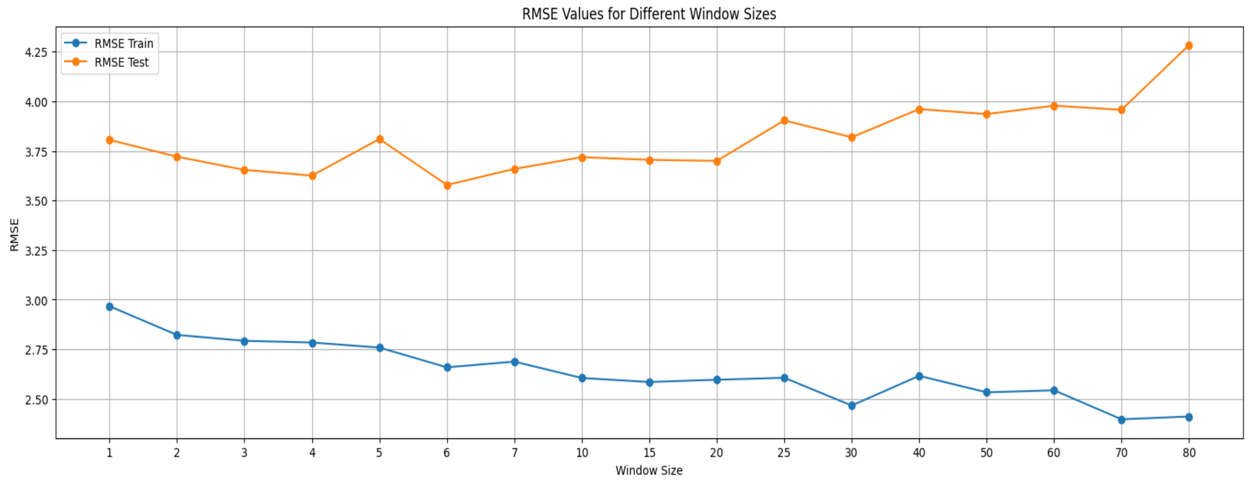


Figure A-1: RMSE and MSE values for training and testing dataset using k-fold

A.2: Model Dimension Selection

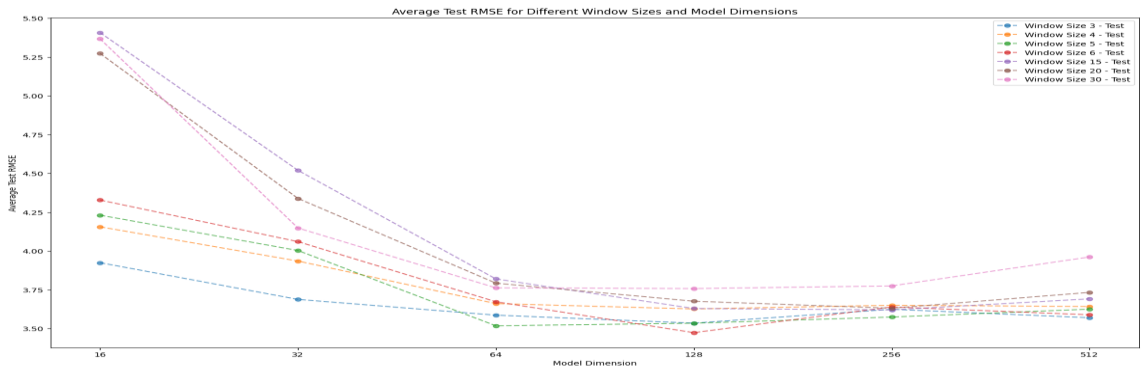
The model dimensionality (d_model) captures how information flows across the model architecture. Higher model dimensions help capture intricate details about the input which could lead to improved model performance while, a lower model dimension helps in focusing on important features that could lead to faster training and inference times. From the above 17 window sizes, selecting 8 window sizes (3, 4, 5, 6, 10, 15, 20, 30) and coupling them with 6 different model dimensions (16, 32, 64, 128, 256, 512) helps to plot different RMSE values for different configurations and select the lowest amongst those values.

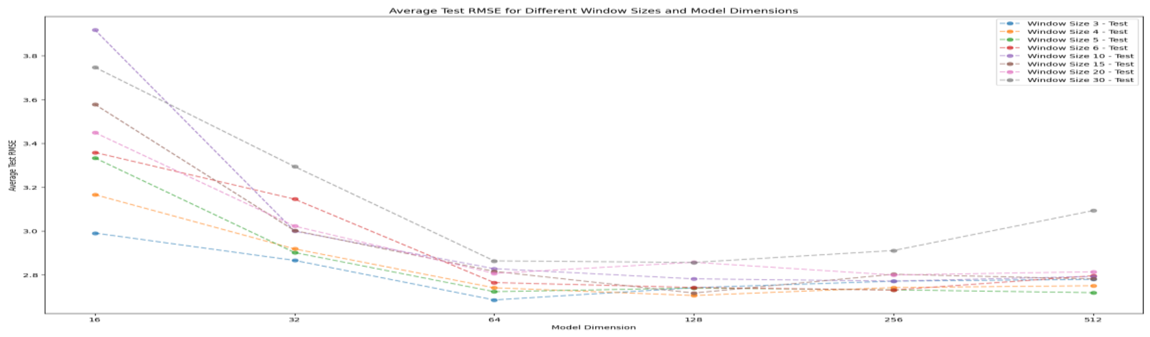
Figure A-2 is a graphical representation of the RMSE values achieved during three-fold cross validation of various model dimensions. This has been achieved using time series transformer model for Apple, Google, IBM, and Amazon (from top to bottom). The x-axis of the graph represents model dimension, and the y-axis represents the average RMSE values achieved by each model dimension for the test fold dataset. Each of the curves for model dimensions are given a unique color for identification purposes. In the first figure, Apple stock achieves the lowest RMSE value with a window size of 6 and model dimension size of 128. In the second figure, Google stock achieves the lowest RMSE value with window size 3 and model dimension 64. In the third figure, IBM stock achieves the lowest RMSE value with window size 3 and model dimension 128. In the fourth figure, Amazon stock achieves the lowest RMSE values with window size 3 and model dimension 64. This leads to the conclusion that the ideal window size for the technology sector should be 3 and model dimension 64. However, using a window size of 4 and model dimension of 512, the model achieves lower RMSE values with the testing data with regards to overall technology stocks.

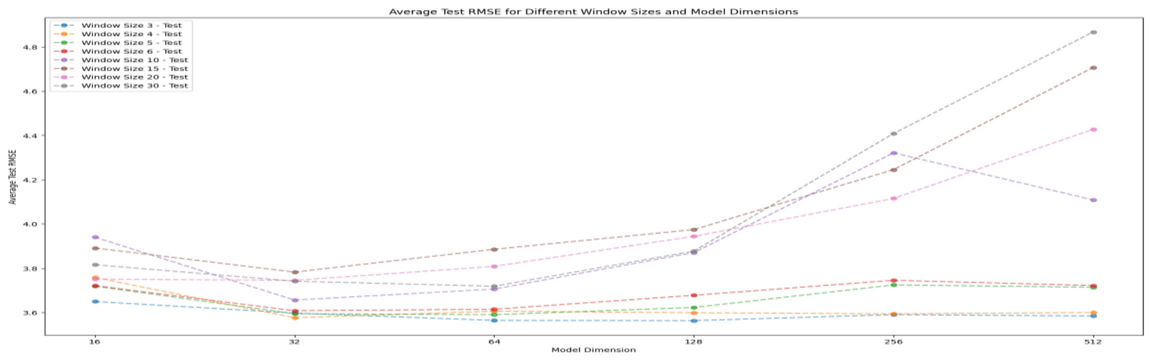
Figure A-3 is a graphical representation of the RMSE values achieved during three-fold cross validation of various model dimensions. This has been achieved using time series transformer model for JP Morgan, Bank of America, Capital One Finance, Well Fargo (from top to bottom). The x-axis of the graph represents model dimension, and the y-axis represents the average RMSE values achieved by each model dimension for the test fold dataset. Each of the curves for model dimensions are given a unique color for identification purposes. In the first figure, JP Morgan stock achieves the lowest RMSE value with a window size of 3 and model dimension size of 64. In the second figure, Bank of America stock achieves the lowest RMSE value with window size 3 and model dimension 64. In the third figure, capital one finance stock achieves the lowest RMSE value with window size 6 and model dimension 128. In the fourth figure, Well Fargo stock achieves the lowest RMSE values with window size 15 and model dimension 128. This leads to the conclusion that the ideal window size for the finance sector should be 3 and model dimension 64.

Figure A-4 is a graphical representation of the RMSE values achieved during three-fold cross validation of various model dimensions. This has been achieved using time series transformer model for Johnson & Johnson, Pfizer, Merck, and Abbott Labs (from top to bottom). The x-axis of the graph represents model dimension, and the y-axis represents the average RMSE values achieved by each model dimension for the test fold dataset. Each of the curves for model dimensions are given a unique color for identification purposes. In the first figure, Johnson & Johnson stock achieves the lowest RMSE value with a window size of 5 and model dimension size of 32. In the second figure, Pfizer stock achieves the lowest RMSE value with window size 3 and model dimension 64. In the third figure, Merck stock achieves the lowest RMSE value with window size 3 and model dimension 512. In the fourth figure, Abbott labs stock achieves the lowest RMSE values with window size 3 and model dimension 512. It can be stated empirically that a common ground of window size 3 and model dimension 128 for the pharmaceutical sector is evident. This is primarily due to similarity in achieving low RMSE values across all four stocks.

Figure A-5 is a graphical representation of the RMSE values achieved during three-fold cross validation of various model dimensions. This has been achieved using time series transformer model for P&G, Pepsi, Nestle, and Unilever (from top to bottom). The x-axis of the graph represents model dimension, and the y-axis represents the average RMSE values achieved by each model dimension for the test fold dataset. Each of the curves for model dimensions are given a unique color for identification purposes. In the first figure, P&G stock achieves the lowest RMSE value with a window size of 4 and model dimension size of 128. In the second figure, Pepsi stock achieves the lowest RMSE value with window size 3 and model dimension 64. In the third figure, Nestle stock achieves the lowest RMSE value with window size 6 and model dimension 64. In the fourth figure, Unilever stock achieves the lowest RMSE values with window size 4 and model dimension 64. It can be stated empirically that a common ground of window size 3 and model dimension 128 for the FMCG sector is evident. This is primarily due to similarity in achieving low RMSE values across all four stocks.







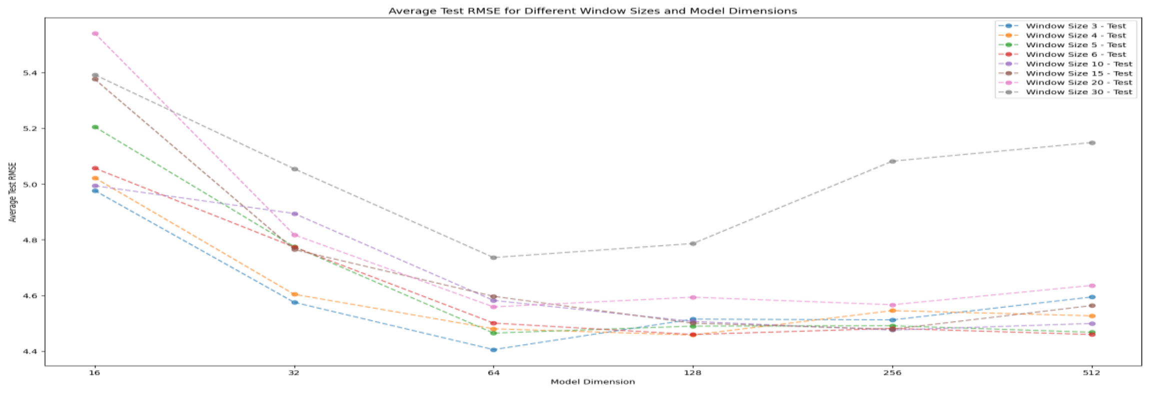
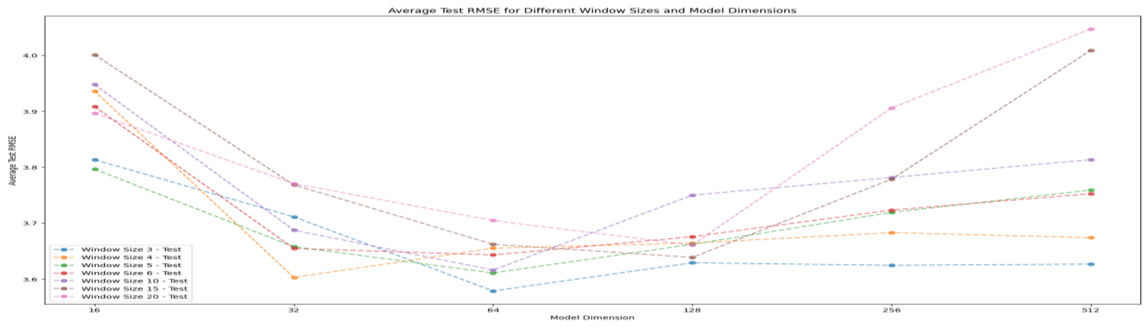
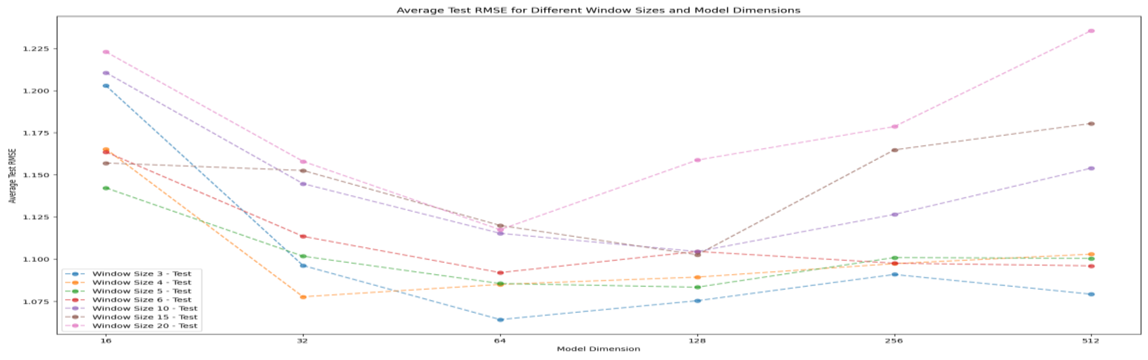
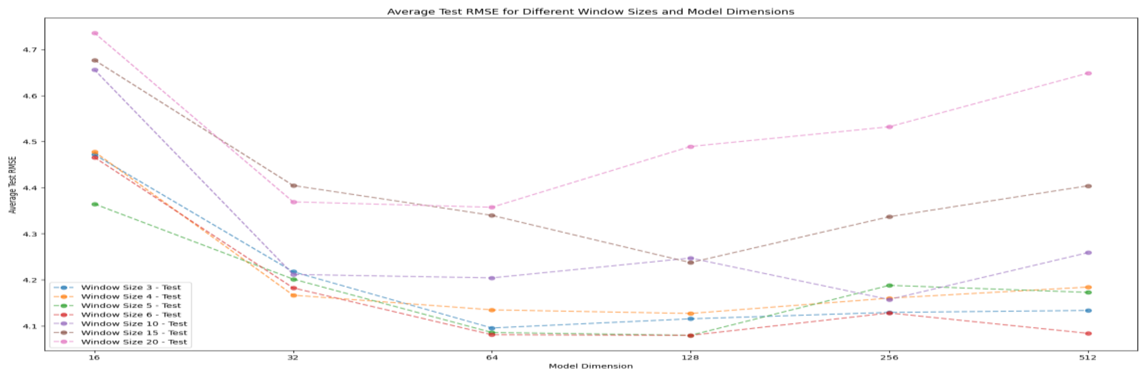


Figure A-2: Model Dimension RMSE Graphs on the Test Dataset using K-fold for Technology Sector Stocks: Apple, Google, IBM, and Amazon (Top to Bottom).







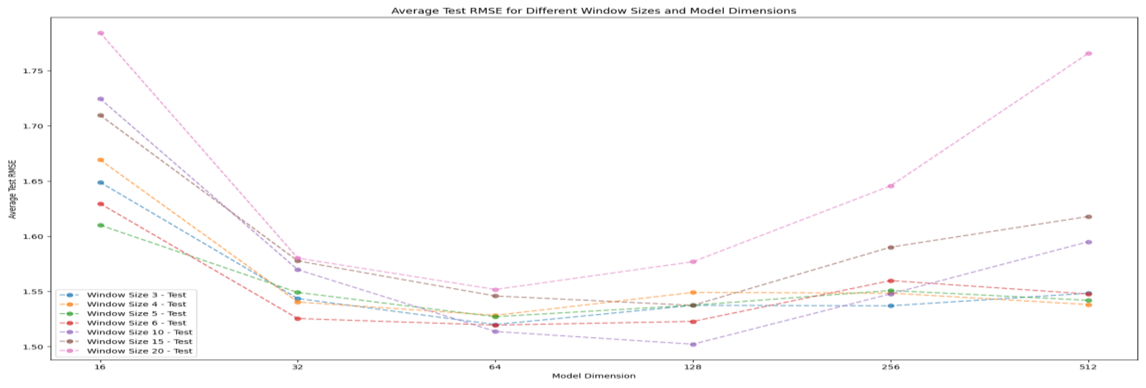


Figure A-3: Model Dimension RMSE Graphs on the Test Dataset using K-fold for Financial Sector Stocks: JP Morgan, Bank of America, Capital One Finance and Well Fargo (Top to Bottom).

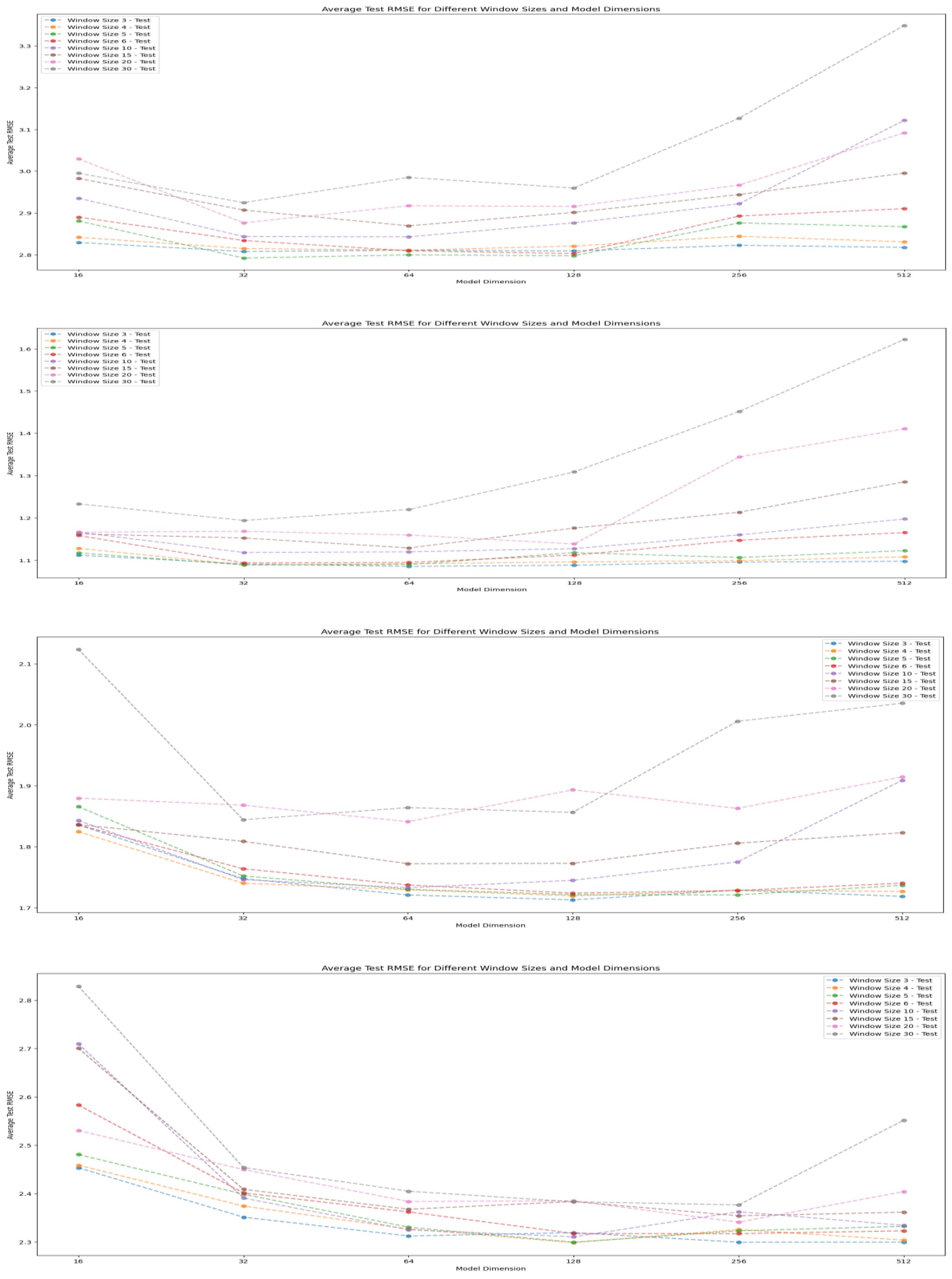


Figure A-4: Model Dimension RMSE Graphs on the Test Dataset using K-fold for Pharmaceutical Sector Stocks: Johnson and Johnson, Pfizer, Merck and, Abbott Labs (Top to Bottom).

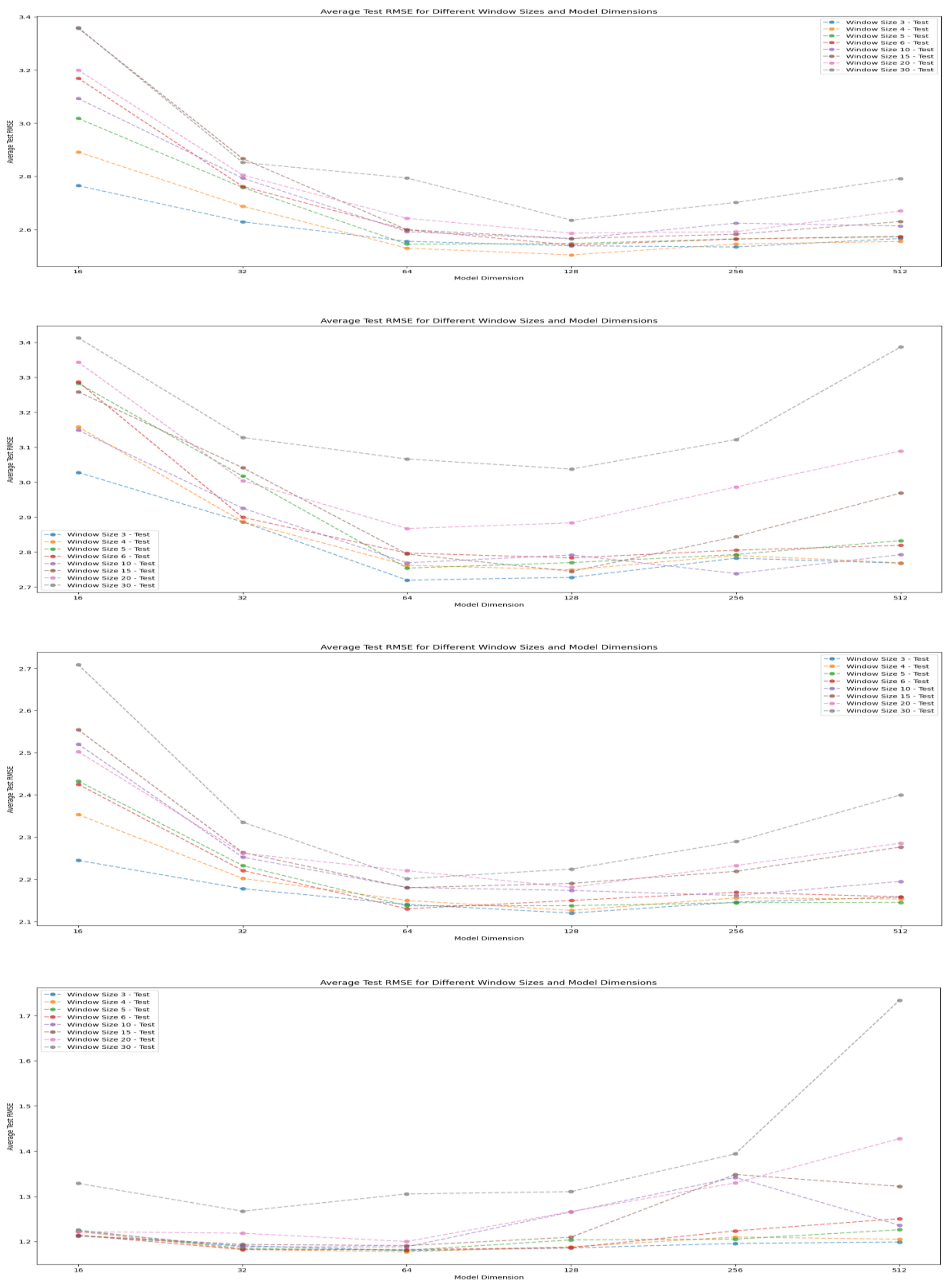


Figure A-5: Model Dimension RMSE Graphs on the Test Dataset using K-fold for FMCG Sector Stock: P&G, Pepsi, Nestle and Unilever (Top to Bottom).

.

Appendix B: Model Training Graphs

Time series transformer model training is an iterative process of feeding data to a machine learning algorithm to help it learn and improve its ability to perform a specific task, in this case, predict the future closing price values of individual stocks. After hyperparameter tuning, each of the four sectors has a predefined window size and model dimension size which is used for training and testing.

- Technology sector: window size 4 and model dimension 512.
- Finance sector: window size 3 and model dimension 64.
- Pharmaceutical sector: window size 3 and model dimension 128.
- FMCG sector: window size 3 and model dimension 128.

Figure B-6, B-7, B-8, B-9 is a graphical plot depicting the training losses recorded by the time series transformer model for all the stocks in the four sectors. All the graphs converge close to zero after the 50th epoch. This indicates the model being able to learn, capture and reproduce better predictions after each epoch. Each x-axis in these graphs represents the number of epochs done to train the model while each of the y-axis represents the MSE loss achieved for each epoch during training.

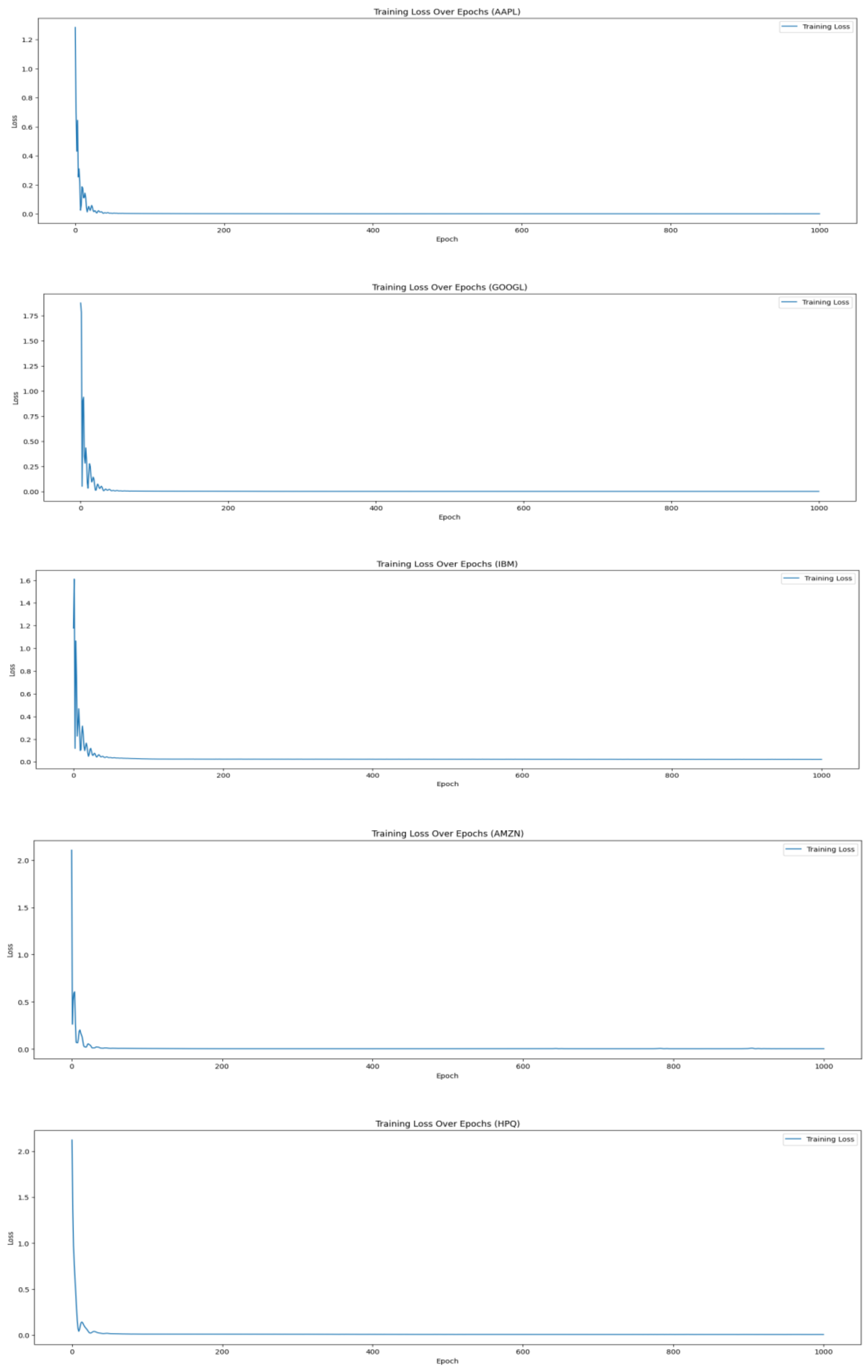


Figure B-6: Training graph for Technology Sector Stock: Apple, Google, IBM, Amazon, HP (Top to Bottom)

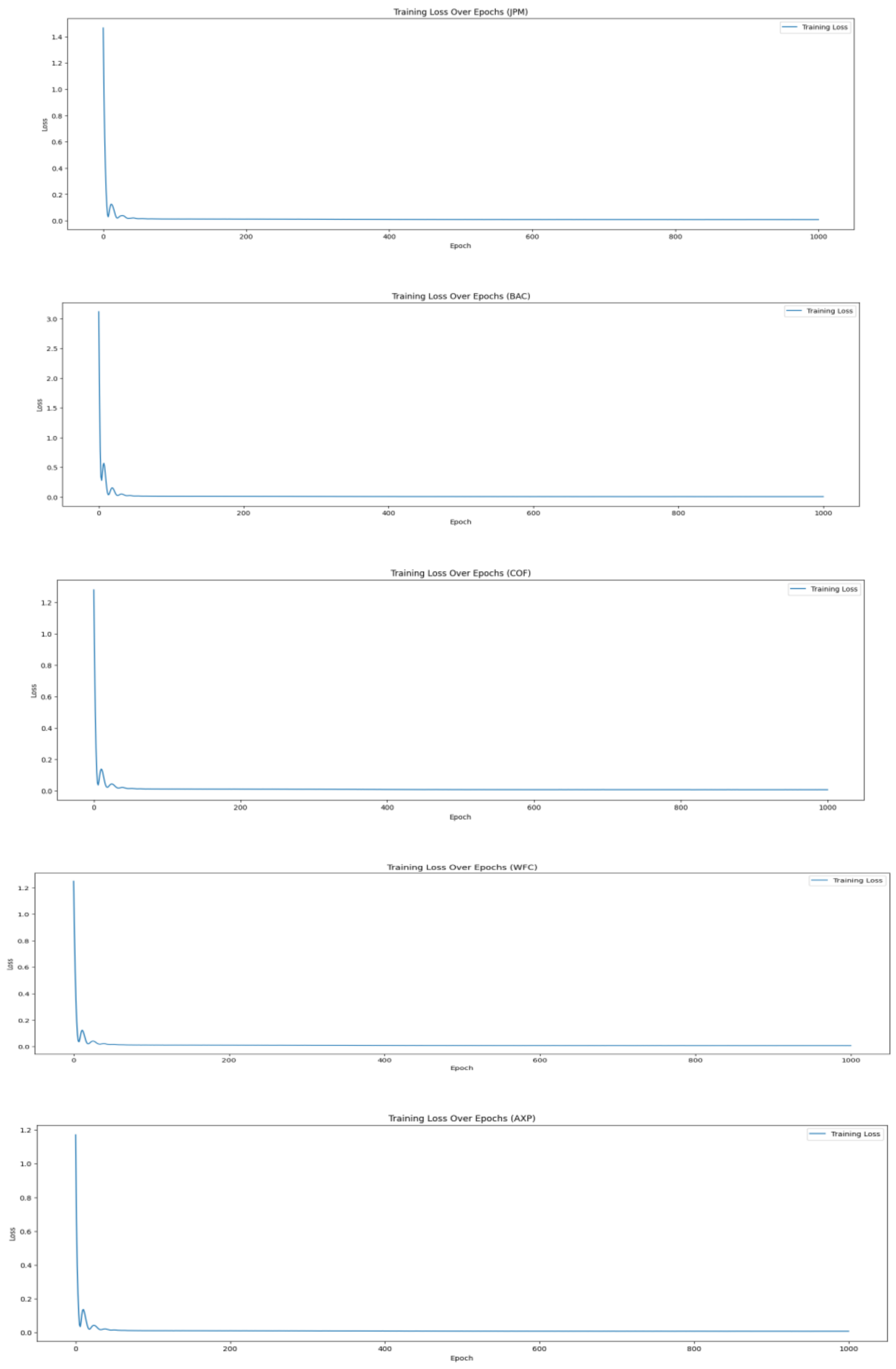
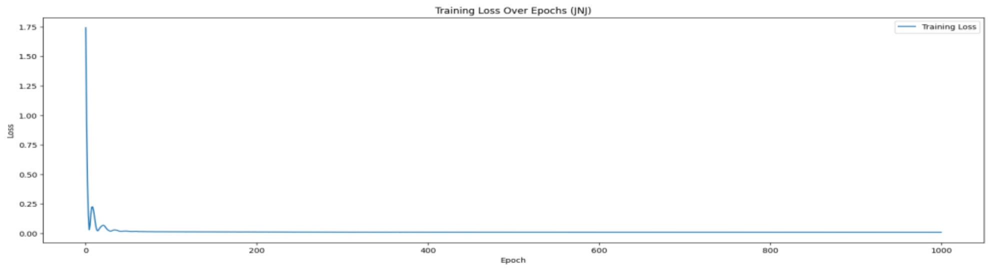
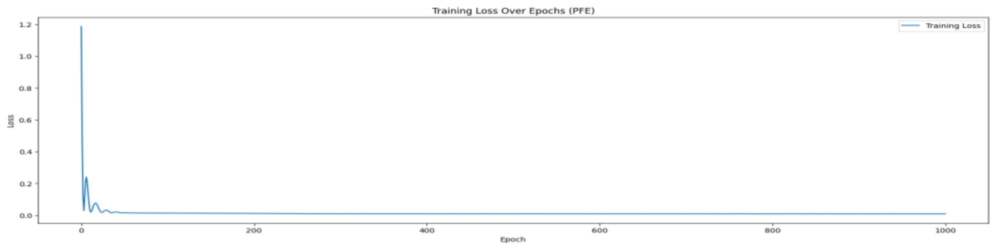
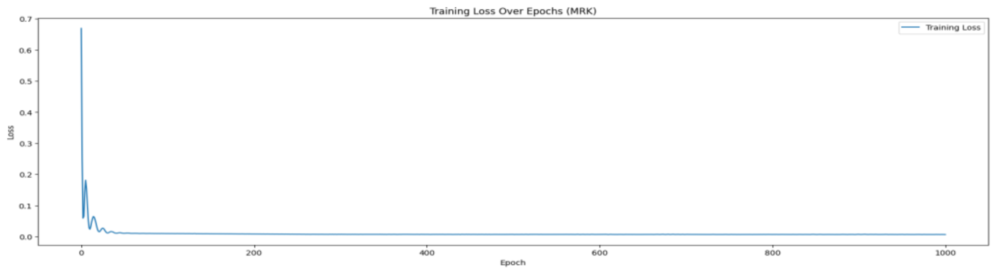
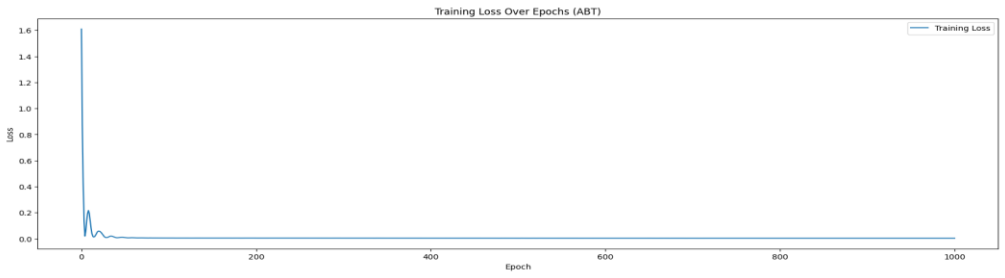


Figure B-7: Training Graphs for Financial Sector Stocks: JP Morgan, Bank of America, Capital One Finance, Well Fargo and American Express (Top to Bottom).









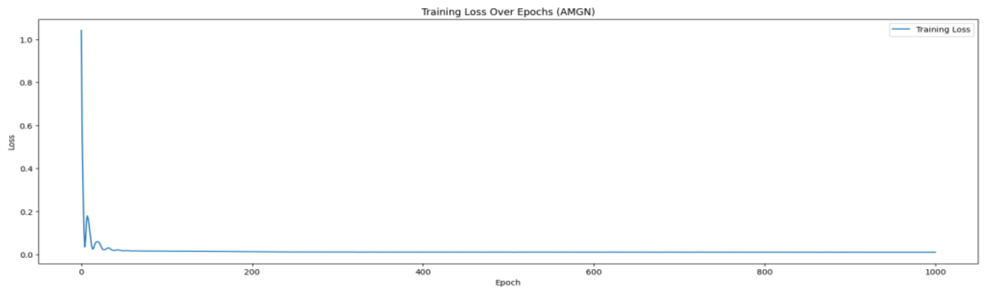


Figure B-8: Training Graphs for Pharmaceutical Sector Stocks: Johnson and Johnson, Pfizer, Merck, Abbott Labs and Amgen (Top to Bottom).

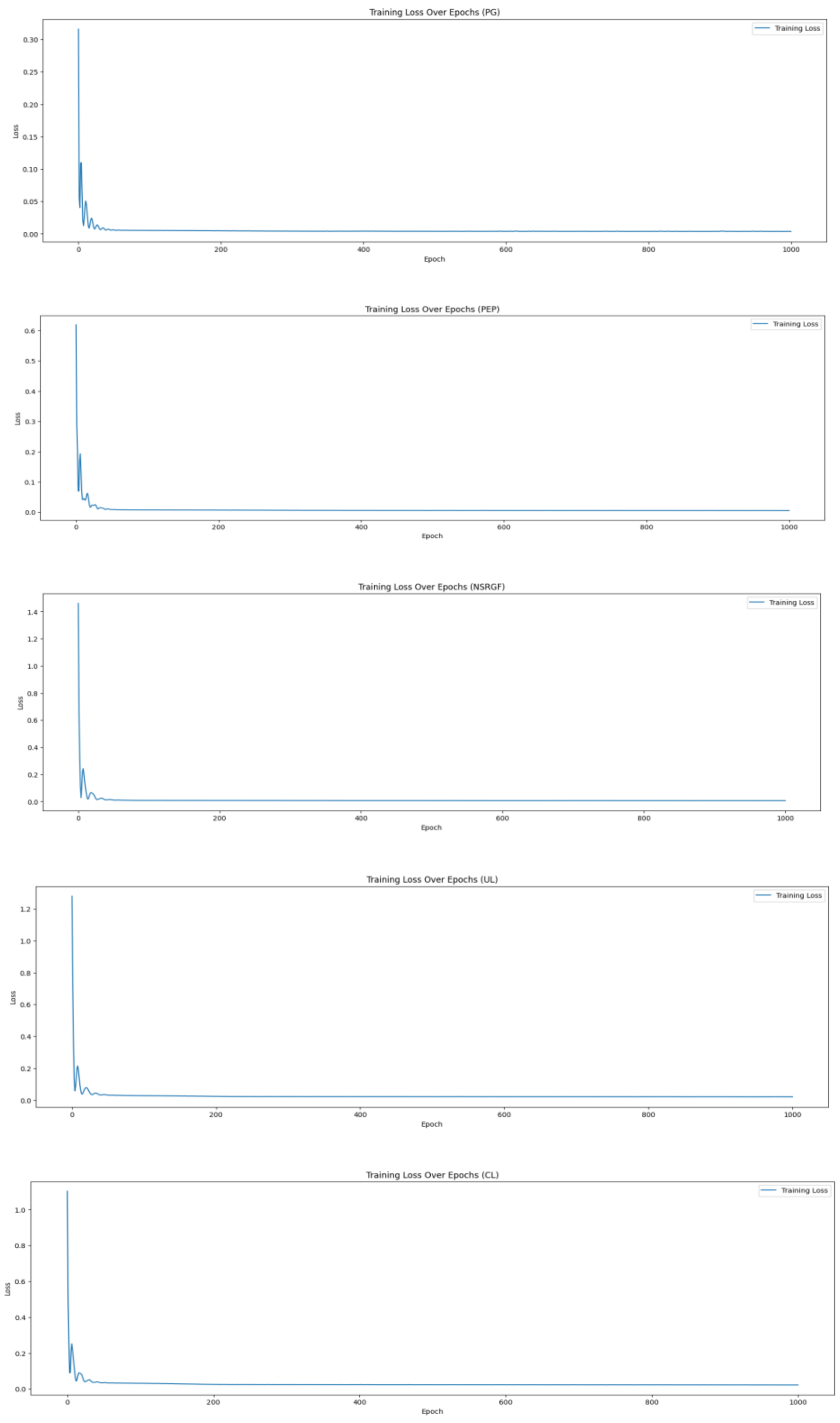


Figure B-9: Training Graphs for FMCG Sector Stock: P&G, Pepsi, Nestle, Unilever, Colgate (Top to Bottom).

Appendix C: Time Series Transformer Vs LSTM Model Prediction Graphs

The LSTM model leveraged similar configurations to remove any discrepancy between the results. PyTorch and python were instrumental in creating the baseline LSTM. With just 3 simple layers added to the LSTM model class, LSTM Layer, Dropout Layer and Linear Layer.

- LSTM Layer: The input data (x) is passed through the LSTM module, which processes the sequence and captures temporal dependencies.
- Dropout: The Dropout layer is applied to the LSTM output (out) with the specified dropout rate. This technique helps mitigate overfitting by randomly setting a portion of the activations to zero during training.
- Fully Connected Layer: The final step involves feeding the processed output from the previous layer (out) into a fully connected layer with a single neuron. This layer maps the hidden representation from the LSTM to the desired output size.

The LSTM model is configured with the following specifications:

- input_size = 1
- hidden_size = 512
- num_layers = 2
- output_size = 1
- dropout_rate =0.5
- learning rate = 0.001

A 95% Confidence interval is calculated on both time series transformer and LSTM Model to extract a lower and upper limit for RMSE Values. From table 5 -8, using 95% confidence interval we could not establish any statistically significant evidence between the two models as they have overlapping confidence intervals.

Ticker Name	**Transformer Model**	**LSTM Benchmark Model**
Technology Stock		
AAPL	[2.43192, 2.88332]	[2.76592, 3.23546]
GOOGL	[1.81623, 2.66567]	[1.82584, 2.59021]
IBM	[1.34461, 1.71947]	[1.148851, 1.50997]
AMZN	[2.0986, 2.75172]	[2.12215, 2.79123]
HPQ	[0.39939, 0.53895]	[0.39678, 0.53930]

Table 5: 95% Confidence Interval for Technology Sector

Ticker Name	**Transformer Model**	**LSTM Benchmark Model**
Technology Stock		
JPM	[1.56427, 2.14689]	[1.54177, 2.14619]
BAC	[0.43584, 0.56191]	[0.43618, 0.57208]
COF	[1.73827, 2.15780]	[1.73538, 2.17447]
WF	[0.35988, 0.44454]	[0.37501, 0.45506]
AXP	[2.16447, 2.71827]	[2.16781, 2.70700]

Table 6: 95% Confidence Interval for Finance Sector

Ticker Name	Transformer Model	LSTM Benchmark Model
Technology Stock		
JNJ	[1.37621, 1.98114]	[1.29465, 1.92131]
PFE	[0.46405, 0.61602]	[0.45775, 0.60005]
MRK	[1.24147, 1.48677]	[1.17266, 1.44375]
ABT	[1.54052, 1.05254]	[1.06666, 1.54477]
AMGN	[3.00707, 3.75275]	[2.94412, 3.76176]

Table 7: 95% Confidence Interval for Pharmaceutical Sector

Ticker Name	Transformer Model	LSTM Benchmark Model
Technology Stock		
PG	[1.19235, 1.53796]	[1.17193, 1.51487]
PEP	[1.47598, 1.95971]	[1.58020, 2.06253]
NSRGF	[1.10465, 1.37216]	[1.10871, 1.36207]
UL	[0.43497, 0.57334]	[0.37617, 0.528932]
CL	[0.62417, 0.79225]	[0.62389, 0.78917]

Table 8: 95% Confidence Interval for FMCG Sector.

A paired Z-test has been applied on both models to check for any statistical significance between time series transfosixrmer and LSTM model. Table 9-12 showcases the paired z-test values for all 20 stocks at a 95% confidence level. 7 out of the 20 stocks were able to reject null hypothesis and provide statistical significance.

Paired Z-Test Scores	
Ticker	Score
AAPL	-2.77120
GOOGL	-2.64613
IBM	-3.74566
AMZN	-3.18168
HPQ	-3.88885

Table 9: Paired Z-Test Values for Technology Sector

Paired Z-Test Scores	
Ticker	Score
JPM	-0.12057
BAC	-0.019616
COF	-0.29197
WF	0.05240
AXP	2.45037

Table 10: Paired Z-Test Values for Financial Sector

Paired Z-Test Scores	
Ticker	Score
JNJ	0.34524
PFE	0.269179
MRK	2.6275
ABT	-0.16304
AMGN	0.64755

Table 11: Paired Z-Test Values for Pharmaceutical Sector

Paired Z-Test Scores	
Ticker	Score
PG	0.15217
PEP	-0.16167
NSGRY	0.046835
UL	0.675734
CL	0.07914

Table 12: Paired Z-Test Values for FMCG Sector

Figure C-10 showcases 5 different graphs in the technology sector which compare the time series transformer model predictions (dotted yellow), LSTM model predictions (dotted green) and actual stock closing price(red). The x-axis represents each timestamp in the testing data and y axis represents the actual closing price of the stock.

Figure C-11 showcases 5 different graphs in the finance sector which compare the time series transformer model predictions (dotted yellow), LSTM model predictions (dotted green) and actual stock closing price(red). The x-axis represents each timestamp in the testing data and y axis represents the actual closing price of the stock.

Figure C-12 showcases 5 different graphs in the pharmaceutical sector which compares the time series transformer model predictions (dotted yellow), LSTM model predictions (dotted green) and actual stock closing price(red). The x-axis represents each timestamp in the testing data and y axis represents the actual closing price of the stock.

Figure C-13 showcases 5 different graphs in the FMCG sector which compare the time series transformer model predictions (dotted yellow), LSTM model predictions (dotted green) and actual stock closing price(red). The x-axis represents each timestamp in the testing data and y axis represents the actual closing price of the stock.

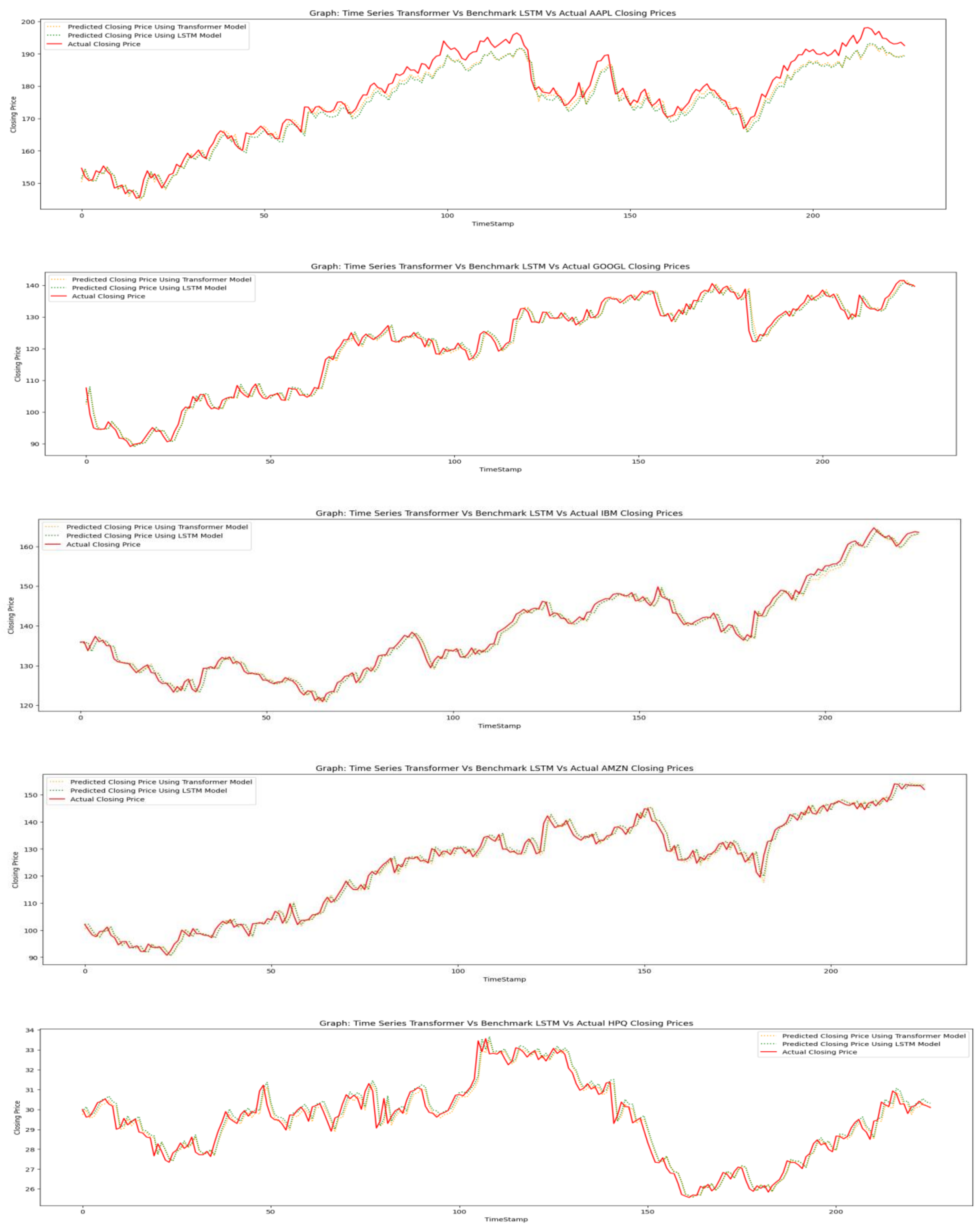


Figure C-10: Actual Closing Price VS Time Series Predictions VS LSTM Predictions for Technology Sector Stocks: Apple, Google, IBM, Amazon, and HP (Top to Bottom).

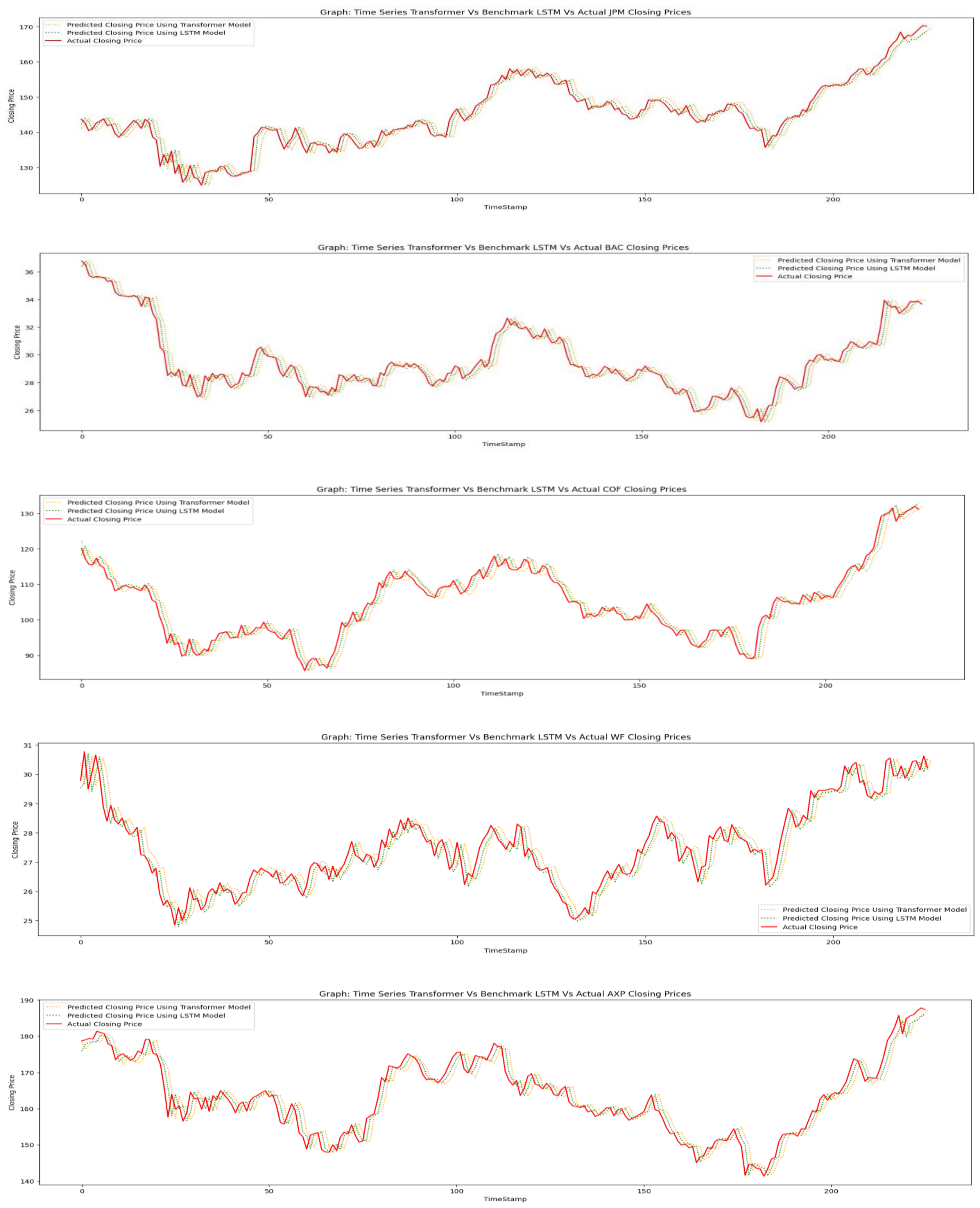


Figure C-11: Actual Closing Price VS Time Series Predictions VS LSTM Predictions for Financial Sector Stocks: JP Morgan, Bank of America, Capital One Finance, Well Fargo, and American Express (Top to Bottom).

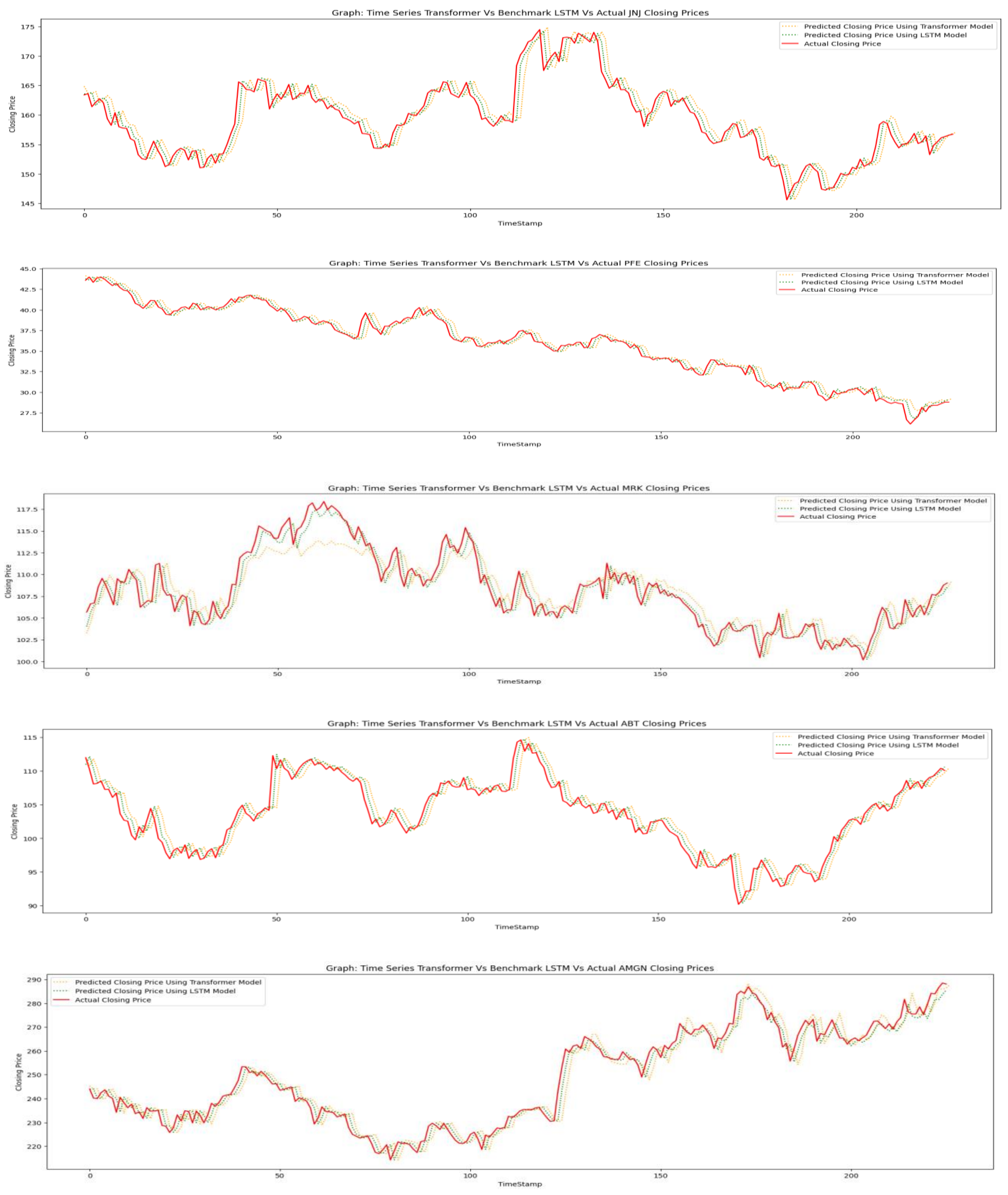


Figure C-12: Actual Closing Price VS Time Series Predictions VS LSTM Predictions for Pharmaceutical Sector Stocks: Johnson and Johnson, Pfizer, Merck, Abbott Labs, and Amgen Labs (Top to Bottom).

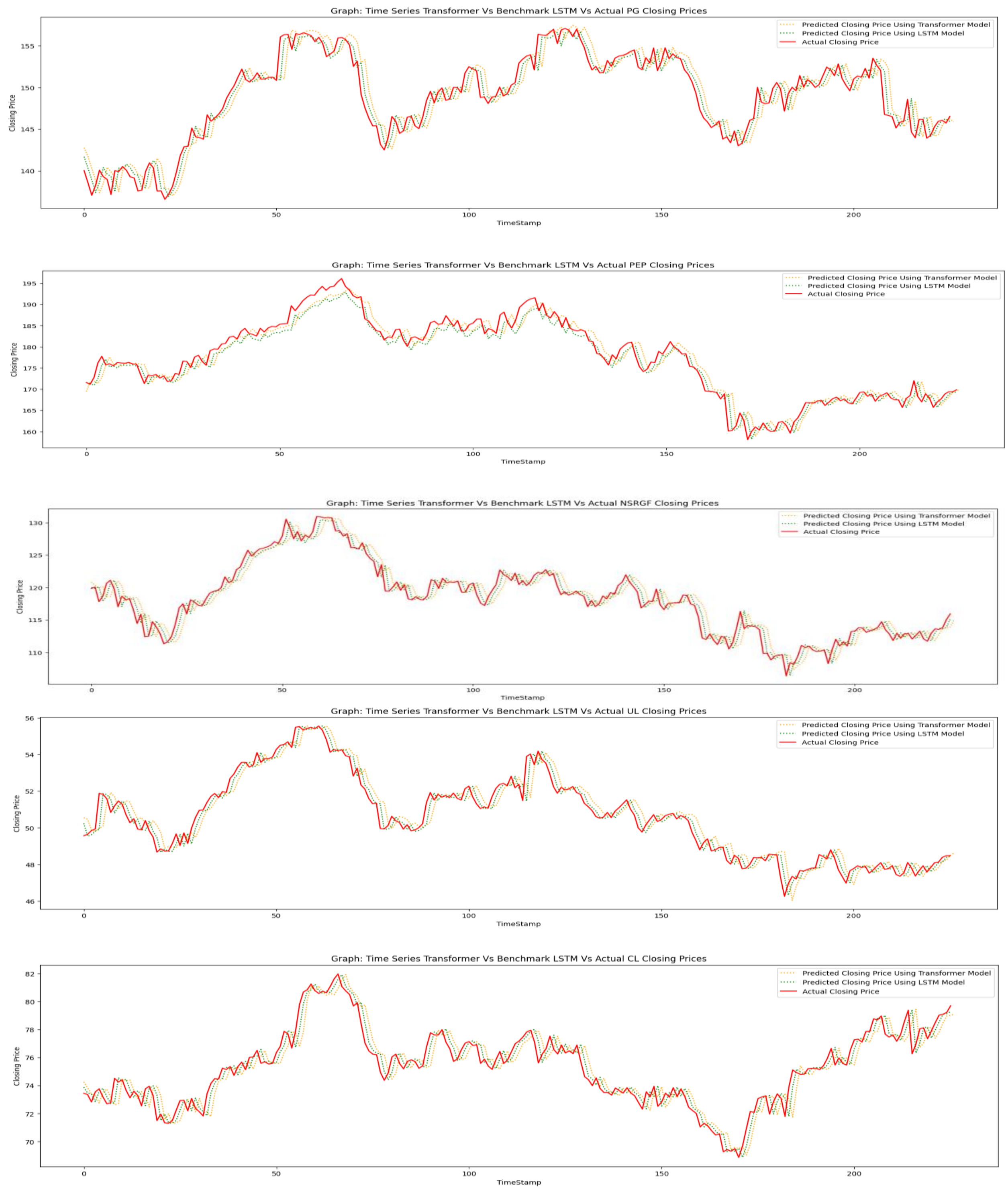


Figure C-13:Actual Closing Price VS Time Series Predictions VS LSTM Predictions for FMCG Sector Stocks: P&G, Pepsi, Nestle, Unilever and Colgate (Top to Bottom).

References

[1] IBM, "Characteristics of time series," 17 08 2021. [Online]. Available: https://www.ibm.com/docs/hu/spss-modeler/saas?topic=data-characteristics-time-series [accessed: 2023-12-15].

[2] A. HAYES, "How Does the Stock Market Work?," Investopedia, 23 12 2023. [Online]. Available: https://www.investopedia.com/articles/investing/082614/how-stock-market-works.asp. [Accessed 2 02 2024].

[3] Wikipedia, "Stock Market Prediction," Wikipedia, 24 12 2006. [Online]. Available: https://en.wikipedia.org/wiki/Stock_market_prediction#:~:text=the%20present%20value.-,Prediction%20methods,(charting)%20and%20machine%20learning.. [Accessed 18 12 2023].

[4] M. Xie b and S. L. Ho, "The use of ARIMA models for reliability forecasting and analysis," *Computers & Industrial Engineering,* vol. 35, no. 1-2, pp. 213-216, 1998.

[5] H. He and S. Chen, "Stock Prediction Using Convolutional Neural Network," *International Conference on Artificial Intelligence Applications and Technologies,* 2018.

[6] Y. Zhu, "Stock price prediction using the RNN model," *International Conference on Applied Physics and Computing,* 2020.

[7] A. Mittal, "Understanding RNN and LSTM," Medium, 19 10 2018. [Online]. Available: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed 24 01 2024].

[8] O. Calzone, "An Intuitive Explanation of LSTM," Medium, 21 02 2022. [Online]. Available: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. [Accessed 24 01 2024].

[9] P. Samarawickrama and I. Fernando, "A Recurrent Neural Network Approach in Predicting," *International Conference on Industrial and Information Systems*, 2017.

[10] P. C. Purnama, N. L. Wiwik, W. A. Surya Darma and P. A. Eka , "Comparison of Support Vector Machine (SVM) and Linear Regression (LR) for Stock Price Prediction," *International Conference on Informatics and Computing (ICIC)*, 2023.

[11] P. Saraf and S. Ravikumar, "Prediction of Stock Prices using Machine Learning (Regression, Classification) Algorithms," *International Conference for Emerging Technology (INCET)*, 2020.

[12] M. Sabokrou, H. Soltanalizadeh, A. Solouki and S. M. Mirjebreili, "Multi-Task Transformer for Stock Market Trend Prediction," *International eConference on Computer and Knowledge Engineering*, vol. 12, no. 12, 2022.

[13] L. D. Costa and A. Machado, "Prediction of Stock Price Time Series using Transformers," *ANAIS DO BRAZILIAN WORKSHOP ON ARTIFICIAL INTELLIGENCE IN FINANCE*, 2023.

[14] Q. Zhang, X. Liu, S. Lv and Y. Li, "Incorporating Transformers and Attention Networks for Stock Movement Prediction," *Complexity*, p. 10, 2022.

[15] N. Malibari, I. Katib and R. Mehmood, "Predicting Stock Closing Prices in Emerging Markets with Transformer Neural Networks," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 12, 2021.

[16] M. S. Alam, S. . I. Khan, M. Ibrahim, M. M. Muhu, M. Ahsan, A. B. Aftab and T. Muhammad, "Transformer-Based Deep Learning Model for Stock Price Prediction: A Case Study on Bangladesh Stock Market," *International Journal of Computational Intelligence and Applications,* vol. 22, no. 1, 2022.

[17] N. Bagherzadeh, Y. Qiao and H. Kaeley, "Support for Stock Trend Prediction Using Transformers and Sentiment Analysis," in *The International Institute of Social and Economic Sciences*, London, 2023.

[18] Q. Zhang, S. Zhang, Y. Chen and C. Wang, "Stock market index prediction using deep Transformer model," *Expert Systems with Applications,* vol. 208, 2022.

[19] Q. Xu, L. Zhang, M. Chen and A. Zeng, "Are Transformers Effective for Time Series Forecasting?," *The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23),* 2023.

[20] I. Polosukhin, Ł. Kaiser, A. Gomez, L. Jones, J. Uszkoreit, N. Parmar, N. Shazeer and A. Vaswani, "Attention Is All You Need," *31st Conference on Neural Information Processing Systems,* 2017.

[21] G. Lokare, "Preparing Text Data for Transformers: Tokenization, Mapping and Padding," Medium, 10 02 2023. [Online]. Available: https://medium.com/@lokaregns/preparing-text-data-for-transformers-tokenization-mapping-and-padding-9fbfbce28028. [Accessed 5 02 2024].

[22] A. Lopardo, "Word2Vec to Transformers," Medium, 07 01 2020. [Online]. Available: https://towardsdatascience.com/word2vec-to-transformers-caf5a3daa08a. [Accessed 12 03 2024].

[23] Y. Bengio, K. Cho and D. Bahdanau, "Neural Machine Translation by Jointly Learning to Align and Translate," *International Conference on Learning Representations,* 2015.

[24] B. T, "Transformer Architectures for Dummies - Part 1 (Encoder Only Models)," Linkedin Articles, 23 12 2023. [Online]. Available: https://www.linkedin.com/pulse/transformers-architectures-dummies-part-1-encoder-only-bhaskar-t-jr94c/?trackingId=UoonxqlMST2xflVTMiYdmA%3D%3D. [Accessed 01 03 2024].

[25] B. T, "Transformer Architectures for Dummies - Part 2 (Decoder Only Architectures)," LinkedIn Articles, 02 01 2024. [Online]. Available: https://www.linkedin.com/pulse/transformer-architectures-dummies-part-2-decoder-only-bhaskar-t-hj9xc/. [Accessed 01 03 2024].

[26] S. Hochreiter, . T. Unterthiner and D.-A. Clevert, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *International Conference on Learning Representations,* 2016.

[27] Yahoo, "Yahoo Finance," Yahoo, [Online]. Available: https://finance.yahoo.com/quote/AAPL?.tsrc=fin-srch. [Accessed 23 01 2024].

[28] M. Granat, "How to Use Convolutional Neural Networks for Time Series Classification," Towards Data Science, 04 10 2019. [Online]. Available: https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57. [Accessed 24 01 2024].

[29] M. . Z. Mulla, "Word Embeddings in Natural Language Processing | NLP," Medium , 28 06 2020. [Online]. Available: https://medium.com/@zeeshanmulla/word-embeddings-in-natural-language-processing-nlp-5be7d6fb1d73. [Accessed 24 01 2024].