# Deep Learning Project

## Forex Stock Market Price Prediction Using Deep Neural Networks ( Transformer )

Shaurya Mishra (B21EE064)

## 1. Introduction

The foreign exchange (Forex) market is a global decentralized market for the trading of currencies. It is one of the largest financial markets in the world, with an average daily trading volume of over $5 trillion. Predicting the price movements of Forex markets is a challenging task, as these markets are affected by a variety of factors, including economic and political events, interest rates, and global news. In recent years, deep neural networks have shown great potential in predicting stock market prices. In this report, we explore the use of transformer-based deep neural networks for Forex stock market price prediction.

## 2. Background

Deep neural networks (DNNs) are a type of machine learning algorithm that can be used to predict stock market prices. DNNs consist of multiple layers of interconnected nodes, and they are trained using large amounts of historical data to make predictions about future stock market prices. Recently, transformer-based DNNs have gained popularity in the field of natural language processing over RNNs, LSTMs and GRUs because of their attention mechanisms allowing them to have large reference windows, they have also been applied to other domains, including stock market price prediction. We have also explored stock price prediction using RNNs and LSTMs, however, transformers superseded them in the prediction task and hence our report sticks strictly towards the analysis of stock price prediction using Transformers.

## 3. Methodology

### 3.1 Dataset Description and Processing

The dataset we have used is that of IBM stock price history. The dataset starts from 02-16-1962 and ends on the date 31-01-2020. The data contains the **Open, High, Low, Close** as well as the trading **Volume(OHLCV)** of the IBM stock for every day between the aforementioned dates, leading to a total size of 14588 entries.


The data has been extracted from Yahoo Finance, the link for the same has been provided in the references.

For this project, we will be predicting the closing price of the testing days.

The data has been preprocessed by

Normalizing the dataset using min-max normalization

split into training, validation and testing datasets in 80:10:10 ratio

The datasets have further been reshaped into (128,5), that is, a single entry now consists of the 5 feature values of 128 days. i.e. features = 5, sequence_length = 128.

### 3.2 Implementation of Time Embeddings

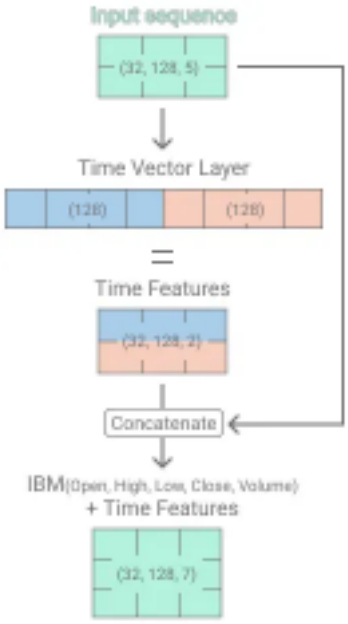
In order to encapsulate the time feature we have implemented the technique discussed in the paper **Time2Vec: Learning a Vector Representation of Time.**

**Firstly**, the authors identified that a meaningful representation of time has to include both **periodic and non-periodic patterns**.

**Secondly**, a time representation should have an **invariance to time rescaling,** meaning that the time representation is not affected by different time increments

In terms of implementation we have created a class Time2Vector. This takes a tensor input with shape (batch_size, sequence_length, features) and returns a tensor output with shape (batch_size, sequence_length, 2).

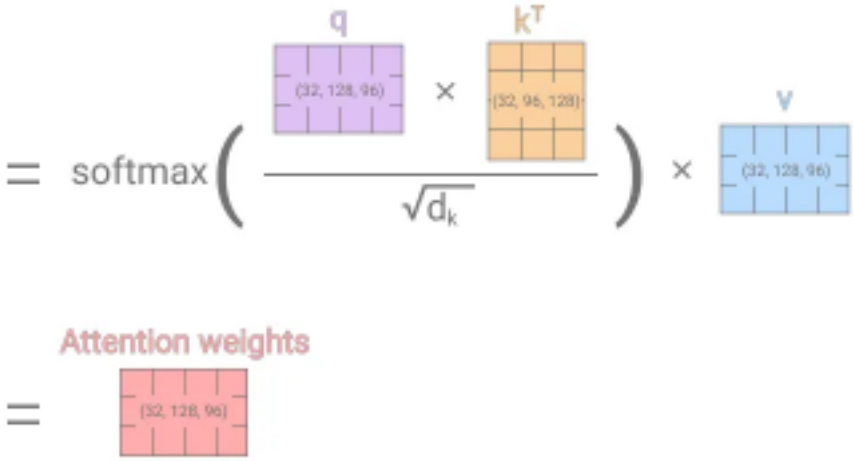
This vector is the time embedding of the input. the 2 channels indicate the periodic and non-periodic channels. we then concatenate this time embedding to the input and now we have 5+2 = 7 features.




### 3.3 Transformer Architecture and Implementation

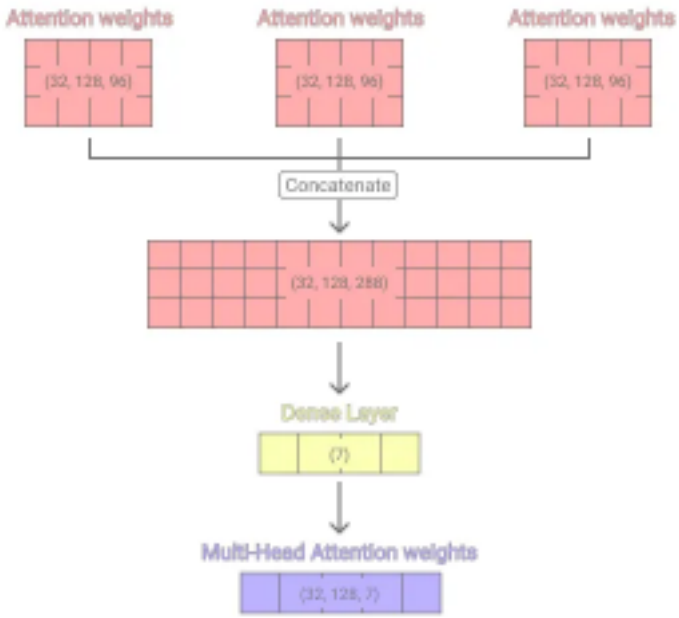
We have implemented various classes and used them to implement the transformer architecture to predict stock price. following are the classes

1. SingleAttention : This class represents the attention mechanism in the Transformer. It takes three inputs, all of which are 3D tensors: the query, the key, and the value. These inputs are processed through three dense layers to produce the query, key, and value vectors. The attention weights are calculated by taking the dot product of the query and key vectors, divided by the square root of the dimensionality of the key vector. The attention weights are then normalized using a softmax function. Finally, the output is calculated as the dot product of the attention weights and the value vector.

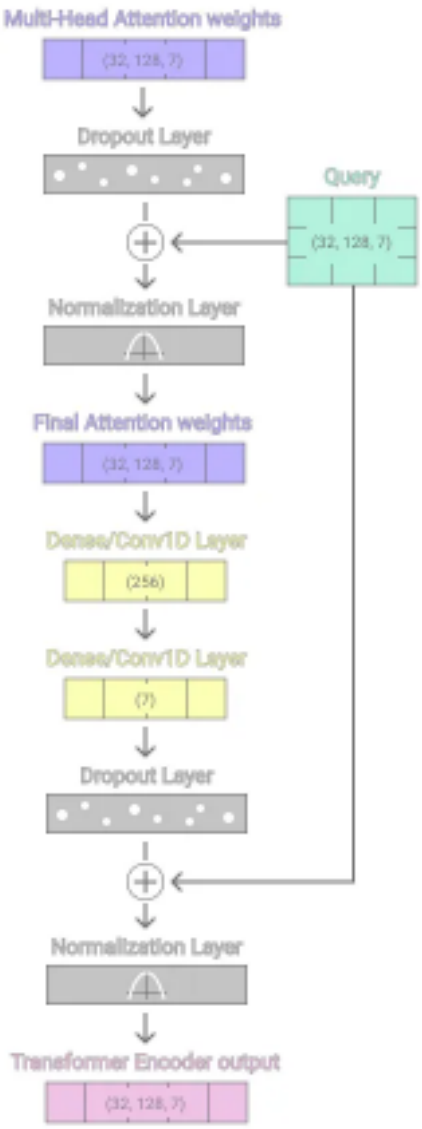





2. MultiAttention : This class represents the multi-head attention mechanism in the Transformer. It consists of multiple instances of SingleAttention , each with its own set of learnable parameters. The outputs of the individual attention heads are concatenated and passed through a dense layer to produce the final output.




3. TransformerEncoder : This class represents a single layer of the Transformer encoder. It consists of a multi-head attention layer followed by a feedforward layer. The output of the multi-head attention layer is passed through a residual connection and layer normalization before being input to the feedforward layer. The feedforward layer consists of two 1D convolutional layers with dropout and layer normalization.



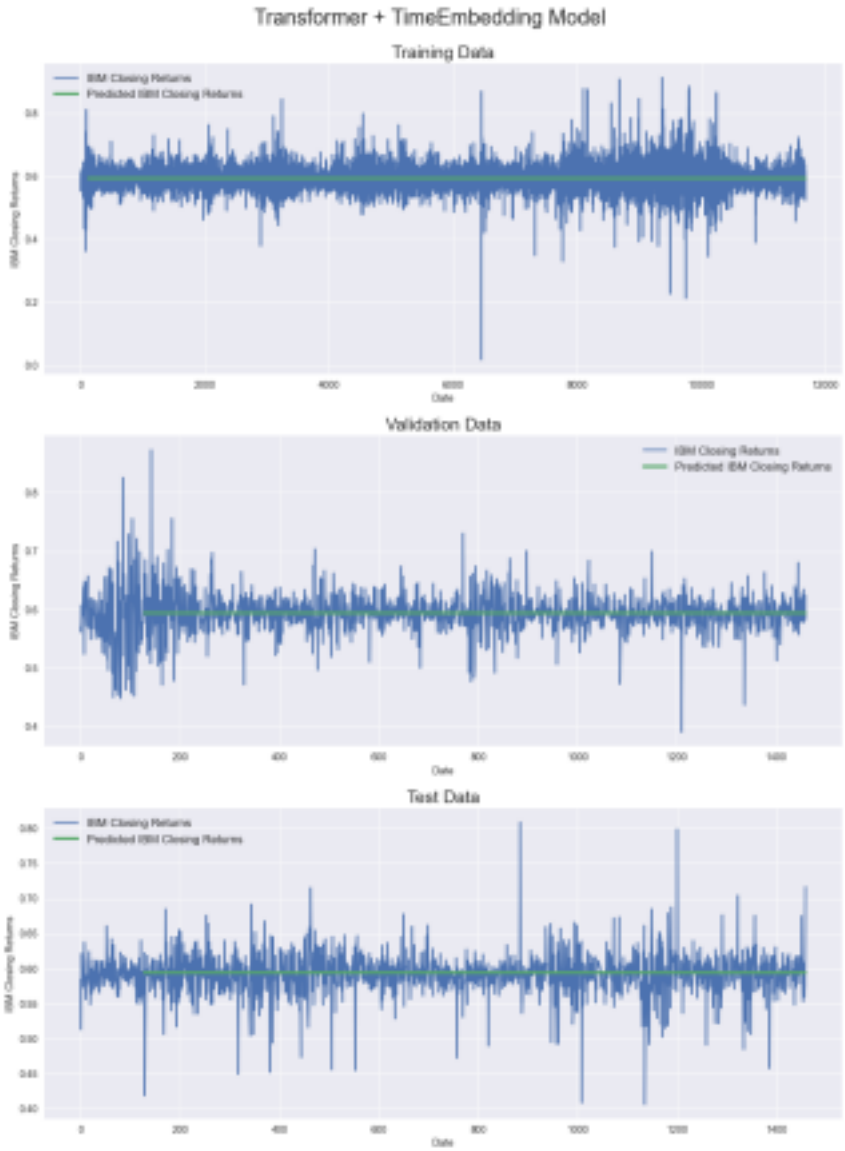

The `TransformerEncoder` class takes four parameters:

- `d_k` : the dimensionality of the key and query vectors
- `d_v` : the dimensionality of the value vectors
- `n_heads` : the number of attention heads to use
- `ff_dim` : the number of units in the feedforward layer

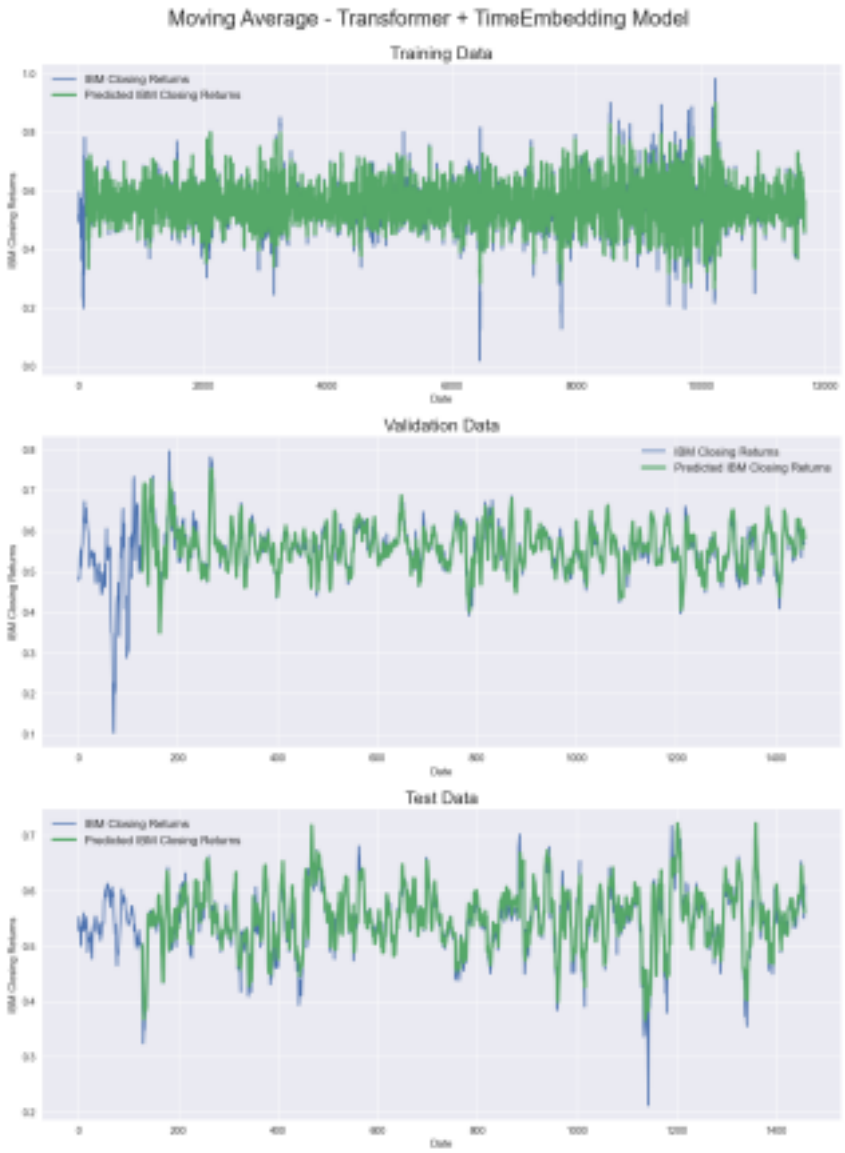
Finally we implement the model using the final classes. the model initiates 3 transformerencoders and uses them in series to generate the output and then uses further feedforward layers and normalization to provide a prediction of the closing price.

## 4. Results and further discussion

when training this model for 35 epochs we find that the results are not up to the mark. we see huge differences in predicted and actual trends.




So now we use a **Moving average,** with a sliding window of 10, which smoothes the trends. this feature engineering helps us achieve way better results with training of 35 epochs on the new data.




Further metrics to quantify the results have been provided in the ipynb file that has been submitted.

## 5. References


Dataset: https://finance.yahoo.com/quote/IBM/history?period1=-252288000&period2=1590278400&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true&guccounter=1

https://towardsdatascience.com/stock-predictions-with-state-of-the-art-transformer-and-time-embeddings-3a4485237de6

https://arxiv.org/pdf/1907.05321.pdf

https://www.youtube.com/watch?v=4Bdc55j80l8&t=3s&ab_channel=TheA.I.Hacker-MichaelPhi