Machine Intelligent Forex Trading

ALLEN HUANG (AHUA9723)

Supervisor: Dr Matloob Khushi

COMP5709 Individual Data Science Captsone Project

The University of Sydney
Australia

6 December 2020

Abstract

Every change of trend in the Forex market presents a great opportunity as well as a risk for investors. Accurate forecasting of Forex prices is a crucial element in any effective hedging or speculation strategy. However, the complex nature of the Forex market makes the predicting problem challenging, which has prompted extensive research from various academic disciplines.

Deep learning has achieved multiple state of the art results in various domains. Much of this progress is a result of exploiting certain structural features present in the data. The Forex data has a lot of these structural features as well by being a multivariate time series with graph like structure. This project proposes a new deep learning architecture GAT TPA-LSTM, combining graph attention networks (GAT) with Temporal Pattern Attention Long Short Term Memory networks (TPA-LSTM), specialised to deal with Forex data.

Contents

List of Figures

CHAPTER 1

Introduction

Deep learning has gained a lot of traction in recent years by being able to outperform humans in complex tasks in various domains. It has dramatically improved state of the art in domains such object recognition, object domain, speech recognition, natural language processing and many more. One of the reasons deep learning methods are favoured compared to traditional machine learning methods is its feature extraction abilities. Traditional machine learning methods generally rely on experts with domain knowledge to select good features as input while deep learning learns these high level abstractions.

The majority of tasks that deep learning outperforms machine learning in are tasks with structural information that only specific deep learning architecture can exploit. For example, convolutional layers are designed to deal with data with a grid like structure, graph neural nets are designed to deal with data with a graph like structure, recurrent nets and attention are designed to learn dependencies for sequential data.

Forex data has a lot of these structural features present that make it suitable for deep learning. Historical stock price is a time series with sequential information. Forex currency pairs also have a clear dependencies between each other and can be arranged in a graph like structure.

There are many previous articles that have invested instruments such as Forex price prediction by genetic algorithm (Zhang and Khushi 2020), RNN and ARIMA Zhao and Khushi 2020, CNN and LightGBM (Zeng and Khushi 2020), feature engineering with RNN (Kim and Khushi 2020) or reinforcement learning (Meng and Khushi 2019, Kim and Khushi 2020). Most of these techniques for Forex price prediction have been focused on prediction for a single Forex currency pair.

There are two main goals for this project. The first goal is to see whether or not you can improve predictions for a Forex currency pair using information from other Forex currency pairs. The second goal is to see if a deep learning model can be built that is able to deal with both Multivariate time series data and Graph structured data.

CHAPTER 2

Background and Literature Review

This section will cover the background knowledge required to understand this project. Firstly, I will cover the characteristics of the Forex dataset which will help motivate the design of the deep learning architecture. Secondly, I will cover the Deep Learning concepts that are used in this project. Thirdly, I will cover techniques from related works used in similar problems.

2.1 The Forex Dataset

Forex (also known as Foreign Exchange Market) is a global decentralized market for the exchange of national currencies against one another. A currency pair is defined as the relative value of a currency unit against the unit of another currency on the Forex market. For example, the Forex pair AUD/NZD having a value of 1.08 means 1 AUD can be exchanged for 1.08 NZD.

There are multiple reasons to trade on Forex over the regular stock market. Forex is the world's largest market which means it has substantial liquidity when compared to the regular stock market. This means it can it can more easily handle large increases in trading volume without experiencing significant changes in price. This makes the Forex prices more predictable than normal stock prices and more suitable to apply deep learning methods to learn trends and patterns in the data. There is also a 24 hour availability meaning there is more data available for training for Forex pairs than there is for stocks.

There are five main input features used to describe the movement of a Forex pair price for each time period: 1. Open price which is the opening price for the time period, 2. High price

which is the highest price for the time period, 3. Low price which is the lowest price for the time period, 4. Close price which is the closing price for the time period and 5. Volume is the amount of trades made for the time period. The first four of these features are commonly expressed in a stock market chart known as an Open-High-Low-Close chart as shown in Figure 2.1.

FIGURE 2.1: OHLC chart

Technical indicators are heuristics or mathematical calculations made using historical open, high, low, close and volume values. Technical indicators are crucial for helping traders identify and interpret patterns for future prediction. One example of a commonly used technical indicator is the moving average which filters out noise and can help give traders information on the trend of the price. Most traditional machine learning techniques would require the

Forex data has multiple properties that make it an interesting problem to apply deep learning on:

Firstly, each Forex pair is a time series so it has sequential information and dependencies. There has been a lot of research done on models dealing with sequence to sequence prediction, particularly in the field of Natural Language Processing. The most notable ones are Recurrent

Neural Networks, Long Short Term Memory networks and Attention based models such as Transformers.

Secondly, it is a multivariate time series problem with respect to a single Forex pair. Each Forex pair has five features, each of which can be considered a time series dataset by itself. We need a model that not only learns dependencies for a time series dataset, but also learn inter dependencies between other time series as well.

Thirdly, it is a multivariate time series problem with respect to all Forex pairs. There are clear connections between Forex pairs because they represent the values between currencies. This is important because it should be possible to improve predictions for a Forex pair by also using information from related Forex pairs. For example, information from AUD/USD and NZD/USD might contain important information for the prediction AUD/USD. We need a model that is able to deal with the relational information between Forex pairs.

2.2 Deep Learning

This section will cover three main aspects of Deep Learning. Firstly, it will cover fundamental deep learning concepts. Secondly, it will cover models specialised for processing sequential data. Thirdly, it will cover models specialised to process graph structured data.

2.2.1 Fundamental Knowledge

Neural Networks are heavily inspired by the biological functions of the brain. One of the first attempts at using mathematical models to mimic biological neurons is the Perceptron (Rosenblatt 1958). The Perceptron, as shown in Figure 2.2, takes in a set of inputs, multiplies each input by a learnt weight and applies a threshold function on the sum of these weights. This is analogous to a neuron receiving signals from nearby neurons, with certain neurons having stronger synaptic strengths with the current neuron and firing a signal only when the threshold potential is reached.

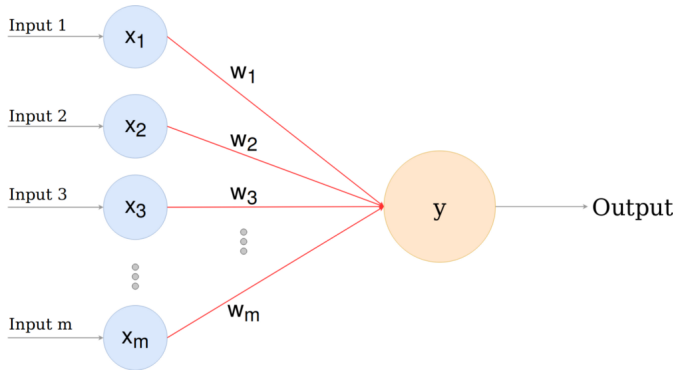

FIGURE 2.2: Perceptron

The simplest neural network is is known as a Multi-Layer Perceptron (MLP) or a Dense Neural Network. There are 3 main components in a MLP: the input layer that feeds data into the network, the hidden layers that consists of layers of neurons that learn higher level representations and an output layer that gives the predictions. An example of a MLP is shown in Fig 2.4.

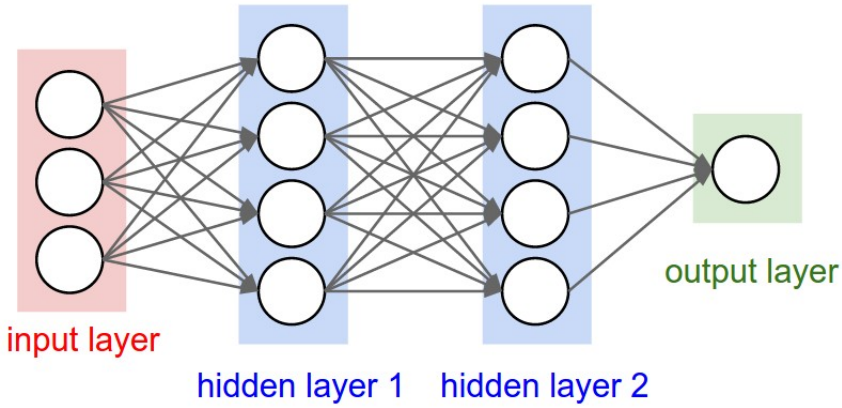

FIGURE 2.3: Multi-Layer Perceptron

Each hidden or output unit in a MLP takes a vector from the previous layer, applies an affine transformation and then applies an element wise transformation using the activation function f to obtain a hidden representation. The mathematical formula for this process is shown in equation 2.1.

$$z^{(l)} = f(W^{(l)} z^{(l-1)} + b^{(l)}) \tag{2.1}$$

The activation function is normally chosen to be a non-linear transformation. There is little interest in multi-layer networks of linear units as for any such network, we can always find an equivalent network without hidden units. This follows from the fact that the composition of linear transformations is a linear transformation. Common activation functions include: sigmoid functions such as equation 2.2, equation 2.3 and the relu function which is shown in equation 2.4. (Bishop 2006)

$$f(x) = e^x/(e^x + 1) \tag{2.2}$$

$$f(x) = \tanh(x) \tag{2.3}$$

$$f(x) = \max(0, x) \tag{2.4}$$

MLP are often used for the task of supervised learning. The goal of supervised learning is to predict a set of outputs $(\mathrm{y}_1, \mathrm{y}_2, ...\mathrm{y}_n)$ given their corresponding inputs $(\mathrm{x}_1, \mathrm{x}_2, ...\mathrm{x}_n)$. The loss function measures the difference between the predicted outputs from the model $(\hat{\mathrm{y}}_1, \hat{\mathrm{y}}_2, ...\hat{\mathrm{y}}_n)$ and the actual outputs $(\mathrm{y}_1, \mathrm{y}_2, ...\mathrm{y}_n)$. Given a loss function l, the equation for calculating loss in shown in equation 2.5.

$$L = \sum_i l(y_i, \hat{y}_i) \tag{2.5}$$

Training a neural network involves minimising a loss function. The inputs are initially forward propagated through the network using equation 2.1. The loss can then be calculated from the difference of the output of the network and the original label. For regression problems, a common loss function used for training and evaluation is the Mean-squared error as shown in equation 2.6.

$$\text{MSE} = \frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{N} \tag{2.6}$$

Backpropagation (Rumelhart et al. 1986) is the preferred method to train a neural network and it propagates information backwards through the network from the output. It is used to find the gradient of each weight of the network with respect to the loss. Gradient descent can then be applied to update the weights, this is shown in equation 2.7. The amount the weights are updated by is controlled by the learning rate hyperparameter η.

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau}) \tag{2.7}$$

2.2.2 Long Short Term Memory

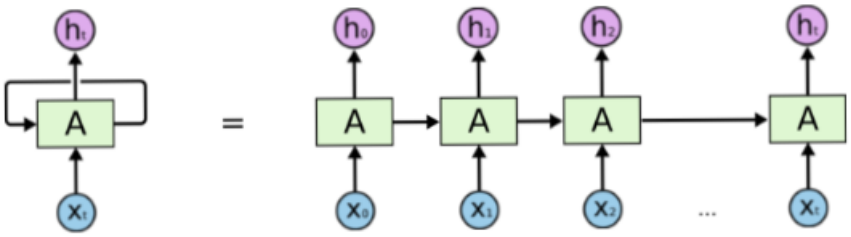

FIGURE 2.4: Recurrent Neural Network

Recurrent Neural Networks are a deep learning architecture specialised for dealing with sequential data (Rumelhart et al. 1985). Given sequential data $\{x_1, x_2, ...x_n\}$, a RNN has a recurrent function F that calculates $h_t \in R^m$ for each time step t, as shown in equation 2.8. The function F depends on what type of RNN cell is used. The main intuition behind RNN is that the recurrent function allows it to persist information through timesteps allowing the model to learn dependencies over time.

$$h_t = F(h_{t-1}, x_t) \tag{2.8}$$

Long Short Term Memory cells (Hochreiter and Schmidhuber 1997) are the most popular type of recurrent neural network because its architecture has a lot of flexibility in controlling the persisted information. LSTMs have the following recurrent function:

$$h_t, c_t = F(h_{t-1}, c_{t-1}, x_t) \tag{2.9}$$

which is defined by the following equations

$$i_t = \sigma(W_{x_i} x_t + W_{h_i} h_{t-1}) \tag{2.10}$$

$$f_t = \sigma(W_{x_f} x_t + W_{h_f} h_{t-1}) \tag{2.11}$$

$$o_t = \sigma(W_{x_o} x_t + W_{h_o} h_{t-1}) \tag{2.12}$$

$$f \odot c_{t-1} + i_t \odot tanh(W_{x_g} x_t + W_{h_g} h_{t-1}) \tag{2.13}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.14}$$

A diagram of a LSTM cell is shown in Figure 2.5.

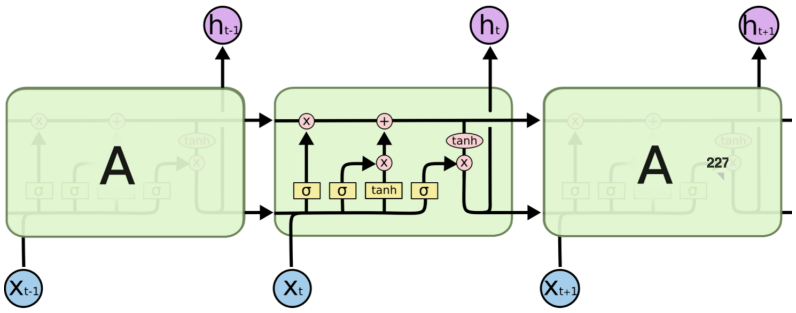

FIGURE 2.5: Long Short Term Memory Cell

2.2.3 Attention

The attention mechanism was first introduced for dealing with the problem of sequence to sequence natural language processing tasks (Bahdanau et al. 2015) and has become one of the most influential deep learning concepts introduced in recent years. There are multiple

models that use attention to achieve state of the art results in the fields of Natural Language Processing (Vaswani et al. 2017) and Computer Vision (Zheng et al. 2017).

The Transformer is a neural network architecture based on an encoder decoder architecture that has achieved state of the art results for various sequence to sequence natural language processing tasks (Bahdanau et al. 2015, Vaswani et al. 2017). Transformers were developed to solve the problem of neural machine translation which is any task that transforms an input sequence to an output sequence, such as language translation and speech recognition. An example of a transformer is shown in Figure 2.6.

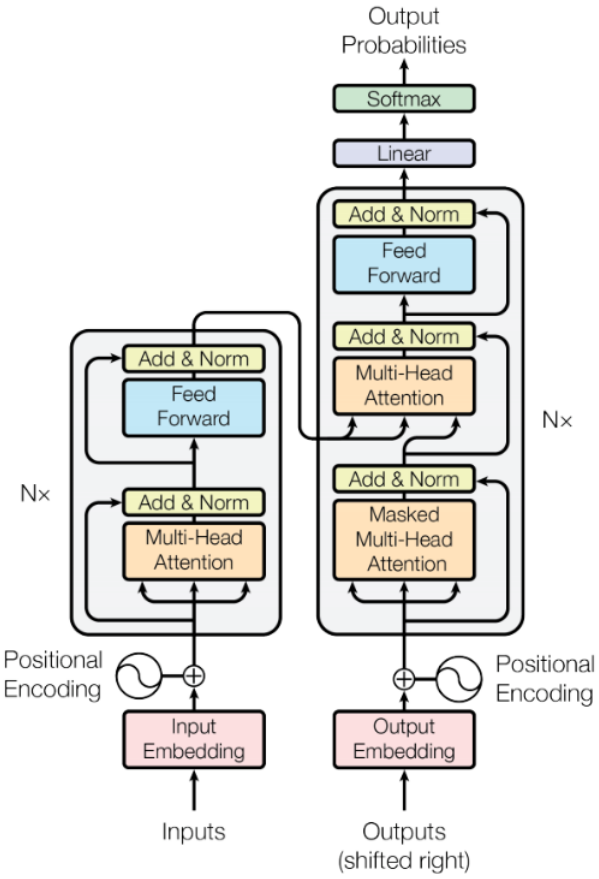

FIGURE 2.6: Transformer

One of the key mechanisms of a Transformer is the attention mechanism. An attention function is the mapping of a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values where the weight assigned to each value is

computed by a compatibility function of the query with the corresponding key. The formula for scaled dot product attention is shown in Formula 2.15.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\text{T}}}{\sqrt{n}})\mathbf{V} \tag{2.15}$$

Multi head attention involves stacking multiple attention layers in parallel. Multi head attention allows the model to jointly attend to information from different representation subspaces at different position. With a single attention head, averaging inhibits this (Vaswani et al. 2017).

The attention has also been successfully incorporated into RNN models to improve performance, this is known as Luong attention (Luong et al. 2015). Given the previous hidden states $H = \{h_1, h_2, ...h_n\}$, a context vector v_t is extracted from the previous states. v_t is the weighted sum of each column h_i in H and represents the information relevant to the current time step. v_t is further integrated with the present state h_t to yield the prediction. Assuming a scoring function f which calculates the relevance between its input vectors, we have the following formulas

$$\alpha_i = \frac{\exp(f(h_i, h_t))}{\sum_{j=1}^{t-1} \exp(f(h_j, h_t))} \tag{2.16}$$

$$v_t = \sum_{i=1}^{t-1} \alpha_i h_i \tag{2.17}$$

2.2.4 Graph Convolution Networks

Graphs are a data structure used to model relational data. A graph is a model consisting of a set of objects (nodes) and their relationships (edges). It is an ordered pair *G* = (*V*,*E*) consisting of

- *V*, a set of nodes
- $E \subseteq \{\{x, y\} | x, y \in V \text{and } x \neq y\}$, a set of edges

Graph Neural Networks (GNN) are a broad range of deep learning methods focused to operate on the graph domain. (Zhou et al. 2018, Scarselli et al. 2009). The first motivation for GNNs come from convolutional neural networks (CNN) which has achieved state of the art results in the computer vision domain. CNNs have the ability to extract spatial features over data with a grid like structure and can compose them to construct highly expressive representations. The key features behind CNN's success are its local connection, shared weights and the use of multiple layers to learn higher level features. The second motivation comes from graph embedding which learns to represent graph nodes, edges or sub graphs in low-dimensional vectors.

Graph Convolution Networks are based on CNNs and graph embedding and collectively aggregate information from the graph structure. By stacking the graph convolutional layers, higher level representations from the graph structure can be obtained. An example of a GCN is shown in Figure 2.7.

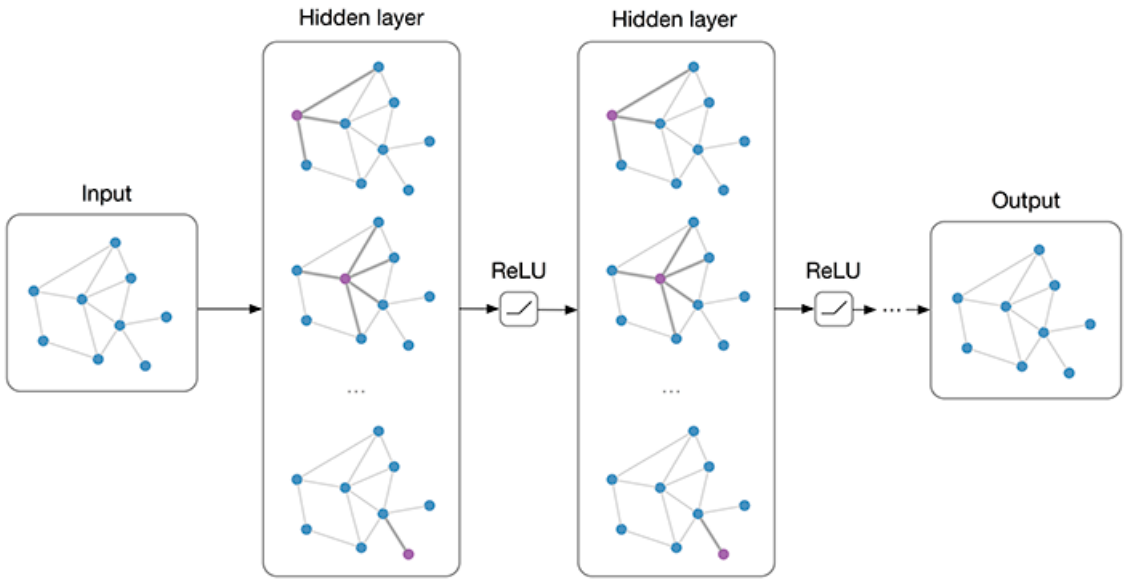

FIGURE 2.7: Graph Neural Network

GCNs use the spectral approach (Bruna et al. 2014) to forward propagate information through the GCN. The GCN aggregator is given in equation 2.18 and the GCN updater is given in equation 2.19.

$$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X} \tag{2.18}$$

$$\mathbf{H} = \mathbf{N}\Theta \tag{2.19}$$

2.2.5 Graph Attention Networks

Graph attention networks (GAT) are a deep learning architecture that operates on graph structured data by leveraging masked self-attentional layers (Velickovic et al. 2018). By stacking layers in which nodes are able to attend over their neighborhoods' features, the model can learn which nodes to pay attention and attend to. This addresses one of the main shortcomings of GCN methods which assigns a non-parametric weight via the normalization function during neighbourhood aggregation.

The input to the GAT layer is a set of node features $\mathbf{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, ...\overrightarrow{h_N}\}, \overrightarrow{h_i} \in \mathbb{R}^F$, where N is the number of nodes and F is the number of features in each node. The layer produces a set of node features $\mathbf{h}'' = \{\overrightarrow{h_1}'', \overrightarrow{h_2}'', ...\overrightarrow{h_N}''\}, \overrightarrow{h_i}'' \in \mathbb{R}^{F''}$ as output. There are 4 main components inside a GAT layer.

The first component is a simple linear transformation. This is to obtain sufficient expressive power to transform the input features into higher level input features. The equation in Formula 2.20 gives an output gives an output $\overrightarrow{h_i}' \in \mathbb{R}^{F'}$.

$$\overrightarrow{h_i}' = \boldsymbol{W}\overrightarrow{h_i} \tag{2.20}$$

Self attention is then applied on the nodes. This is a shared attentional mechanism $a : \mathbb{R}^{\mathbb{F}'} \times \mathbb{R}^{\mathbb{F}'} \longrightarrow \mathbb{R}$ that computes attention coefficients $e_{i,j}$. $e_{i,j}$ indicates the importance of node j's features to node i. Graph structure is incorporated into the mechanism by performing masked attention, where $e_{i,j}$ is only computed for $j \in N_i$ where N_i is some neighborhood of

node i in the graph. The main neighborhood used is the first order neighbours of i, include i itself.

$$e_{i,j} = LeakyReLU(\overrightarrow{a}^T[\overrightarrow{h_i}'\|\overrightarrow{h_j}']) \tag{2.21}$$

This is followed by a softmax operation which normalizes the attention coefficients.

$$\alpha_{i,j} = softmax(e_{i,j}) = \frac{\exp(e_{i,j})}{\sum_i \exp(e_{i,j})} \tag{2.22}$$

The embeddings from neighbours are aggregated together and scaled by attention scores giving $\overrightarrow{h_i''}$.

$$\overrightarrow{h_i''} = \sigma(\sum_i \alpha_{i,j}\overrightarrow{h_i'}) \tag{2.23}$$

A diagram showing how the GAT mechanism is constructed for a single output node is shown in Figure 2.8.

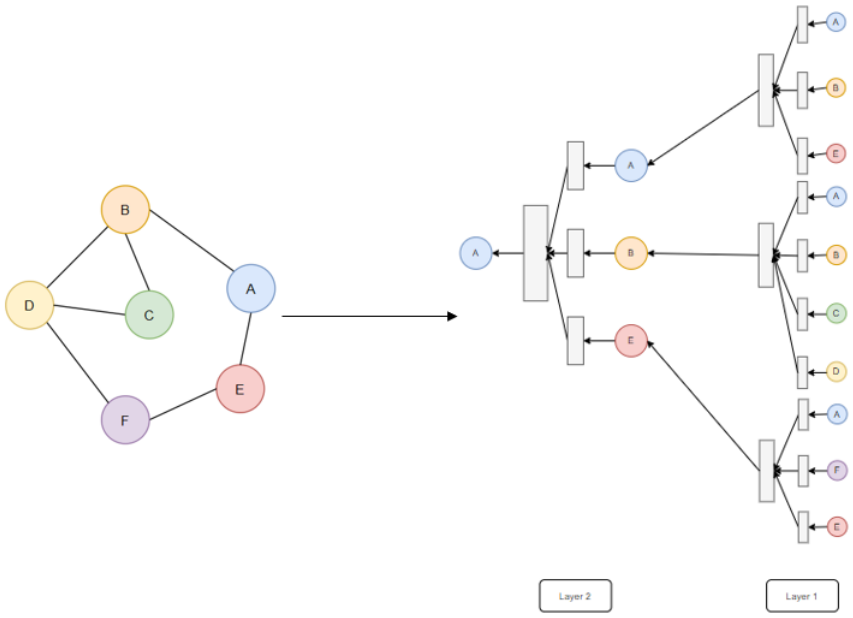

FIGURE 2.8: Graph Attention Network

2.3 Related Works

2.3.1 Multivariate Time Series Forecasting

Multivariate time series (MTS) data refers to time series data with multiple variables (Lai et al. 2017). There are often complex inter-dependencies between different variables that deep learning models need to learn to get a good forecast. Formally, given a series of fully observed time series signals $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_W\}$ where $\mathbf{y}_w \in R^n$, where n is the variable dimension and W is the window size, we aim to predict future signals in a rolling forecasting fashion. That is, to predict $\mathbf{y}_{W+\Delta}$ where Δ is the desired horizon ahead of the current timestamp. Likewise, to predict the value of the next time stamp $\mathbf{y}_{W+\Delta+1}$, we use the signals $\mathbf{Y} = \{\mathbf{y}_2, \mathbf{y}_3, ...\mathbf{y}_{W+1}\}$ to ensure a constant input size to the deep learning model.

The Forex data can be formulated as a MTS problem in 2 ways. Firstly, it can be formulated as a MTS problem with respect to each input feature (open price, high price, low price, close price and volume). Secondly, can be formulated as a MTS problem with respect to each Forex pair. Although it is possible to formulate the problem as a single MTS problem, we want some method of exploiting the relational structure between Forex pairs.

2.3.2 LSTNet

The Long and Short-Term Time-Series Network (LSTNet) is one of the first deep learning models designed for MTS forecasting (Lai et al. 2017). The first layer consists of LSTNet consists of CNNs to capture short term patterns. The second layer consists of using LSTMs or GRU to predict long-term dependencies. Recurrent skip layers are used in combination with LSTMs or GRUs to overcome the problem of training instability and the vanishing gradient problem associated with trying to learn long term dependencies with LSTMs and GRUs. The final layer is a fully connected dense layer and an element wise sum of the outputs. A diagram of the LSTNet is shown in Figure 2.9. Attention may also be incorporated with a LSTNet by selecting relevant final hidden states of the recurrent skip layer. .

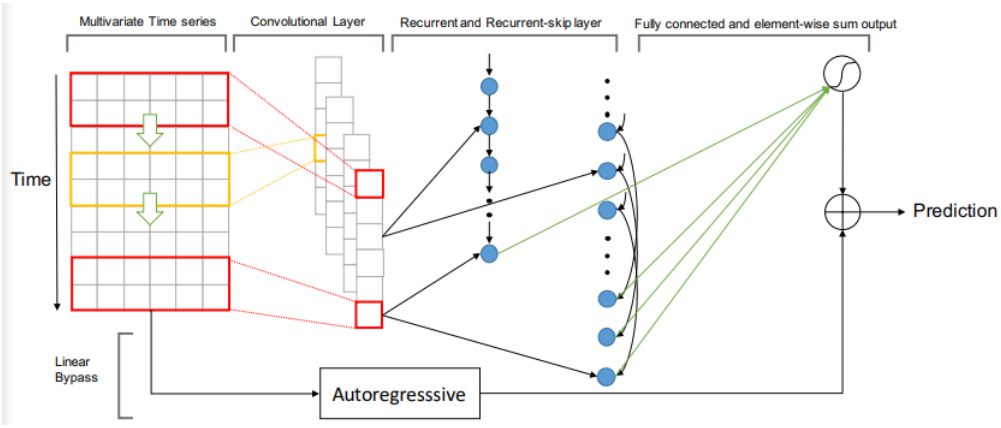

FIGURE 2.9: LSTNet

However, LSTNet is only designed to work for MTS with periodic patterns. The recurrent skip layer must be manually tuned to be similar to the periodic pattern. This is not suitable to be used on the Forex dataset since there is no consistent periodic pattern.

2.3.3 Temporal Pattern Attention LSTM

The Temporal Pattern Attention (TPA) mechanism is the current state of the art model for MTS forecasting (Shih et al. 2018). Temporal pattern attention is able to select relevant time series which is a more suitable mechanism for MTS data than selecting relevant hidden states in the LSTNet-Attn model. As a result, TPA-LSTM works on both periodic and non-periodic data and is the current state of the art for MTS forecasting problems.

Temporal patterns in the MTS data are detected using CNNs. CNN filters are applied on the row vectors of the previous RNN hidden states H. Specifically, we have k filters $C_i \in \mathbb{R}^{1\times T}$, where T is the maximum length of attention we are paying to. Convolutional operations yield $H^C \in \mathbb{R}^{n\times k}$ where $H^C_{i,j}$ represents the convolutional value of the i-th row vector and the j-th filter. Formally, this operation is given by

$$H^C_{i,j} = \sum_{l=1}^{w} H_{i,(t-w-1+l)} \times C_{j,T-w+l} \tag{2.24}$$

The context vector v_t is calculated as a weighted sum of the row vectors of H^C. The scoring function $f : \mathbb{R}^k \times \mathbb{R}^m \longrightarrow \mathbb{R}$ used to evaluate relevance is defined below

$$f(H_i^C, h_t) = (H_i^C)^T W_a h_t \tag{2.25}$$

where H_i^C is the i-th row of H^C, and $W_a \in \mathbb{R}^{k \times m}$. The attention weight α_i is obtained as

$$\alpha_i = sigmoid(f(H_i^C, h_t)) \tag{2.26}$$

The sigmoid function is used instead of softmax since more than one variable is expected to be useful for forecasting.

The context vector $v_t \in \mathbb{R}^k$ is obtained by weighting the row vectors of H^C by α_i

$$v_t = \sum_{i=1}^{n} \alpha_i H_i^C \tag{2.27}$$

Then v_t and h_t are integrated together to yield the final prediction

$$h'_t = W_h h_t + W_v v_t \tag{2.28}$$

$$y_{t+\Delta} = W_{h'} h'_t \tag{2.29}$$

where $h_t, h'_t \in \mathbb{R}^m$, $W_h \in \mathbb{R}^{m \times m}$, $W_v \in \mathbb{R}^{m \times k}$, and $W_{h'} \in \mathbb{R}^{n \times m}$ and $y_{t+\Delta} \in \mathbb{R}^n$

A diagram showing an overview of this process is shown in Figure 2.10.

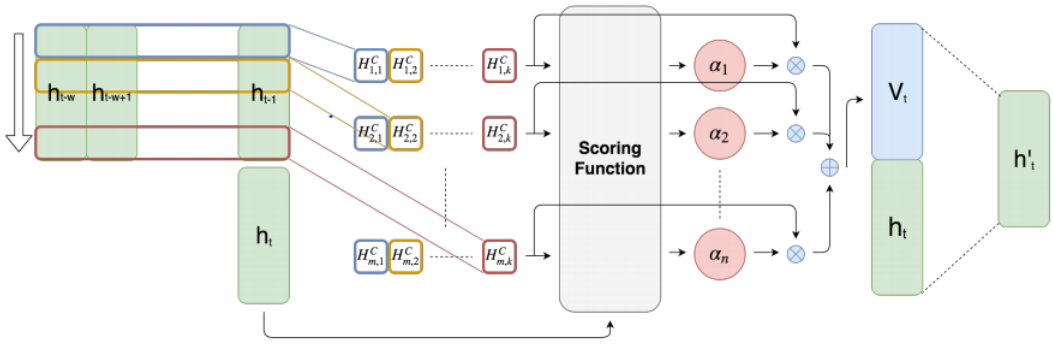

FIGURE 2.10: Temporal Pattern Attention

CHAPTER 3

Research Problems

3.1 Research Aims & Objectives

The overall aim of this project is to use deep learning to develop new methods to predict the Forex stock price. There are financial incentives for this project if a good model is able to be built and this is most likely also the reason there is little research on this area.

There are multiple objectives that this project plans to achieve. Firstly, I need to come with a deep learning model that is specialised for Multivariate Time Series data with graph structured features between the variables. Secondly, I need the show the effectiveness of this model by comparing it to (simpler) baseline models with a similar number of parameters. Thirdly, if there is enough time, to back test on unseen Forex trading data.

3.2 Research Questions

The main research questions considered for this project is: Is it possible improve the predictions of for a Forex currency pair using information from other Forex currency pairs? (e.g. using information from AUDUSD to improve the prediction of AUDNZD).

3.3 Research Scope

This project involves experimenting with various deep learning techniques that I believe should work well on the Forex dataset. This might not necessarily result in an improvement

on existing methods and might not be profitable due to the unpredictability and volatility of the markets. This project will also be more focused on the theoretical side of things, that is the first two objectives. The third objective may be completed if there is enough time. There may be issues with the proposed method and state of the art methods, such as model size, that might make it perform poorly during back testing.

CHAPTER 4

Methodologies

4.1 Data Collection and Preprocessing

4.2 Data Collection

In total, 20 Forex currency pairs collected from the online database forexsb are used for this project. The following Forex currency pairs are used: AUD/USD, AUD/JPY, EUR/USD, AUD/EUR, JPY/USD, JPY/EUR, AUD/CHF, GBP/USD, AUD/CAD, CHF/USD, CAD/USD, AUD/GBP, JPY/GBP, CAD/EUR, EUR/CHF, GBP/EUR, CAD/JPY, JPY/CHF, NZD/USD and AUD/NZD.

Forex data with 1 day time periods from 2007 to 2020 are used for this project. The proposed deep learning model does not use technical indicators and focuses on using deep learning to learn higher level features directly from the data. This means that it does not use technical indicators which are summaries of historical prices like 100 day moving averages and all the time periods need to be fed in as input. The proposed deep learning model takes in 60 time periods as input. Although using a smaller time period would be more useful for prediction, it also drastically increases the model size.

4.2.1 Data Preprocessing

The Forex data for the different Forex currency pairs are first aggregated together by the date giving a dataset with 4144 data points. There are around 20 data points missing values for one

of the features and they are removed from the dataset. Overall, this gives a dataset containing 4124 data points with 100 variables, 5 variables for each Forex currency pair.

This is then formulated as a multivariate time series problem. Given a window size w and horizon Δ, we want to use $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_W\}$ to predict $\mathbf{y}_{W+\Delta}$ where $\mathbf{y}_i \in \mathbf{R}^{100}$. This gives 4124 - ($W$ + Δ) + 1 data points to work with.

Data is split up into training data, validation data and testing data in the ratio of 0.6:0.2:0.2 in chronological order. Normally the data is shuffled and k-fold stratified cross validation is used to evaluate performance in deep learning models. This is not used because we are working with a time series dataset and we do not want validation data or testing data being leaked into the training data.

Normalisation is often applied on data to speed up and stabilise training for deep learning models. There are normally two types of normalisation considered for MTS forecasting: applying min-max normalisation over all the whole dataset or min-max normalisation over each variable. For the problem of forecasting Forex currency pair prices, min-max normalisation is more suitable since the Forex currency prices do not lie close together and have very different ranges in values. The formula for min-max normalisation is shown below

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4.1}$$

4.3 GAT TPA-LSTM

4.3.1 Representing Forex currency pairs as a Graph

A Forex currency pair represents the value of a currency against another. This means that there is no currency pair that is ever traded independently from others and all the currency pairs are interlinked with one another. These relationships between Forex currency pairs can be represented in a graph structure.

An undirected graph can be used to represent the relationships between Forex currency pairs. To construct a graph for a set of currency pairs, a node is created to represent each currency pair and a connection is made between two nodes if the two currency pairs they represent have a shared currency between them. This is the simplest method to model the relationships between currency pairs, an alternative more complex method is proposed in future work.

An example of the graph model constructed from AUD/USD, AUD/NZD, NZD/USD, JPY/USD is shown in Figure 4.1.

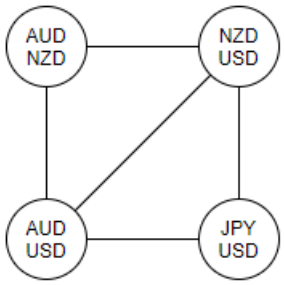

FIGURE 4.1: Example Forex Currency Pairs Graph

The full graph model when all 20 Forex currency are considered is shown in Figure 4.2.

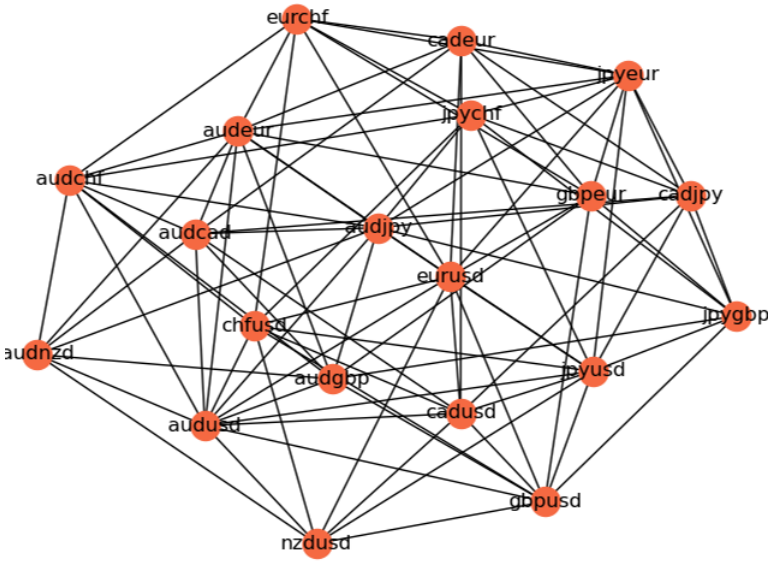

FIGURE 4.2: Complete Forex Currency Pairs Graph of 20 Forex Currency Pairs

4.3.2 GAT TPA-LSTM Layer

To the best of our knowledge, there is currently no existing deep learning model that is suitable for dealing with data that is a multivariate time series with a graph like structure between the variables. A new deep learning architecture that combines both the TPA-LSTM's ability to handle MTS data and the GAT's ability to handle graph structured data is proposed. The key idea is replace the standard attention mechanism with Temporal Pattern Attention when data is forward propagated through the graph attention layer. This allows embeddings with both MTS and graph structured properties to be learnt.

The first component of the GAT TPA-LSTM layer is similar to GAT. The input to the GAT TPA-LSTM layer is a set of node features $\mathbf{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, ...\overrightarrow{h_N}\}, \overrightarrow{h_i} \in \mathbb{R}^F$, where N is the number of nodes and F is the number of features in each node. The layer produces a set of node features $\mathbf{h'''} = \{\overrightarrow{h_1}''', \overrightarrow{h_2}''', ...\overrightarrow{h_N}'''\}, \overrightarrow{h_i}''' \in \mathbb{R}^{F'''}$ as output.

The first component is a simple linear transformation, the same as a GAT layer. This is to obtain sufficient expressive power to transform the input features into higher level input features. The equation in Formula 4.2 gives an output gives an output $\overrightarrow{h_i}' \in \mathbb{R}^{F'}$.

$$\overrightarrow{h_i}' = \boldsymbol{W}\overrightarrow{h_i} \tag{4.2}$$

The embedding $\overrightarrow{h_i}'$ is then passed through at least one LSTM layer. This is to obtain sufficient expressive power to learn higher level sequential features. The equation is given in Formula 4.3 where F represents LSTM Layer(s). This gives gives two outputs where $\overrightarrow{h_i^{h''}} \in \mathbb{R}^{F''}$ is the LSTM output and $\overrightarrow{h_i^{c''}} \in \mathbb{R}^{F''}$ is the LSTM hidden state.

$$\overrightarrow{h_i^{h''}}, \overrightarrow{h_i^{c''}} = F(\overrightarrow{h_i'}) \tag{4.3}$$

Temporal Pattern Attention is now computed instead of the standard attention mechanism on the LSTM hidden state $\overrightarrow{h_i^{c''}}$. Graph structure is incorporated into the mechanism by performing masked convolution on only the row vectors $j \in N_i$ where N_i is some neighborhood of node

i in the graph. Let G denote the application of equation 2.24 on only the row vectors $j \in N_i$ and G_i denote the i-th row of G.

$$f(G_i, \overrightarrow{h_i^{c\prime\prime}}) = (G_i)^T \mathbf{W}_a \overrightarrow{h_i^{c\prime\prime}} \tag{4.4}$$

$$\alpha_i = sigmoid(f(G_i, \overrightarrow{h_i^{c\prime\prime}})) \tag{4.5}$$

The sigmoid function is used instead of softmax since more than one variable is expected to be useful for forecasting.

The context vector is obtained by weighting the row vectors of G_i by α_i and integrated with $\mathbf{h}_i^{h\prime\prime}$, which represents the vectors $\overrightarrow{h_j^{\prime\prime}}$ for the row vectors $j \in N_i$, to give $\overrightarrow{h_i^{\prime\prime\prime}}$.

$$\overrightarrow{h_i^{\prime\prime\prime}} = \mathbf{W}_o \mathbf{h}_i^{h\prime\prime} + \mathbf{W}_v \sum \alpha_i G_i \tag{4.6}$$

The GAT TPA-LSTM layer can then be stacked together to extract higher level features with both MTS and graph structured properties.

4.4 Metrics

The following evaluation metrics are commonly used to evaluate performance for multivariate time series forecasting: Relative absolute error (RAE), Root relative squared error (RSE) and the empirical correlation coefficient (CORR).

$$\mathbf{RAE} = \frac{\sum_{t=t_0}^{t_1} \sum_{i=1}^{n} |(y_{t,i} - \hat{y}_{t,i})|}{\sum_{t=t_0}^{t_1} \sum_{i=1}^{n} |(\hat{y}_{t,i} - \overline{\hat{y}_{t_0:t_1,1:n}})|} \tag{4.7}$$

$$\mathbf{RSE} = \frac{\sqrt{\sum_{t=t_0}^{t_1} \sum_{i=1}^{n} (y_{t,i} - \hat{y}_{t,i})^2}}{\sqrt{\sum_{t=t_0}^{t_1} \sum_{i=1}^{n} (\hat{y}_{t,i} - \overline{\hat{y}_{t_0:t_1,1:n}})^2}} \tag{4.8}$$

$$\textbf{CORR} = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{t=t_0}^{t_1}(y_{t,i} - \overline{y_{t_0:t_1,i}})(\hat{y}_{t,i} - \overline{\hat{y}_{t_0:t_1,i}})}{\sqrt{\sum_{t=t_0}^{t_1}(y_{t,i} - \overline{y_{t_0:t_1,i}})^2 \sum_{t=t_0}^{t_1}(\hat{y}_{t,i} - \overline{\hat{y}_{t_0:t_1,i}})^2}} \tag{4.9}$$

RAE and RSE are the normalized version of the mean average error (MAE) and root mean square error (RMSE). CORR is a normalized version of correlation. Using normalized metrics are necessary since the time series for each Forex pair may be on different scales and this is needed for a valid comparison. For RAE and RSE, the lower they are, the better the model is performing. For empirical correlation, the higher it is, the better the model is performing.

4.5 Training

Mini batch learning involves doing training passes to the deep learning model in mini batches rather than the whole batch. The batch size is a hyperparameter that controls the trade off between performance and convergence speed. Larger batch sizes generally converge faster but generalise worse on unseen data. Smaller batch sizes introduces more noise during training since the gradient computed in the mini batch is less representative of the true gradient. This makes it harder to converge but also lets the model generalise better on unseen data. A batch size of 32 with 50 epochs is used to train the GAT TPA-LSTM model. This is chosen because the larger models already take a relatively long time to train, the larger models take up to 12 hours.

Adam, the phrase originating from adaptive momentum, is used as the optimizer to train the model. It is one of the most popular optimization algorithms that works well with mini batch learning for deep learning models. The Adam optimizer combines momentum with an adaptive learning rate for different parameters for back propagation. The model is trained using a learning rate of 5e-3.

Dropout is a popular regularisation technique used during training to improve the generalisation of deep learning model. Hidden units are randomly dropped out during training during the forward pass and any weight updates are not applied to the hidden unit during the backward pass. Dropout is not applied during validation or testing. This technique approximates training

a large number of neural networks with different architectures in parallel and introduces noise during training which helps with generalisation. Dropout is applied to the LSTM hidden units with a rate of 0.2

Both exponential learning rate decay and gradient clipping are used to help with convergence during training. Exponential learning rate decay exponentially decays the learning rate during training to help the model converge to local minimas instead of bouncing around local minimas when the learning rate is too high. Gradient clipping is often used with deep recurrent neural networks to deal with the problem of exploding gradients. This involves clipping the gradient if the value is too large. Exponential learning rate decay is applied at a rate of 0.995 every 1000 global steps and the gradient is clipped with it is larger than 5.0.

4.6 Baseline Models

There are multiple baseline models used to show the effectiveness of the proposed GAT TPA-LSTM model. In order to obtain a valid comparison, the GAT TPA-LSTM model and the baseline models need to have a similar number of trainable parameters. However, increasing the number of parameters for simpler baseline models to have the same number of parameters as the GAT TPA-LSTM model often results in overfitting of the simple baseline models resulting in a bad generalisation on unseen data. Bayesian optimisation is used on baseline models so the total number of trainable parameters is equal to or less that of the GAT TPA-LSTM model.

The final GAT TPA-LSTM model trained on 20 Forex currency pairs is trained using the following hyperparameters: Number of GAT TPA-LSTM Layers = 3, Number of LSTM layers in a single GAT TPA-LSTM layer = 1, Number of LSTM hidden units = 8. In total, the model has 145745 trainable parameters.

The first baseline model is a simple LSTM model. Since LSTMs are not designed for MTS data, every variable in the MTS is passed through a LSTM model and the LSTM outputs

are passed through dense layers to give the final output. This is a simpler model than GAT TPA-LSTM and the tuned hyperparameters are larger than those of the GAT TPA-LSTM.

The second baseline model is a TPA-LSTM model. There is no graph layers compared to the GAT TPA-LSTM and the data is passed through at least 3 TPA-LSTM layers instead. This is a simpler model than GAT TPA-LSTM and the tuned hyperparameters are larger than those of the GAT TPA-LSTM.

The third baseline model is a simple fusion model that first passes the data through a TPA-LSTM model and then passing the outputs through a GAT model. This model has a similar complexity to GAT TPA-LSTM and the tuned hyperparameters have similar values.

CHAPTER 5

Resources

5.1 Software

(1) Pycharm IDE with Anaconda Virtual Environment

(2) Tensorflow for implementing deep learning models

(3) Networkx for representing graphs

(4) Standard data science softwares such as NumPy, Scikit-Learn, Pandas and Matplotlib

(5) Weights and Biases for Hyperparameter Tuning using Bayesian Optimisation

5.2 Hardware

The project is run on AWS Sagemaker - Deep Learning AMI (Ubuntu 18.04) Version 36.0 on t2.2xlarge. This has 8 vCPUs and 32GB worth of RAM. Training the larger GAT TPA-LSTM models is memory intensive and requires the larger amounts of processing power and RAM. Smaller models may be trained on less computing resources.

5.3 Setup

(1) Create a new anaconda virtual environment with Python version = 3.6
conda init myenv python=3.6

(2) Activate newly created anaconda virtual environment
conda activate myenv

(3) Install required dependencies using git
pip install −r requirements.txt

(4) Run main.py file and pass in arguments through the command line

python main.py

−−mode train

−−num_var 20

−−horizon 3

−−num_graph_layers 3

CHAPTER 6

Milestones

6.1 Project Deliverables

(1) Code for GAT TPA-LSTM

(2) Trained Tensorflow model of GAT TPA-LSTM on 20 Forex currency pairs

(3) Final written report

(4) Presentation

6.2 Implications

The main motivation for this project is that there is a financial incentive if a model that can accurately predict Forex currency pair prices can be built. This project explores the application of state of the art deep learning techniques for predicting Forex currency pair prices. In particular, this project aims to come up with a deep learning architecture that is suitable for Multivariate Time Series Forecasting problems with Graph Structured data between the variables.

6.3 Project Plan

The project plan summarising what has been achieved during this project is shown in Table 6.1. There are two other tasks not listed in the table that I had originally planned to complete but ran into difficulties while trying to complete them. The first task is fine-tuning the hyperparameters for the GAT TPA-LSTM model. This was mainly due the time taken to

Milestone	Tasks	Reporting
Week 2	Data Collection + Literature Review	
Week 3	Literature Review	
Week 4	Proposal Report Due	Proposal Report
Week 5	Coding GAT TPA-LSTM	
Week 6	Coding GAT TPA-LSTM	
Week 7	Preliminary GAT TPA-LSTM results Preliminary baseline model results	Preliminary GAT TPA-LSTM results Preliminary baseline model results
Week 8	Progress Report Due	Progress Report
Week 9	Training GAT TPA-LSTM + TPA-LSTM while varying # Forex currency pairs	GAT TPA-LSTM and TPA-LSTM results while varying # Forex currency pairs
Week 10	Hyperparameter Optimization for baseline models	Baseline model results
Week 11	Writing Final Report	
Week 12	Writing Final Report + Presentation	
Week 13	Writing Final Report + Presentation	Final Report and Presentation

TABLE 6.1: Project Milestones

evaluate one set of hyperparamters taking longer than expected (took around 12 hours). It would take approximately a month training on a single EC2 instance and cost quite a lot for the cloud computing resources as well. The second task is back testing on Forex data. This was mainly due to the back testing code being unable to run on an Ubuntu OS and also me not having any background knowledge on how trading strategies are performed.

CHAPTER 7

Results

7.1 Baseline Model Comparisons

All the baseline models are trained on 20 Forex currency pairs; with hyperparameter optimisation for only the baseline models. The performance of the models evaluated using the 3 metrics RAE, RSE and CORR are shown below in Table 7.1.

Model	RAE	RSE	CORR
LSTM + Dense Layers	0.01742	0.19101	0.92342
TPA-LSTM	0.01556	0.10932	0.96291
TPA-LSTM feature extractor + GAT	0.01542	0.11167	0.96312
GAT TPA-LSTM	**0.01465**	**0.10409**	**0.96598**

TABLE 7.1: Baseline Model Comparisons

The first baseline model consisting of LSTM and Dense layers has the worst performance. This is to be expected since LSTMs are only designed to work on sequential data with a single variable. Using just dense layers is not suitable to learn inter dependencies between different variables in a MTS and is also not suitable to learn structural features from graph structural data data.

Compared to just LSTM + Dense layers, the TPA-LSTM performs much better. This is to be expected since TPA-LSTM is designed for MTS data and is able to learn the inter dependencies between different variables in a MTS. This model however does not factor in the graph structured nature of Forex currency pairs.

The TPA-LSTM feature extractor + GAT also does not perform better the TPA-LSTM model. This model passes the data through 2 different deep learning architectures, the first is to deal

with the MTS data and the second is to deal with the graph structured data. I believe the main reason this model does not perform better than just TPA-LSTM is because there is an overlap between learning inter dependencies between variables and learning the graph structured relationships between the variables.

GAT TPA-LSTM has the best results for all of the metrics by having the lowest value for RAE, RSE and having the highest value for CORR. This means that GAT TPA-LSTM gives the most accurate predictions when compared to the baseline models. The improvement over TPA-LSTM suggests that the model is able to learn structural features.

7.2 Effect of Currency Pairs on Overall Performance

The effect of increasing the number of currency pairs on overall performance is also tested. For this experiment, currency pairs are arranged in decreasing order depending on the degree of the node of the currency pair in the graph. The TPA-LSTM and GAT TPA-LSTM models are then tested on an increasing number of currency pairs depending on this order. The performance of the models when different numbers of currency pairs are considered are shown in Table 7.2

# Forex Pairs	TPALSTM Test RAE	TPALSTM Test RSE	TPALSTM Test Correlation	GAT TPALSTM Test RAE	GAT TPALSTM Test RSE	GAT TPALSTM Test CORR
2	**0.01530**	0.18879	**0.94732**	0.01541	0.18910	0.94730
3	0.01500	0.16827	0.95778	**0.01360**	**0.15421**	**0.96175**
4	0.01462	0.13446	0.95472	**0.01342**	**0.12452**	**0.96175**
5	0.01587	0.16139	0.95871	**0.01431**	**0.14733**	**0.96547**
6	0.01534	0.12870	0.96372	**0.01421**	**0.12107**	**0.96795**
7	0.01489	0.12769	0.95987	**0.01385**	**0.11871**	**0.96465**
8	0.01453	0.11967	0.96299	**0.01376**	**0.11421**	**0.96532**
9	0.01581	0.12222	0.95699	**0.01463**	**0.11256**	**0.96523**
10	0.01527	0.12008	0.95258	**0.01335**	**0.11378**	**0.96436**
11	0.01545	0.12252	0.95484	**0.01447**	**0.11151**	**0.96682**
12	0.01552	0.11259	0.95721	**0.01344**	**0.10269**	**0.96520**
13	0.01557	**0.09854**	0.95942	**0.01474**	0.10191	**0.96630**
14	0.01616	0.10559	0.95220	**0.01468**	**0.09830**	**0.96004**
15	0.01549	0.10343	0.95822	**0.01483**	**0.09750**	**0.96086**
16	0.01589	0.10843	0.96001	**0.01504**	**0.10168**	**0.96364**
17	0.01550	0.10540	0.96261	**0.01471**	**0.10004**	**0.96380**
18	0.01634	0.11192	**0.96238**	**0.01441**	**0.10286**	0.96108
19	0.01600	0.11419	0.96173	**0.01451**	**0.10308**	**0.96276**
20	0.01556	0.10932	0.96291	**0.01465**	**0.10409**	**0.96598**

TABLE 7.2: Performance of increasing the number of Forex pairs for TPA-LSTM and GAT TPA-LSTM

Graphs showing the direct comparisons between RAE, RSE and CORR for TPA-LSTM and GAT TPA-LSTM are shown in the Figures below.

FIGURE 7.1: TPA-LSTM vs GAT TPA-LSTM RAE

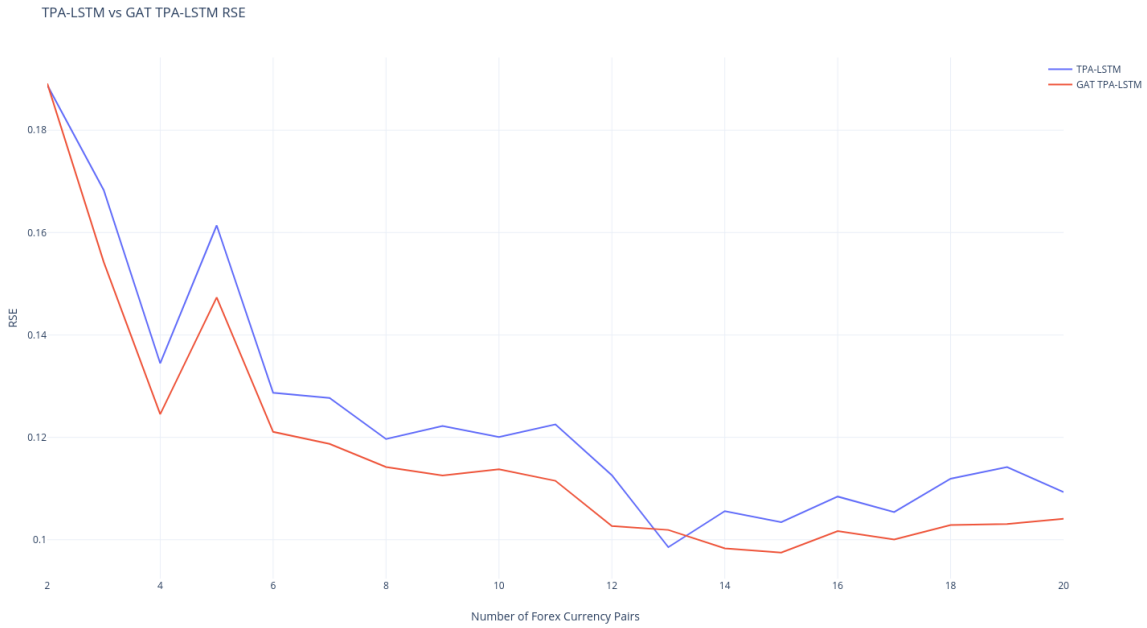

FIGURE 7.2: TPA-LSTM vs GAT TPA-LSTM RSE

RAE and CORR tend to not change much when the more Forex currency pairs are considered. However, there is a significant improvement in RSE when more Forex currency pairs are

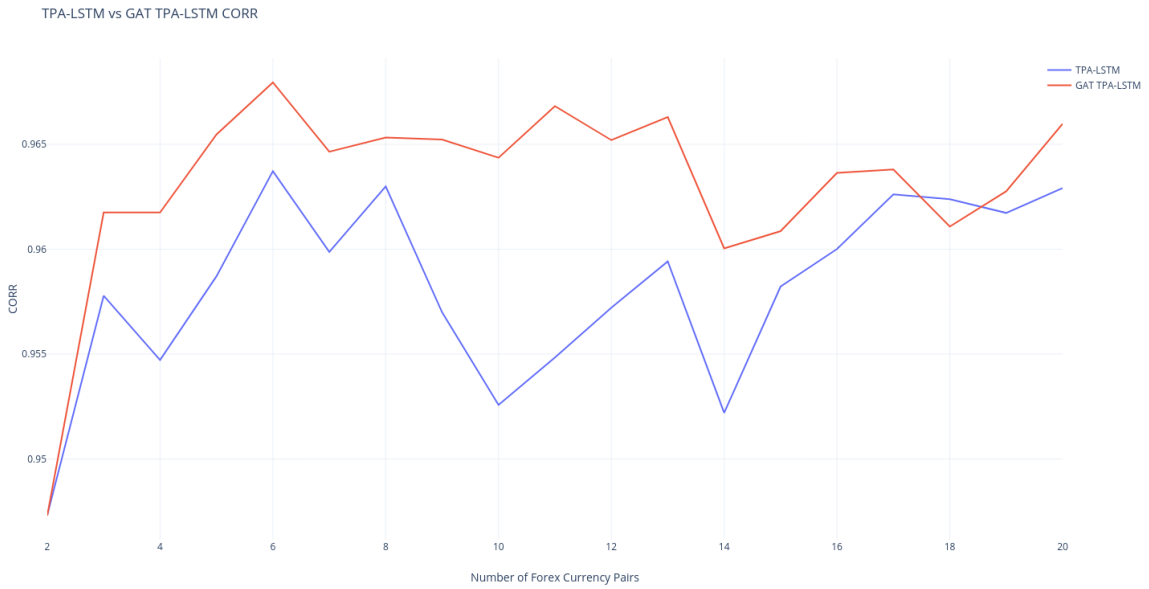

FIGURE 7.3: TPA-LSTM vs GAT TPA-LSTM CORR

considered. GAT TPA-LSTM also tends to have the better scores for all the metrics RAE, RSE and CORR than TPA-LSTM for almost all the currency pairs.

One interesting observation to note is that GAT TPA-LSTM has a similar performance to TPA-LSTM when only 2 Forex currency pairs considered. This is to be expected because it is a fully connected graph. When GAT TPA-LSTM is applied on a fully connected graph, each node in the graph layer is computing the same operation as a normal TPA-LSTM. For this scenario, there is no benefit in using GAT TPA-LSTM over TPA-LSTM.

CHAPTER 8

Discussion

There are two main conclusions that can be obtained from the results of this project:

Firstly, both GAT TPA-LSTM and TPA-LSTM show a significant improvement in overall performance as the number of Forex currency pairs considered are increased. This shows there are important inter dependencies between Forex currency pairs that are useful for prediction. This has the implication that Forex traders trying to accurately predict Forex currency pair prices should always be focusing on the MTS problem of analysing multiple Forex currency pairs rather than treating Forex currency pairs as independent from one another.

Secondly, there is a significant improvement in performance when GAT TPA-LSTM are used compared to the baseline models. This shows that the graph structured nature of the relationships between Forex currency pairs can be used to obtain more accurate predictions. The implication of this is that both the MTS nature and the graph structured nature of Forex data should be considered when trying to predict Forex.

The current project has been more focused on proof of concept rather than the practicality of applying the model for real Forex Trading. There are ways to build a more practical model for Forex Trading. Firstly, the duration of the time period of the Forex data can be changed from 1 day time period to a shorter time period (e.g. 15 minutes). This would make the model more reactive to the market and be more useful for traders. Secondly, Bayesian Optimisation can be used to optimise hyperparameters for the GAT TPA-LSTM model. The reason why both of these were not applied in the current project is because the current model on 20 Forex Currency pairs already takes 12 hours to train. Increasing the complexity of the model would increase the training duration and it could take up to a month of hyperparameter testing using

Bayesian optimisation when running on a single instance to properly tune the hyperparameters (and the costs associated with it).

More time should have been allocated to back testing the model. The main problem was not focusing on the background knowledge of how trading strategies and back testing works, followed by technical isseus the back testing code not working on Ubuntu operating systems and the GAT TPA-LSTM being too large that I could only run it on AWS Sagemaker.

Overall, I believe this project is a success with the contribution of a novel deep learning architecture GAT TPA-LSTM that performs better than state of the art models when predicting Forex data.

CHAPTER 9

Limitations and Future Work

9.1 Limitations

9.1.1 Limitation compared to TPA-LSTM

One of the main downsides of GAT TPA-LSTM is that it scales worse than TPA-LSTM with respect to the number of variables in the MTS forecasting problem. GAT TPA-LSTM has properties similar to a standard GAT where the number of parameters scales linearly with respect to the number of nodes in the graph (number of variables in MTS problem) and the number of edges in the graph. TPA-LSTM only scales linearly with the number of variables in the MTS problem. While TPA-LSTM can perform well on datasets with hundreds of variables, it would not be feasible to apply GAT TPA-LSTM on those datasets if there are too many connections between variables.

9.1.2 Limitation compared to GAT

Another downside of GAT TPA-LSTM is that it is no longer as storage efficient as GAT. It is possible to implement a GAT layer by only using sparse matrix operations and requires no more than $O(V + E)$ entries to be stored. However the TPA-LSTM component can not be implemented using only sparse matrix operations and takes up more space to be stored. GAT TPA-LSTM also greatly increases the number of parameters trainable parameters in a single graph layer. Both of these downsides reduce the number of feasible graphs that GAT TPA-LSTM can be applied on.

9.2 Future Work

9.2.1 Alternate graph representations

An alternative method to represent Forex currency pairs as a graph is to create a node to represent each currency as well as each currency pair. Nodes representing currency pairs are only connected to the two currencies in the currency pair. This is probably the more intuitive method to represent currency pairs as a graph. An example using the same currency pairs AUD/NZD, AUD/USD, NZD/USD and JPY/USD are shown in Figure 9.1.

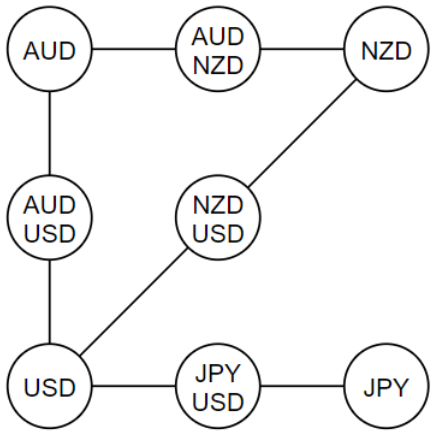

FIGURE 9.1: Alternate Currency Pair Graph

The main advantage of using this representation is that there is more detail on the structural information between currency pairs and is probably going to yield better structural information when data is propagated through it. There are 2 main disadvantages: The first disadvantage is that it is a larger graph, increasing the number of nodes and edges, resulting in a larger GAT TPA-LSTM model, The second disadvantage is that there are empty inputs for the nodes representing currencies so standard forward propagation can no longer be used.

One possible solution to propagate Forex data through this graph structure is to rely on the unique property of this graph where every currency node is only connected to currency pair nodes and every currency pair node is only connected to currency nodes. We can use 2 GAT TPA-LSTM sub layers instead of a standard GAT TPA-LSTM layer to propagate the data.

The first sublayer can be used to propagate data from the currency pair nodes to the currency nodes. The second sublayer would be used to propagate data from the currency nodes back to the currency pair nodes.

This alternative graph representation method would likely work better since the graph structure is more representative of the actual relationships between currency pairs and it should extract more useful structural information. The downside is that it is a significantly more complex model and would greatly increase the number of parameters compared to the graph representation used in the project.

9.2.2 Application to other datasets

I believe GAT TPA-LSTM should work on any dataset consisting of a multivariate time series dataset with graph structured data. For example, it should also be applicable on normal stock prices if meaningful graph relations can be constructed between stocks like what is done in for HATS, a deep learning model that uses LSTM feature extractor and GAT to predict stock prices (Kima et al. 2019).

Bibliography

Bahdanau, Dzmitry, KyungHyun Cho and Yoshua Bengio (2015). 'Neural Machine Translation by Jointly Learning to Align and Translate'.

Bishop, Christopher (2006). 'Pattern Recognition and Machine Learning'.

Bruna, J. et al. (2014). 'Spectral networks and locally connected networks on graphs'.

Hochreiter, Sepp and Jurgen Schmidhuber (1997). 'LONG SHORT-TERM MEMORY'.

Kim, Tae Wan and Matloob Khushi (2020). 'Portfolio Optimization with 2D Relative-Attentional Gated Transformer'.

Kima, Raehyun et al. (2019). 'HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction'.

Lai, Guokun et al. (2017). 'Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks'.

Luong, Minh-Thang, Hieu Pham and Christopher D. Manning (2015). 'Effective Approaches to Attention-based Neural Machine Translation'.

Meng, Terry Lingze and Matloob Khushi (2019). 'Reinforcement Learning in Financial Markets'.

Rosenblatt, Frank (1958). 'The Perceptron: A Probabilistic Model for Information Storage and Organisation in the Brain'.

Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams (1986). 'Learning representations by back-propagating errors'.

Rumelhart, David E, G E Hinton and Ronald J Williams (1985). 'Learning internal representations by error propagation'.

Scarselli, F. et al. (2009). 'The graph neural network model'.

Shih, Shun-Yao, Fan-Keng Sun and Hung-yi Lee (2018). 'Temporal Pattern Attention for Multivariate Time Series Forecasting'.

Vaswani, Ashish et al. (2017). ‘Attention Is All You Need’.

Velickovic, Petar et al. (2018). ‘GRAPH ATTENTION NETWORKS’.

Zeng, Zhiwen and Matloob Khushi (2020). ‘Wavelet Denoising and Attention-based RNN-ARIMA Model to Predict Forex Price’.

Zhang, Zezheng and Matloob Khushi (2020). ‘GA-MSSR: Genetic Algorithm Maximizing Sharpe and Sterling Ratio Method for RoboTrading’.

Zhao, Yiqi and Matloob Khushi (2020). ‘Wavelet Denoised-ResNet CNN and LightGBM Method to Predict Forex Rate of Change’.

Zheng, Heliang et al. (2017). ‘Learning Multi-Attention Convolutional Neural Network for Fine-Grained Image Recognition’.

Zhou, Jie et al. (2018). ‘Graph Neural Networks: A Review of Methods and Applications’.