Abstract

There is a disparity within the financial sector, marked by a population of financial organization’s pushing ahead and taking advantage of growing data centric ideology, new methodologies, and technology centric intelligence such as neural networks, machine learning and artificial intelligence. Deep learning traditionally known as neural networks has been a breakout area for a large number of sectors including that of the financial sector, with the advantages in using more bespoke and data driven approaches to financial modelling. The purpose of this paper is to test performance of selected deep learning and neural network architectures for forecasting the prices and movement of selected assets consisting of foreign exchange, derivatives, and cryptocurrency. The study aims for forecasting prices using different types of neural network architectures given the input of a variety of data sources. The testing results provides a more practical playbook for practitioners within the financial sector and show the importance of the applications of the growing landscape of artificial intelligence, deep machine learning, and FinTech.

Keywords: Cryptocurrencies; Bitcoin; Ethereum; Litecoin; Pricing; Machine Learning; Deep Learning; Artificial Neural Networks, Recurrent Neural Networks, Convolutional Neural Networks, Derivatives, Foreign Exchange, Algorithmic Trading.
Introduction

The growth of data in relation to financial markets has increased in volume with ease of access to historical transaction data ranging from intervals of daily, hourly, minute, and even seconds. Data has also grown in variety, with stronger and easier access to traditional data points for modelling such as interest rates, market volatility measures, and exchange rates, but now there is greater than ever focus on using unstructured and alternative data sources such as text-based data from market news, public announcements and even social-media. The open-sourced nature of financial modelling has brought about more sophisticated, experimental, and bespoke applications for forecasting the price, risk, and overall market movement of both traditional assets, as well as alternative assets such as cryptocurrency. However, with the nature of open-sourced and global digital participation in financial technology or simply known as FinTech we are also seeing a cloud of confusion form, ranging from hurdles for adoptions due to misinterpretation, misguidance, and growing tendencies of risk aversion to adopting cutting edge techniques. This is partnered with experimental mindset of financial industry which can been defined as a balancing act between hunger for growth balanced with the need for transparent and traditional methodologies.

Deep learning traditionally known as neural networks have been circulating both in academia and industry since the early 80s, with the historical applications including everything from weather forecasting, behavioral modelling, as well as financial forecasting and pricing. Historically neural networks were both computationally and technically expensive, with the need to invest in both skilled-niche expertise to design and build solutions, combined with the cost of hardware necessary to meet the computational resources needed. The overall accuracy of the neural networks in forecasting prices of assets such as derivatives were arguably superior to the traditional approaches as that of the Black-Scholes model. However, the cost of implementation measured up to the gain in market intelligence of these solutions were on the majority not seen to be commercially viable. These historical times of neural network were also inherent with no standardization or widely acceptable frameworks in neural network design, and the need for large volumes of data needed for the neural network to be trained were not so easily accessible.

With the release and open sourcing of Google’s deep learning framework Tensorflow in 2015, the growth of deep learning models ranging from image processing with convolutional neural networks, to time-series forecasting using recurrent neural networks, as well as generating
television scripts using sequence to sequence neural networks has shown that the once viewed experimental and hard to implement solutions of neural networks, are now both easily accessible and forever improving as participants globally ranging from universities, individuals, and the private sector continue to experiment and advance the features, design, and use cases. The opposite side of the open sourcing and continually improving movement in this technology is that there has also been greater levels of confusion in how, why, and what when it comes to solutions leveraging deep learning, something that is seen in the financial sector as there is a disparity between organization’s pushing ahead and taking advantage of the data boom, new methodologies, and incorporating technology such as neural networks, machine learning, and artificial intelligence into everything from customer interactions, asset pricing, algorithmic trading; and the exploration and navigation of new markets such as cryptocurrency. The opposing side of the financial sector, being made up of organization’s who are still either resistance or confused in where to start in adopting these new approaches.

The purpose of this paper is to test performance of selected deep learning and neural network architectures in forecasting the prices and movement of selected assets consisting of foreign exchange, derivatives, and cryptocurrency. The models chosen to be included within this study are the traditional feedforward neural network, recurrent neural network with LSTM, bi-directional recurrent neural network, peep-hole recurrent neural network, convolutional neural network, and generative adversarial network. The study will also show the advantages, limitations, assumptions, and overall practical use cases for forecasting prices using these different types of neural network architectures given the input of a variety of data sources including transaction, text, market indicators, and investor behavior. Each model’s hyperparameters will also be adjusted and detailed in order to aim for maximum model performance. The data used for modelling will be broken up into testable populations, allowing for a more robust study in the advantages of limitations of each neural network architecture given different variety of data types. The groups will consist of simple transactional data which will be known as group 1, transactional and market indicators know as group 2, and transaction, market indicators, and text know as group 3, and group 4 being all the listed plus indicators of investor behavior such as Google searches, open trading accounts and digital wallets, and participation and sentiment of investor forums.
This study acknowledges that the list of tested neural network architectures is not exhaustive, as we have chosen these architectures as they are one of the more commonly written about, studied, implemented, and mentioned within the data science, finance, and technology industry sectors. The testing, results, and commentary associated with this paper and study has the hope of providing a more practical playbook and framework for practitioners within the financial sector, in order to take advantage of the growing landscape and approaches of artificial intelligence, deep learning, and FinTech.

**Past Studies**

The application of neural networks for pricing financial assets such as equities, options, and futures has been studied since the early 90’s, following the revival of research and investment into computational neural networks seen in the mid to late 80’s. Malliaris & Salcenberger 1993 used a neural network consisting of a minimal architecture of seven input nodes, four middle layer nodes or hidden nodes, and one output node with a small population of data consisting of option-price transactions from January the 1st 1990 to June the 30th 1990. Their results showed that the approach using a neural network as compared to the traditional Black-Scholes option pricing model is a valuable alternative to estimating option prices, although equally they identified some limitations such as at the time there was no formal theory or universal design for neural network architecture or topography, therefore no-one could determine the optional structure, number of nodes, layers, and also training of a neural network. Another earlier attempt for using neural networks to price options was done by Lajbcygier & Connor 1997, in which they argued that the assumptions of underlying the models such as that of Black-Scholes citing the lognormal nature of prices not reflecting the true nature of the market, and rather was done for the sake of easier mathematical tractability. Such assumptions as these resulted in lower appeal and accuracy as compared to modern approaches to pricing. Their research using an artificial neural network model concluded in showing that neural networks allow for making fewer assumptions in relation to the data, and overall market.

These earlier work’s applying neural networks within the setting of financial markets and price forecasting, changed the assumptions of how technology and finance could co-exist, as the growth of market and financial data became more readily available, in addition to the growth of more techno-driven financial systems such as improved trading terminals. However, although the use of
neural networks seemed promising from earlier work, its overall application still lay within infancy, as earlier neural networks were both complex to understand and implement, in addition to the investment required to build and maintain compared to the overall yielded performance. This could be argued as a result from the overall lack of understanding and commercial applications of neural networks within the past, and somewhat still argued here into the present; as well as the lack of open-sourcing, universalism, and standardization for approaching, understanding, and implementing a neural network within the financial sector.

Despite the shortcomings of the 90s the use of deep learning and recurrent neural networks in trading and forecasting continued to be studied, along with the applications into other industry sectors such as retail, internet search engines, entertainment, and transport. Carverhill & Cheuk 2003 designed a neural network taking into consideration the inputs of delta and vega used traditionally in option valuations and portfolio measures, to develop a neural network with a custom output of price, delta, and vega, showing that the neural network produced better price and hedging parameters for American style options, as compared to traditional models such as Cox-Ross-Rubinstein model. Yamin & Li 2004 used an artificial neural network (ANN) to predict price forecasting within electrical power markets; taking advantage of the deregulation and active trading newly adopted in that sector. The ANN by then gained more attention among the existing tools as argued by the authors, due to clearer model designs and understanding, ease of implementation, and improved performance. With the aim of the ANN to provide the modelling framework for analyzing factors that could impact electricity prices, the ANN was able to identify parameters that would fit a predefined mathematical formula based on historical data and use the resulting models to predict future electricity prices based on actual inputs. Bennel & Sutcliffe 2004 used artificial neural networks to price the FTSE 100 options, comparing the performance to the traditional Black-Scholes pricing model. It was argued again in this paper that the presence of biases in the data will have a greater adverse effect on the Black-Scholes prices than on those produced by ANNs because ANNs can learn to allow for such biases, while the Black-Scholes model cannot. This resulted in increased performance and overall usage of the ANN in real world situations. The study compared the performance of the two models using the quality of input data that is likely to be available in a trading situation. A shortcoming of the paper as compared to modern approaches in relation to data volumes for deep learning is that the authors argue that more data does not necessarily lead to better results. Training the network on one year of data appeared
to represent the phenomena well, while a longer sample period often had a detrimental effect on network performance. A majority of modern applications for deep learning would argue that the more data the better, as the deep learning model either a sequence-based model or otherwise aims to learn the inherent structure and function of the data in relation to a given output. Reduced data populations may run the risk of not representing the true nature of all given situations, examples, and time-periods associated with the real-world either within stock-prices for forecasting future prices, or images of a cat and dog in relation to classification. The results of Bennel & Sutcliffe 2004 study show that, for out-of-the-money options, the ANN is clearly superior to Black-Scholes apart from options that are in-the-money whereas the performance of Black-Scholes is superior. Overall the study concluded that the ANN approach is generally superior to Black-Scholes in pricing European style FTSE 100 call options. Andreou & Charalambous 2006 also study pricing European Options with robust artificial neural networks, and saw that regarding out-of-sample pricing, their ANN hybrid model which they developed to combine that of a modified Black-Scholes, outperform a standard ANN. The results from the Andreou & Charalambous 2006 show that a standard or one-size fits all approach may not be the case when it comes to both neural network architectures as well as their application to different asset classes.

Mantri et al 2010 used artificial neural networks to measure stock market volatility comparing the results to a GARCH model which is an autoregressive model focusing on heteroskedasticity within the data distribution. The study showed that the volatility under multiple inputs and single output should be considered in the purpose of option valuation, portfolio construction and risk hedging as it is calculated by considering, high, low, opening & closing index data, however overall, they concluded that comparing the ANN model to a GARCH model showed no difference in terms of overall volatility estimates. Ding et al 2015 propose a predictive model using deep learning to capture the influence of news events to stock prices and stock volatility. They used a convolutional neural network (CNN) to perform semantic composition over the input text sequence, and a pooling layer to extract the most representative global features within the text. Using a feedforward neural network, they associate the global features extracted from the text with stock trends through a shared hidden layer and an output layer. The results of the study showed that a deep convolutional neural network can capture longer-term influence of news event than compared to a standard feedforward neural network, in relation to price and news events.
The extension of using newer and also experimental types of neural networks within the financial sector was also studied by Dash & Dash 2016 who propose a novel decision support system using a computational efficient functional link artificial neural network and a set of trading rules based on technical analysis with the aim of generating trading decisions, treating the decision output as a classification task, yielding favorable results in trading simulations. Sharang & Rao 2015 study proposed a change in data inputs for option pricing and the use of neural networks, in which they designed a prediction using the derivatives instead of the underlying entities themselves, arguing that this leads to a more feasible problem, since derivatives are less volatile and hence have clearer patterns. Fisher & Krauss 2017 approach the deep learning and neural network application to financial market predictions using a sequence based neural network architecture taking advantage of recurrent neural networks and the advent of the long-short term memory architecture commonly known as the LSTM. They applied the LSTM framework to S&P 500 stocks from the period of 1992 to 2015. Their results indicate that the returns forecasted from the LSTM consisted mean returns of 0.46 percent per day compared to other machine learning methods such as the random forest, deep neural network, and the logistic regressor. The model itself took advantage of the ease of use and power of TensorFlow with the Keras framework and Python programming language implementation, all of which are free to source, use, and have a large population of training and coding resources online. When implemented into a trading situation and a simple rules-based trading strategy the LSTM network achieved a daily return of 0.23 percent prior to transactional costs.

The open sourcing of Google’s framework for deep learning TensorFlow, created a booming effect for new computational packages, frameworks, and overall experimentation and studies through open-sourcing and easier access and understanding from a global digital participation and experimentation. We can argue now that deep learning, neural networks and their applications have moved beyond the restrictions in understanding, implementation, and performance seen in the 90s. Feuerriegel & Ferhrer 2018 applied deep learning to the task of predicting stock market movement, using a non-traditional data source which as the being the release of a financial disclosure. The aim of the study was to train the deep learning model to learn appropriate features from the underlying textual corpus of the financial disclosure and thus surpass other state-of-the-art classifiers. The use of alternative data sources such as this enables higher predictive
performance and increased opportunities investors, financial governing bodies, and automated traders.

The extensive studies of the application of neural networks for traditional financial assets and instruments, can be extended into non-traditional and somewhat misunderstood alternative financial assets as cryptocurrency, however this has not widely been studied. This may be a result of cryptocurrency not hitting the radar of universities, financial bodies, and traditional investors until the large surge in price seen in 2017 through to late 2018. Although only being a short period of time of less than two-years the growth in both interest within assets such as Bitcoin, as well as the explosion of the alt-coin and initial coin offering (ICO) universe. This time was also characteristic of increase interest of cryptocurrency within both industry and academic circles, as well as the number of cryptocurrency experts globally grew. Many studies aimed to apply traditional methods of market and asset measures to this newly desirable digital investment asset. Dyhrberg 2016 study on applying a GARCH volatility analysis and comparing Bitcoin to have similar volatility and portfolio characteristics as gold argued that Bitcoin is a hedging tool for the risk adverse investor, in anticipation of bad news in the market. However, late 2018 and early 2019 saw the growth of broken promises as the market went from the peaks of price of Bitcoin $17k USD down to $3k USD. Market acclaimed cryptocurrency pricing approaches such as using Metcalfe’s law fell aside as previously pushed by bullish investors and market commentators, as the increased amount of price-fixing associated with cryptocurrency exchanges came to media attention, along with fraud, and mislead promises inherent in the alt-coin and ICO universe. The unforeseen and manic scramble for both entry and exit of this market in such a short period of time may be attributed to misunderstanding, greed, and overall ignorance of what the cryptocurrency market really is, in addition to the application of traditional mindsets and models to a continually evolving digital and decentralized asset. However, this time also saw a new mindset in applying deep learning and its computational advantages to tackle forecasting and understanding the unknown world of cryptocurrency. Rebane et al 2018 applied deep learning and specifically the application of a multi-layer sequence-to-sequence (seq2seq) neural network for pricing cryptocurrencies focusing on Bitcoin pricing, comparing it to a traditional but still popular model being the autoregressive integrated moving average model (ARIMA). The use of social and text-based data is identified within their research as being linked to the inherent fluctuations in price of cryptocurrencies such as Bitcoin, Ripple, and Litecoin as being a result to investor
reactions to market sentiment and trader behavior. The seq2seq model outperformed the ARMIA model, although fell short in trying to model the December 2017 Bitcoin price crash, as a crash such as this was not considered within the past training data population. The result of the Rebane et al 2018 study show that there is both the potential of use of deep learning within cryptocurrency price forecasting and trading, as well as there needs to be a focus on data in terms of sources, as well as domain understanding.

**Proposed Methodology and Data**

The approach for this comparative study will employee recent and most widely considered approaches to deep learning architectures, although as mentioned previously this may not encompass all past and current architectures, it is not proposed to be an exhaustive list, but rather a representative sample gained from approaches within both academia, as well as the approaches most widely mentioned within financial industry.

The deep learning architectures will be separated into sequence and non-sequence-based approaches, which is defined as the treatment of the input data as a sequence of linked events such as a timeline, with the non-sequence-based architecture treating the data input as fixed or within a window within a defined space. Under each of these categories separate models will be tested, in which each model will be unique through the addition of features within the given architecture such as the ability of memory of past events within a sequence; the ability to peep-forwards into future timing of events within a sequence.

The study will also test some traditional growing experiment areas including the traditional feed forward neural network, as well as a convolutional neural network which has been popularized with application to image processing but has seen growing interest in classification of pricing movements within financial data. Each models hyperparameters will also be adjusted and detailed in order to aim for maximum model performance.
**Sequenced based neural networks**

The sequence based neural networks that will be used in this study are primarily based on variations, addition of parameters and features from the design of the vanilla recurrent neural network; the most widely used neural network architecture for treating data as a sequence. The recurrent neural network differs from traditional feedforward neural networks, by taking into consideration the ordering of the data and how this influences future data values such as stock pricing, asset pricing, and even a sequence of words in a sentence. Through treating data as a sequence of events, we can factor in previously learned and contribution of variable inputs in order to construct a more informative and adaptive model prediction or forecast. Using the previous

![Simplified recurrent neural network representation for time-series](image)

**Figure 1.0 – Simplified recurrent neural network representation for time-series**

sequence of data, and its influencing weights on the target variable, is a form of conditional memory which is an advantage seen in the use of recurrent neural networks.

In order to understand the basics of recurrent neural networks and their application to time-series or sequence data, we can use the simplified representation as shown in figure 1.0. In this figure, you can see that the data inputs as shown by X1 to Xn. Whilst for this simplified approach there is only a single input, a majority of models will have multiple inputs for all the different variables
along the time series, such as when applied to forecasting asset prices a model may leverage volume of trades, price, volume of currency, open, close, and weighted price in the form of a matrix of values.

This matrix of inputs will run through the recurrent network and pass into the hidden state, which has a recurrent flow or feedback of the data to learn and adjust the weights of each input into the model. The hidden state is also connected to not only the output or model prediction represented by Y1 to Yn along a time series, it is also connected to the next hidden state of the network, thereby allowing for memory in the network and feeding of previously learned states in the form of weights and inputs to the next state along with the new input data.

These weights are used in terms of how much influence each variable input has to the target output, with the weights being adjusted through learning in the network conducted through backpropagation of the cost/error, measured through the application of such cost functions as root mean squared error (RMSE), represented as:

\[
RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (y_t^l - \hat{y}_t^l)^2}
\]

RMSE measures the difference between the actual target and the model output, which, through backpropagation or passing these values back through the network, each node/cell can adjust the weights of inputs using optimisation and gradient descent.

The use of gradient descent along with optimisation approaches and algorithms such as ADAM, ADAGRAD, or Momentum, can be viewed as adjusting the steps along a descending gradient in order to find a local minima for the purpose of updating the weights within a composite function such that, the errors in the function are minimised and the values for the weights are optimised where the gradient is near or equal to zero.

To further drill down and understand each step taken in the recurrent network, we can see a simplified representation of an unpacked node/cell in the network as shown in figure 2.0 below. We can see the weight representations from the input to hidden state, to the recurrent feedback inside the hidden state, and then to the update of weights to the output. The hidden state of the neural network is represented with the following calculation:
\[ h_t = f(W_{hh}h_{t-1} + W_{xh}x_t) \]

In this formula, \( f \) is the activation function for the hidden state such as a sigmoid, tangent line function, etc. and \( w \) is the weight assigned to the inputs and outputs from the hidden state to the output of the cell. \( Y \) is the output of the cell in response to both the input/s of \( x \) along a time series, in addition to the previous hidden state connected within the network as seen in figure 10.0. The output of a single cell in a recurrent neural network can be represented as:

\[ y_t = W_{hy}h_t \]

where \( h \) is the hidden state at time \( t \) and \( w \) is the weight attributed to the hidden state.

![Simplified unpacked node of recurrent neural network](image)

**Figure 2.0 – Simplified unpacked node of recurrent neural network**

The deployment of a solution such as deep learning and the use of a recurrent neural network allows for an adaptive and learning based solution which takes into consideration the current and previous time-steps in the data and how they influence such outcomes as price movements, investor behaviour, as well as shaping volatility.

Construction the targeted contributing variables along the time-series, will help the recurrent neural network learn the weighted influence each variable has on the price throughout time, based on the interaction each variable has in relation to the actual price as seen in the data. The network will then commence to iterate through a training dataset and through a process of measuring the cost or error associated with the actual and predicted outcomes of the model. It will adjust the
weights of each, allowing for a more robust and accurate prediction of price. Once the number of specified training iterations have taken place, the model will use a testing dataset in order to see how accurately it can predict and forecast future time-periods and data samples.

**Recurrent Neural Network with LSTM**

Long Short Term Memory networks are a variant on the recurrent neural network as introduced by Hochreiter & Schmidhber in 1997, as a solution to remedy the vanishing and exploding gradient problem caused by the use of back-propagation through time as part of the method used by recurrent neural networks to adjust the weights of the input through gradient descent with the aim of reducing the overall error measure of the observed and modelled result. The solution brought about by introducing the LSTM as described by Hochreiter & Schmidhber in 1997 is by ‘enforcing a constant error flow through the internal states’ (Hochreiter & Schmidhber in 1997). This can be interpreted as maintaining a memory vector in which through each timestep the LSTM can adjust, use, or reset the memory vector using a system of gates and thresholds within the LSTM cell logic. The gates can be represented as an element wise operations within a matrix, with each gate with either being a sigmoid function or a tangent function, serving as a binary control on opening or closing the logic gate. The gates allow for ‘gradients on the memory cells to flow backwards through time uninterrupted for long time periods, or until the flow is disrupted through multiplicative interaction of an active forget gate’ (Karpathy et al 2016). The LSTM also enables the idea of memory into the neural network, in which the contribute of each input through time can be measured and remembered, allowing for modelling specific contribution and structure of individual data points on the overall time-series or sequence of data and the overall outcome.

The long short term memory cell (LSTM) replaces the hidden component of a recurrent neural network, allowing for the previous mentioned advantage of conditional memory, in which the network can learn what to remember, what to pass on, and what to forget – thus allowing for an increased level of insight and accuracy within the network, especially when dealing with complex data and issues such as forecasting prices based on sequences of input variables.

A simplified representation of a LSTM cell is seen in figure 3.0 below, in which the cell shows the different movements of data throughout the gates of the cell. The cell is defined by a number of activation functions associated with control the flow of data and weights throughout the cell, which contribute to the different cell operations and cell states, and the x and + are point wise or element
wise operations on data vectors and matrixes within the cell. The Cell state is represented by \( C \) in which \( C_{t-1} \) is the previous cell state from the previous node passing into the new cell contained within the new node of the recurrent neural network, and \( C_t \) is the new cell state from this cell and node. The hidden state, as represented by \( h \), is now calculated using the cell state of the LSTM, and passed onto the output \( Y \) of the network.

The sigma activation function, or a sigmoid activation function, is associated with transforming values to a range of 0 to 1, and controls the forget gate. This is a combination of the input \( x_t \) and \( H_{t-1} \) which is the previous hidden state input of the network, which then, through element wise multiplication with the previous cell state \( C_{t-1} \), will transform the cell state, where values close to 0 will make the cell forget the previous information and weights going forward. Whereas, values close to 1 will pass through the cell unchanged onto the next cell state in the next node. This is helpful as the network can forget information that causes incorrect predictions for the model outcome. Further, information from previous states for certain variable weights which are helpful in prediction can flow through the cell freely and unchanged.

The next gate in the LSTM cell uses the tangent activation function which will transform inputs into the range -1 to 1, which is fed by information from \( x_t \) and \( H_{t-1} \) the input and previous hidden state respectively. This tanh is gated by another sigmoid activation function and is known as the update gate or update state of the LSTM cell and thereby updating the \( C_t \) or new cell state that is passed along the time step. This is done when new weights for variables are updated in the network and thereby the cell state will change.

The last gate is the cell state to hidden output which is shown here by a sigmoid activation function connected to the previous hidden state \( H_{t-1} \) and the input at time step \( x_t \) which feeds into the new hidden state \( h_t \). The use of a sigmoid function will help the cell and network know what information to forget and what information to pass on to the new cell state and hidden state.
Using solutions such as the RNN and LSTM allows for numerous types of data and inputs to be used in predicting the outcomes such as pricing, which traditional pricing models cannot account for. These advancements into the realm of smarter models with memory, which can continually mine, clean, and process data in the form of iterative learning and communicate these findings while integrating into existing processes, business intelligent tools, as well as connecting to other models to help develop better investment and derivative pricing strategies.

**Recurrent Neural Network with GRU**

Gated recurrent unit or know as a GRU is a variant of the LSTM in leveraging the use of an update gate and reset gate, in order to solve the vanishing and exploding gradient problem encountered during backpropagation in the training of recurrent neural networks. Like the LSTM the GRU can also retain information as a memory. However, the GRU differs from the LSTM in terms of training times being reduced and less volumes of data needed for the network to generalize, this is a result of fewer parameters being used as compared to the LSTM.
**Bi-directional recurrent neural network**

The bi-directional recurrent neural network allows for the network to learn by both looking ahead and behind in the sequence. Using an LSTM cell framework, the network can go forward in time which aids in learning from the previous or historical sequence of data inputs and their contribution to the future. As well as being able to look back in time and learn how the future contributed to previous events.

**Peep-hole recurrent neural network**

The peep-hole variant of the RNN LSTM network allows for the LSTM cell to peak or look at the cell state in order to add the element of timing into the cell state changes.

**Non-sequenced based neural networks**

**Feedforward Neural Network**

The feedforward network is a neural network made up of a number multi-layer perceptrons, connected in a feedforward architecture and is seen as the vanilla architecture of neural networks. The neural network takes in the input data which is feed into the hidden layers of the network, which determine the pattern existing in the data and associated influencing weights that contributes to the output. The network will learn via backpropagation with the goal of finding a global minimum using gradient decent to minimize the error between an observed outcome and the modelled output.

**Convolutional neural network**

Convolution neural networks have been made popular due to their strength in image processing and classification. The convolution neural network architecture differs from the traditional neural network as the layers are not fully connected, in that only the receptive field neurons are connected allowing for the network to focus on breaking down the input into lower level features throughout the learning process. These features obtained from the data are pooled into the successive layers of the network in which the features can be reconstructed in order to form a prediction.
Generative Adversarial Network

Generative adversarial networks (GANs) are deep neural net architectures comprised of two neural nets which engage in competitive game. One neural network called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity in which it compares if that data has been built from features of the training set. THE GAN has gained popularity in images and video, however there are some experimental approaches gaining popularity for using GANs in stock market prediction and automated trading solutions.

Approach

Combined with testing the listed neural network architectures, the study will also employ a variety of data types including transaction, text, market indicators, as well as investor behavioral data. The approach will consist of testing the performance metrics, accuracy of forecast, and requirements of each neural network using the population of data, in addition to combinations of data types which will be broken down into sub-populations. The sub-populations of the data will consist of transactional based data (Group 1), transaction and market, data (Group 2) transactional, transaction, market, and text (Group 3), and transaction, market, text, and investor behavior (Group 4). This will have the aim of not only measuring which architectures do better given different types of data inputs, but also what data inputs contribute to the overall performance of the forecast and predictive outcome of the neural network.

Each asset type will be characterized by different data sources; however, the overall classes of data will be inherently defined as either being transactional examples being price, daily and hourly trade volumes, and market capital. Market news and text-based data such as announcements and headlines from news and financial news sources. Market indicators such as interest rates, debt volumes, and USD currency price. Investor behavior examples being number of google searches, opened digital wallets and accounts, number of posts on forums, and forum sentiment.

Data will undergo the required transformation and normalization in order to meet the input requires of each neural network architecture, including normalization, embedding, vectorization, and fitting into a sequence if using a sequence-based architecture.
The neural networks will incorporate the ADAM algorithm optimizer which is named the Adaptive Moment Estimation Algorithm. The ADAM optimizer does element-wise matrix operations for adjusting gradient descent in order to find the local minima through the update of two separate parameters and bias corrections using the derivative of the composite function of the network. The ADAM algorithm is one of the most commonly used and high performing algorithms for neural networks and is proven to be very effective for a number of problems addressed in a variety of neural network architectures.

**Cryptocurrency**

The three cryptocurrencies under our pricing consideration are Bitcoin, Ethereum, and Litecoin. These cryptocurrencies have received an exponential increase in their value within a short period of time during 2017, however have reduced drastically in value during the end of 2018 and start of 2019. The data will be mined from the following exchanges.

- Bitcoin – Kraken Exchange using their API prices for date range 07/01/2014 to 14/01/2019
- Ethereum – Poloniex Exchange using their API prices for data range 08/08/2015 to 14/01/2019
- Litecoin – Poloniex Exchange using their API prices for data range 01/01/2015 to 14/01/2019

The Ethereum and Litecoin data will only mined back to 2015, as data previous to this doesn’t show much trade activity, considering that Ethereum was only launch in August 2015, and Litecoin was only circulated and purchased in exchanges widespread in 2015.

The data will be cleaned in which gaps in the data were replaced using a propagation of the values forward and backward. Missing values are very common in cryptocurrency data as cryptocurrency exchanges commonly encounter both technical issues in recording data for that day, or the exchange not trading for the given day due to compliance or various other reasons. A timestamp for the transactions will be applied using the date range for the data mine. The dates will be validated using the price to ensure that any UNIX timestamps inherent in Blockchain data ledgers to date conversions did not result in any mistakes.

The data will be then split into a 70% training set in which the recurrent neural network would train to adjust and optimise the input weights of the variables in order to reduce the cost/error in
difference between the actual and predicted price for the cryptocurrency; with the remaining 30% being used as a validation and testing set for the model’s predictions and forecasting. The training of the neural network will leverage mini-batching or stochastic gradient descent, where, in training a recurrent neural network, we took samples of the data sequence and train and learn and then continue for every iteration throughout the training set in which batches data will be sent through a complete epoch within the neural network, where an epoch is one complete run of the training data through the network as well as backpropagation for weight adjustment.

**Derivatives**

The derivative being used for this study will be the Bermudan Swaptions, in which the study aims to find the both the optimal neural network model and parameters, in addition to the relationships and contributing weights inherent in the data that contribute to the short-term interest rate curve that is necessary to price the Bermudan Swaption.

The outcome will address the question of how the short-term curve forecasts can be improved through deep learning approaches compared to models such as Vasicek, Cox-Ingersoll-Ross (CIR), Brennan-Schwartz and Black-Karasinski, Ho–Lee Model, Hull–White Model, in which these models define the instantaneous interest rate (short rate) using different form of stochastic differential equations; in addition to also comparing which neural network architecture and data sources will be optimal for pricing the Bermudan Swaptions for establishing the short-term rate curve.

The assumption behind the use of ML techniques is that by following data-driven optimization, is expected to have advantages in terms of reduced complexity, while also producing outcomes which can adapt to changing market environments. This is through the belief that neural networks are particularly well-suited to the type of data found in financial markets, such as high dimensional data sets of noisy data with apparent non-linear relationship between the variables.

Calibration Methodology we are using in the derivative price forecasting will commence with find the bootstrapped curves, as the discount and forward curves calculated at time to the commencement date of the Bermudan Swaption contract. Then our Calibration objective will be to find the short interest model (such as a Hull-White one factor model, a Linear Gaussian two-factor model, a LIBOR Market Model and Vasicek’s one factor model) optimal parameters, the
ones that will give us the best curves Discount predicted and Forward predicted, where best means the closest approximation to the bootstrapped ones. Then we will have to define the error metric. Given the predicted curves, we can derive the variables of the Bermudan Swaption prices thus the prices itself.

**Foreign Exchange**

Foreign exchange currencies which will be considered within the modelling are USD/EURO, AUD/USD, USD/RMB, USD/YEN, and USD/BAHT.

**Performance Measures**

The performance metrics and criteria for comparing each neural network will consist of classification accuracy measures, error rate, precision and recall, and forecast accuracy. These will be represented for each architecture and data population.

**References**


