Portfolio Optimization using Deep Learning for Momentum Trading

Sourav Patel

Under the supervision of

SUVAJIT MUKHOPADHAY

Final Report

Liverpool John Moores University – Master's in Data Science

December 2021

ACKNOWLEDGEMENT

Throughout the process of writing my dissertation, starting from creating a proposal, drafting an interim report and finally completing my thesis, I have received constant help, guidance support and motivation.

Firstly, I would like to thank my thesis supervisor, Mr. Suvajit Mukhopadhay, for his insightful comments and timely feedback, which refined my thinking process, during the research process as well as while writing the thesis.

Secondly, I would like to thank Dr. Ahmed Kaki from Liverpool John Moores University for his never-ending guidance and assistance through weekly sessions. I would also like to thank the UpGrad team for presenting me with this opportunity of completing my dissertation and their assistance throughout the program.

I owe my gratitude to my family, for their support in pursuing my academic goals and providing motivation throughout the process of researching and writing my thesis. And finally, a sincere thanks to my friends for their continued support.

Thank you,

Sourav Kumar Patel

Abstract

Financial portfolio management refers to the continuous reassignment of funds into financial assets like cash, stocks, bonds, mutual funds, bank deposits and commodities. The financial portfolio is created based on the investors' risk tolerance, investment objectives, and time period. The price of each asset has an impact on the portfolio's risk/reward ratio. It's also known as portfolio asset allocation. Because the investing world is so volatile, portfolio management includes asset allocation and rebalancing. With the fast changes that occur throughout time, one sector may become less popular than another, and vice versa. As a result, only a nimble investor can benefit handsomely from the world's stock markets.

To tackle the challenge of Financial Time Series Forecasting, researchers devised a variety of Machine Learning models, and a lot of papers have been published as a result. Despite the increased interest in creating financial time series forecasting models, few review papers have focused on optimizing financial portfolios to maximize returns. As a result, the goal of this paper is to develop a method for managing our portfolio assets so that we may maximize profits depending on market momentum.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
OHLCV	Open-High-Low-Close-Volume
DFNN	Deep Feedforward Neural Network
FX	Forex
ML	Machine Learning
ANN	Artificial Neural Network
EC	Evolutionary Computation
GP	Genetic Programming
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
DMLP	Deep Multilayer Perceptron
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
AE	Autoencoder
DRL	Deep Reinforcement Learning
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
NLP	Natural Language Processing
ARIMA	Autoregressive Integrated Moving Average
SRNN	Stacked Recurrent Neural Network
DWNN	Deep and Wide Neural Network
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
EDA	Exploratory Data Analysis
VWAP	Volume Weighted Average Price
PLR	Piecewise Linear Regression
ES	Exponential Smoothening
RMSprop	Root Mean Squared Propagation

ReLU	Rectified Linear Unit
MSE	Mean Squared Error
SAE	Stacked Auto Encoder
WT	Wavelet Transform
AE	Auto Encoder
RAF	Random Forest
LOG	Logistic Regression Classifier
DFP	Deep Feature Policy
NMSE	Normalised Mean Squared Error
NL	Neuron Level
SL	Synapse Level
FNN	Feedforward Neural Network
DJIA	Dow Jones Industrial Average
NYSE	New York Stock Exchange
DWNN	Deep and Wide Neural Network
SFM	State Frequency Memory
DFT	Discrete Fourier Transform
IFT	Inverse Fourier Transform
BPTT	Back-Propagation Through Time
AR	Autoregressive
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
MAD	Mean Absolute Deviation
VAE	Variational Autoencoder
MCC	Matthews Correlation Coefficient
ROI	Return on Investment

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Portfolio management is the dynamic cycle of consistently redistributing a measure of financial assets into various diverse monetary investment products, where the objective is to boost the returns while limiting the risk as much as possible. According to the study by (B. Li & Hoi, 2013), Traditional portfolio management strategies can be categorized into four classes, “Follow the Winner”, “Follow the Loser”, “Pattern Matching”, and “Meta Learning”

“Follow the Winner”, attempts to asymptotically accomplish a same growth rate (expected log return) as that of an ideal strategy. According to the “Follow-the-Loser” strategy, wealth should be transferred from winning assets to losing ones, which seems contradictory as per common sense but according to multiple observations, it often generates remarkable results. “Pattern Matching” algorithms anticipate the subsequent market distribution based on the sampled distribution and explicitly optimizes the portfolio using the sampled distribution as per study by (Gyorfi et al., 2006). Lastly, “Meta Learning” strategy consolidates multiple methodologies of different classifications to achieve more consistent and predictable results (Das & Banerjee, 2011).

Many Deep Learning based approaches have been explored for financial stock market prediction so far. However, the majority of them attempt to forecast price changes or trends for a selected number of assets where the focus is on the said assets only rather than the momentum of the market itself. As per design, a neural network generates an output vector of asset prices for the upcoming time period using just historical raw price data (OHLCV) of the financial assets as input. The trading agent can then take action based on the prediction. This can simply be termed as a supervised learning regression problem, and it is straightforward to implement. As a result, the accuracy of these price prediction-based algorithms is largely dependent on their performance, and it turns out that capturing future market momentum is challenging.

In order to manage the portfolio efficiently to maximize returns based on the momentum of the market, it is essential to choose quality assets themselves rather than only focus on the trading strategy itself. The quality of assets may depend on the intrinsic factor of the asset itself like Current Volume, Price Change %, Market Cap, Volatility, and Nearness to 52W High etc.

along with its historical price itself. These intrinsic attributes also depend on some extrinsic factors like the sentiments regarding the asset, the sector of the asset and the market momentum itself.

The most frequently researched financial application field is financial time series forecasting, specifically asset price forecasting where the major focus is always on predicting underlying asset's next movement. This was the focus of the majority of the existing DL implementations. Despite the existence of several sub-problems like price forecasting for Stocks, Indexes, Forex, Cryptocurrency, Bonds, Commodities, Volatility and many more, the fundamental elements in all of these applications are very similar.

Based on their dependent output, which can be either price prediction or price movement prediction, the studies can be divided into two groups. Although price prediction can be termed as a fundamental regression problem, in majority of its applications, correctly recognizing the directional movement or momentum is more essential than precise price predictions. Due to this, analysts regard trend forecasting, or anticipating the directional movement of the price, to be more important field of research than precise prediction of price. In this case, trend/momentum prediction turns into a classification problem. As per certain studies, only up and down price movement (2-class problem) are taken into account but neutral movement is also taken into account in others (3-class problem).

DL models like LSTM and its variations, as well as certain hybrid models, have dominated the financial time series forecasting arena. Because LSTM by design is built to exploit the temporal features of any time series data, identifying financial time series signals is a closely examined and useful application. Some analysts, on the other hand, choose to extract relevant characteristics from time series data or modify time series data such that following financial data is fixed in time. This means that the model can be trained correctly and better test performance can be achieved on unseen data regardless of the data order. DFNN and CNNs were the primarily used models in those implementations.

1.2 Aims and Objectives

This study is aimed at building a strategy to optimise the financial asset portfolio in order to achieve maximum returns based on a chosen period. More emphasis is given on choosing the assets itself based on their intrinsic attributes along with their historical price data so that depending on the momentum of the market, we always choose quality assets with positive momentum. Most of these attributes can be calculated using the historical price data.

Various Traditional and Deep Learning based approaches will be compared for this asset selection strategy to see how the returns of the selected assets compare to fixed indexes like NIFTY50 and SENSEX.

The following points form the research objective for this study:

- Perform Feature-Engineering and come up with intrinsic attributes for the assets.
- Build portfolio using the assets chosen based on the new attributes for the chosen period.
- Build different traditional and Deep Learning approaches to predict the asset price for chosen period.
- Evaluate and compare the different approaches
- Compare the returns % achieved using our portfolio against the standard indexes

1.3 Scope of Study

This study will focus on the financial assets which belong to the Indian Stock Market only. This includes top equities i.e. individual stocks and indices like NIFTY and SENSEX. The main objective of this study is to optimize the financial portfolio given a period of time to choose quality assets for investments. These assets will be chosen based on the intrinsic attributes which will be derived based only on the base attributes of the financial assets. The base attributes of these assets include only the historical price data of the assets including volume in market.

After these financial assets have been filtered and allocated, the financial time series price prediction model will be deployed on top of them for a certain time period to evaluate performance using evaluation metrics such as MAE, MAPE, RMSE, and Return/Profit percent.

We will compare and analyse the returns using various DL models, since we have observed in prior research that DL models outperform their ML counterparts.

We will not be using any traditional ML based models like ANNs, ECs, AEs etc. for the scope of this study as we have already seen that DL models outperform their ML counterparts based on previous researches.

We will only be considering features for asset selection and model building which can be derived/extracted from the base attributes (OHLCV). Any other macroeconomic data, environmental considerations, sector information, geographical or political factors are out of the scope.

1.4 Significance of Study

Financial portfolio management is a well-studied financial application field by both scholars and investors, with the primary goal of maximizing financial returns. We've seen the use of ML models like ANNs, ECs, GP, and Agent-based models in the financial time series forecasting space. With the advent of DL strategies for financial forecasting research, the financial community received a new push recently, and a slew of new articles resulted. We found that DL models outperformed their ML counterparts in the great majority of trials. DMLP, RNN, LSTM, CNN, RBM, DBN, AE, and DRL are some of the DL models described in the literature.

In all of these studies so far we saw that the main research objective was to increase the returns for a chosen specific financial assets which were kept constant and improving the performance of the financial price forecasting algorithm. Also, the dataset which were chosen for these studies contained only the historical price data for the assets and in some cases the user sentiment data extracted from different sources. The quality of the financial asset itself was not given much significance in these studies. One more thing to note is that majority of these studies have been done on US and Chinese Stock market and a very small number of studies have used some specifically chosen financial assets from Indian Stock Market. Based on historical data we have seen that the Indian Stock market is highly volatile compared to the volatility of the US and Chinese counterparts.

In this study, we plan to give more emphasis on the quality of the financial assets itself by deriving various intrinsic attributes for the given assets which can then be used to identify and filter quality assets on top of which we can apply the financial time series forecasting models and evaluate their performance. As we have seen that the DL models were superior to their ML counterparts, we will be using various DL models to compare and evaluate the returns. While deciding on the financial assets for investments, we have to look at the stock market's momentum and reversal impacts. To our knowledge, only a small amount of research has employed machine learning to address this problem.

1.5 Structure of the Study

In this study, we start with the literature review of the currently existing Deep Learning Solutions for solving the problem of Stock market prediction. Then we will proceed with explaining our methodology. Based on the Literature Review, we will be finalizing on the models to go forward with for our use case. We will then proceed with implementing the chosen models on our chosen portfolio and evaluating them according to our chosen evaluation metrics. We will also use the models to evaluate the returns achieved in a portfolio with random stocks or standard indexes like NIFTY, SENSEX etc.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

After doing literature review extensively, we have filtered out the studies which we felt were relevant and suggested solutions comparable to the current state-of-the-art solutions. A brief overview of these studies are as follows:

2.2 Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance(Chandra & Chand, 2016)

2.2.1 Business Problem

For time series prediction issues, the combination of soft computing approaches such as evolutionary algorithms and NNs has yielded extremely promising results.
Cooperative coevolution breaks down a problem into isolated subcomponents, and collaboration usually requires fitness evaluation. Developing efficient subcomponent decomposition algorithms for various neural network topologies has been a problem. The Two decomposition approaches have been compared and evaluated for training feedforward and RNNs for chaotic time series in this study. We also use them to solve financial forecasting challenges from NASDAQ. The constraints of real-time implementation are discussed, as well as a mobile application architecture for predicting financial time series. In general, the results demonstrate that for real-world time series issues, recurrent neural networks outperform feedforward networks in terms of generalisation.

2.2.2 Research Methodology

This paper examines the effectiveness of coevolutionary techniques for training FNNs and RNNs to predict chaotic time series data. Elman recurrent networks are described in detail initially, followed by cooperative neuro-evolution and time series issues.

As illustrated in Figure 2.2.2.1, Elman RNN is made up of an input layer followed by a context layer that provides state information followed by a hidden layer and finally an output

layer. Each layer has one or more neurons that use a non-linear function of their weighted sum of inputs to transmit information from one layer to the next.

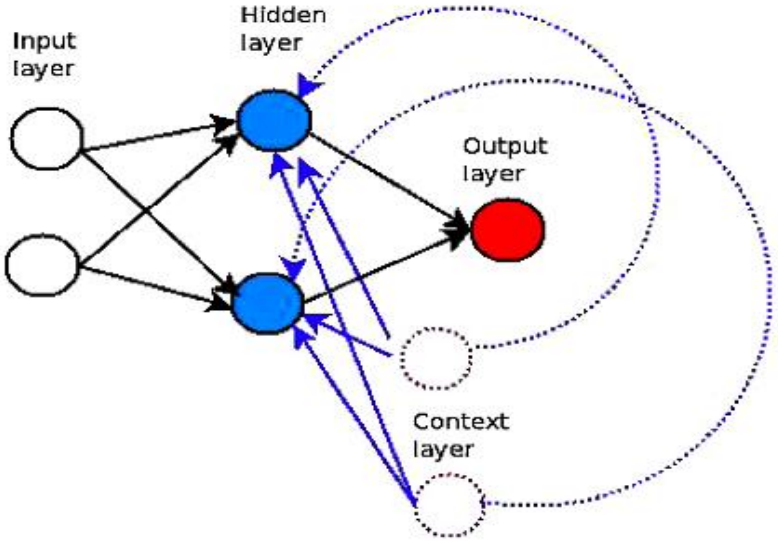


Fig. 2.2.2.1 - Elman RNN(Chandra & Chand, 2016)

Elman RNNs have been trained utilizing cooperative neuro-evolution and back-propagation over time which have proved to be extremely useful for time series prediction issues.

Cooperative coevolution utilizes issue decomposition techniques that deconstruct the network architecture into multiple small components. It also specifies how weights of subcomponents are represented that are implemented as sub-populations. NL and SL issue decomposition are the two most used problem break down approaches.

The NN is decomposed to its simplest level in SL problem decomposition, with each weight link (synapse) becoming a subcomponent. The neurons in the network serve as the decomposition's reference point in NL problem decomposition. Enforced sub-populations and neuron-based sub-populations are two examples.

A huge number of data points are collected regularly at same intervals for time series prediction. We must alter them in order to use soft computing approaches such as NNs to construct prediction models. The time series must be split/sliced down into smaller chunks that are acquired at regular intervals in order to do the fundamental transformation or reconstruction. Following is a detailed description of the procedure.

An embedded phase space Y(t)=[(x(t), x(t - T),..., x(t(D - 1)T)] can be generated from an observed time series x(t), where T represents time delay, D represents embedding dimension, t=0, 1, 2,..., N - D_T - 1 and N = original time series length. The vector series reproduces many important properties of the original time series, according to Taken's theorem. To use Taken's theorem effectively, the proper values for D and T must be chosen. It has been demonstrated that if the original attractor had dimension d, then D = 2d + 1 will suffice to recreate it.

To train the FNN and RNN the reconstructed vector is used for one-step-ahead prediction where 1 neuron is employed in the output layer. The number of input neurons in a feedforward network is denoted by D. In recurrent networks, 1 represents the number of input neurons. The embedding dimension D corresponds to the number of steps taken by the recurrent network in time.
The recurrent and feed-forward networks' prediction performance is measured using the RMSE. For further comparison with literature data, the NMSE is employed.

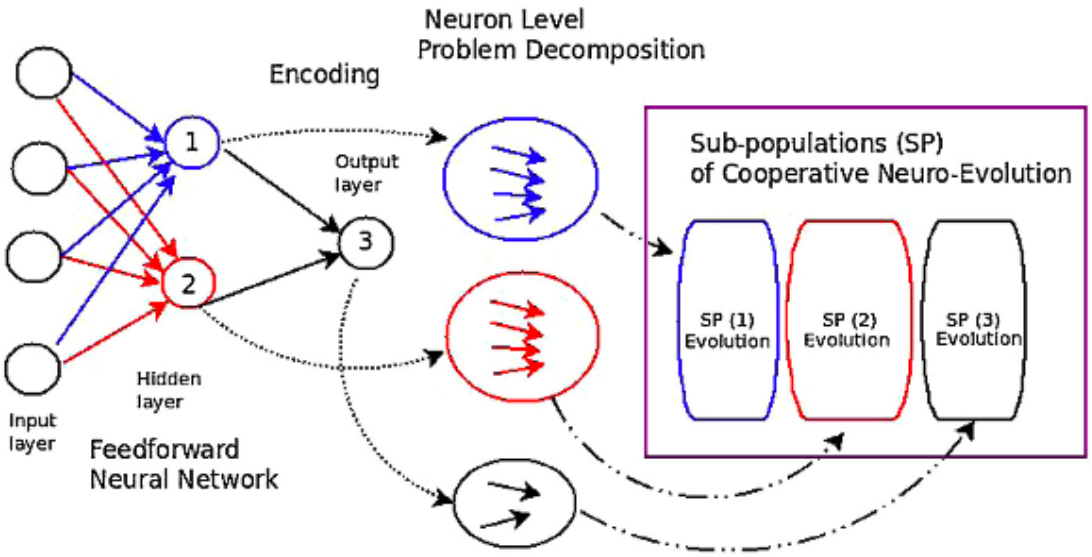


Fig 2.2.2.2 - Decomposition of NL problems using FNN. It's worth noting that the four input neurons indicate time series reconstruction with a four-dimensional embedding dimension. (Chandra & Chand, 2016)

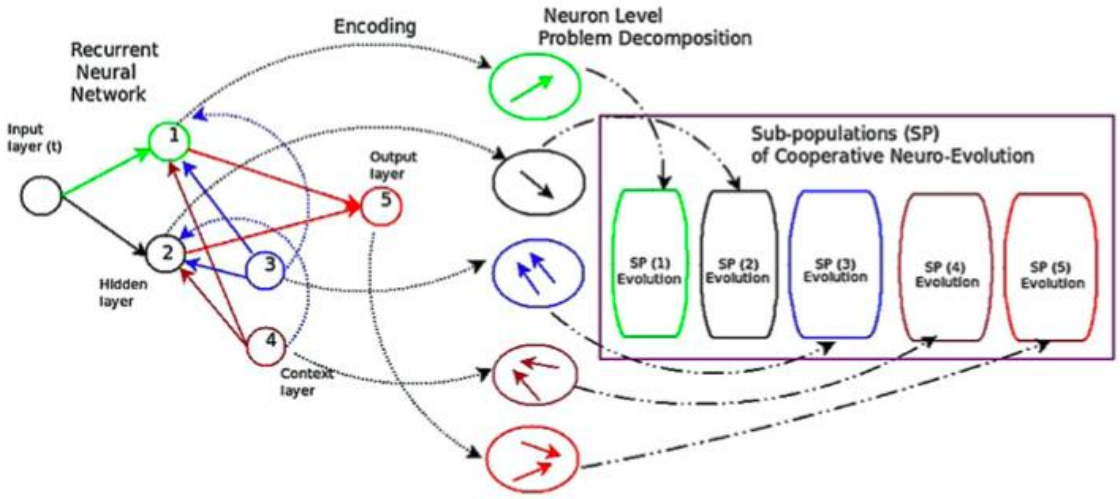


Fig 2.2.2.3 - With the NL problem decomposition, Elman RNN is used. It's worth noting that one input neuron can represent any embedding dimension for time series reconstruction. The network expands to the extent that the embedding dimension allows. (Chandra & Chand, 2016)

The NASDAQ stock exchange provided the financial time series data collection. It includes, , Staples Inc., Seagate Technology Holdings and ACI Worldwide daily closing prices. Each company's data set included closing stock prices from December 2006 to October 2010, totalling roughly 800 data points. Because stock prices varied over a three-year period, the dataset also includes data from the 2008 financial crisis in the United States. To create the training and testing datasets, we utilized Taken's theorem to rebuild the time series data with time lag T = 2 and embedding dimension D = 5.

The stock prices at the end of the day were scaled using min-max scaling. The algorithm's maximum function evaluations was set to 50,000, which served as a terminal condition. We utilized a population size of 300 people, same as the prior data sets. In the hidden and output layers of the corresponding neural network designs, sigmoid units were used.

2.2.3 Results

Their abilities are assessed in terms of training and generalization. The Mackey–Glass time series and the Lorenz time series are the two simulated time series in the benchmark chaotic time series issues. The Sunspot time series is a real-world situation.

For both issue decomposition approaches in the Mackey–Glass problem, the FNN was able to provide better generalization than the RNN. We note that the RNN showed higher generalisation compared to FNN for the Lorenz issue. Furthermore, NL failed to outperform

SL. When compared to the RNN, the FNN performed worse in the Sunspot issue. When NL was compared against SL for both network designs, NL performed better. In general, the results demonstrate that the NN with the least training error does not necessarily perform the best in terms of generalization. This is due to the dataset's over-fitting and noise. With fewer hidden neurons, performance in the Lorenz and Sunspot tasks was typically better. In the instance of the Mackey–Glass issue, however, this was not the case.

In two of the three benchmark problems, the RNN outperformed the FNN. Furthermore, for both neural network architectures, NL has given better performance when compared to SL in the majority of cases.

For both network designs, generalisation performance was extremely competitive. In terms of training and generalization, the RNN outperformed the FNN marginally. In addition, we found that the NL decomposition performed better than the SL decomposition. As the number of hidden neurons rose, so did the performance on NL.

RNN beat FNN in both generalization and training performance in several situations. We also found that the SL decomposition performed better than the NL decomposition for NN designs.

On both network designs, the RNN outperformed the FNN and the NL outperformed the SL. Because the training and generalization performance had a very significant variance, the offered approaches showed evidence of over-fitting with this problem. We also point out that training performance is far superior to generalization performance. This could be owing to a significant difference in the stock market's trajectory, which distinguishes the training dataset from the testing dataset.

In two out of the three experiments, NL outperformed SL. In a comparison of the two network topologies, we find that the RNN outperforms the FNN in terms of generalization. In all three experiments, the RNN outperformed the others. As a result, it was concluded that the RNN is better suited for prediction in general especially in finance domain.

2.2.4 Research Gaps

The experiments were created without a validation set so that the algorithms and neural network topologies could be compared simply on training data with a fixed training duration. To increase the prediction even more, multivariate time series could have been employed. Furthermore, machine learning techniques such as transfer learning might have been utilised to harness basic information in the many marketplaces that create the time series.

2.3 A Deep Learning Approach for Optimization of Systematic Signal Detection in Financial Trading Systems with Big Data(Arpaci & Karaoglu, 2017)

2.3.1 Business problem

The purpose of this research is to create an intelligent trading decision support system by using a RNN based technique to financial trading subsystems in order to discover systematic future signals in Big Data. The model was evaluated using data from the Istanbul Stock Exchange in this study.

2.3.2 Research Methodology

Amazon (S3) is used to store the daily stock market data compressed in gzip format. Because the algorithms demand a variety of types and forms, the data is in a raw format; it is their responsibility to format certain day intervals correctly.
We utilized the stock data of Istanbul Stock Exchange-Nasdaq suite's standard raw format. We used the full stock trading data from the year of 2016 in order to train our model for this study's assessments.

In the construction of successful machine learning models, choosing input parameters is crucial. Market microstructure features should be used to train machine learning models that have a substantial influence on the market which in turn allows learning the policy to condition and enhance forecasting accuracy. We discovered that the properties stated in Fig 2.3.2.1 are beneficial in our model, despite the fact that there are many possible parameters that might be employed in ML models.

Feature	Description
TradeCount15, TradeCount30, TradeCount60	trade volume within 15,30,60 sec time-step
Spread	Bid vs Ask price difference
PriceVwapDiff, PricePeriodicVwapDiff	differences of the price vs Real-time VWAP, differences between price vs periodic VWAP
BidAskTotalTradeVolumeDiff	Difference of trades happening at Ask price and trades happening at Bid price
TradeVol15, TradeVol30, TradeVol60	volume in 15,30,60 sec time-step
lambda15, lambda30, lambda60	Kyle's Lambda values under 15,30,60 sec time-step
FirstLevelImbalance, SecondLevelImbalance, ThirthLevelImbalance, ForthLevelImbalance, FifthLevelImbalance	5 Levels of Orderbook Imbalance (BidVolume-AskVolume) / (BidVolume+AskVolume)
Volatility	Daily highest - lowest price
Pt15, Pt30, Pt60	difference of sum of trades happening at the Ask vs Bid prices under 15, 30, 60 sec time-step x Kyle's Lambda values
DifTotalTurnoverPercent, DifTotalTurnover	Bid Turnover – Ask Turnover and Turnover %
BidOrAskDirection	Bid or Ask Trade relization
OpenChangePercent, LowChangePercent, HighChangePercent, PriceOpenDiff, PriceHighDiff, PriceLowDiff	previous closing price - current day lowest and highest opening prices, day change %
TotalVolume	total volume per day
DailyChangePercent	closing price – yesterday's closing price

Table 2.3.2.1 - Features used for model (Arpaci & Karaoglu, 2017)

PLR is a basic modelling approach that allows for dynamic incremental updates in a time series. The number of line segments may be adjusted dynamically to alter the granularity of PLR approximation (Keogh et al., 2001). The rapid changes in trade points might make it difficult to identify and utilize in prediction systems. As a result, we employ PLR as a sliding window to detect falls and spikes in stock market data. Our method uses PLR to identify turning moments as trading signals.

A complete gradient and bidirectional version of LSTM, Graves LSTM (Graves & Schmidhuber, 2005), (Graves, 2014), is used in the system for data modelling. LSTM (Hochreiter & Schmidhuber, 1997), (Gers et al., 2003) is an effective approach for analysing serial data with gaps of uncertain size between key occurrences as a form of recurrent neural network (RNN). Compared to standard hidden Markov models and RNNs, the major benefit of LSTM is its relative insensitivity to lag duration. By adding a network design

incorporating memory blocks, LSTM overcomes the problem of backpropagation faults (Hochreiter et al., 2001) in standard RNNs with lengthy time delays. The input, output, and forget gates make up the RNN memory blocks. With the activation of the input gate, each cell doubles the input. The forget gate multiplies the preceding cell values. The output gate multiplies the output to the network. As a result, the network's cells communicate through gates (Graves & Schmidhuber, 2005). Graves LSTM is used in our system since it performs better than the original LSTM (Graves, 2014).

Furthermore, our system employs the Exponential Smoothing method. This is a common technique for applying low-pass filters in order to eliminate short-term oscillations while looking at time series data. ES maintains the longer-term trend by flattening noise. In contrast to the moving average, which assigns equal weights to prior data, it reduces weights exponentially with time.

2.3.3 Results

We used trade data of Garanti Bank from September 2016 to January 2017 to analyze the performance of our model, which corresponds to the fourth quarter of 2016. During the time period, the data comprised about 400.000 transactions. To feed each trade into LSTM, we used a sliding window method. In a sliding window method, we employed a total of 20 timesteps for 36 characteristics. PLR with a price limit of 0.06 points were our goal values.

Following parameters were discovered to be a good fit for our three-layer model after examining multiple different parameter combinations inside the hyperparameter set. We used the Xavier init technique to set their weights. Graves LSTM is the first two layers, followed by RNN output layer. With width of the Graves LSTM as 32, and the RNN layer has single output per 20 timesteps. The learning rate was set to 0.1, taking RMSProp as updater function, and ReLU as activation function. The training phase started with a MSE of 14 which decreased down to 0.001.

When we used a batch size of 20, we found that the error rate fell regularly. The error function became noisy when the number was less than 20. When evaluating the standard deviations of gradients, all layers looked to have almost similar learning rate. Subsequently, the model's accuracy was enhanced by first performing standardization and then using min-

max normalization. We selected the first quarter of 2017 as the test period, resulting in an almost 50 percent train-test split.

2.3.4 Research Gaps

The study has been done on a single stock dataset of Garanti Bank. Some additional stocks should have been considered as well to see whether the proposed approach performs similarly in other datasets which contain some inherent noise. For identification of the most ideal probable trade points, a reinforcement learning approach could have been implemented on top of the suggested model.

2.4 Deep learning with long short-term memory networks for financial market predictions(Fischer & Krauss, 2017)

2.4.1 Business Problem

Financial time series prediction problems are notoriously tough, owing to the high noise level and the semi-strong form of market efficiency. The financial models used to create a link between these return prediction signals (features) and future returns (targets) are often transparent and incapable of capturing complicated non-linear relationships. Non-linear patterns in financial time series data can be recognized using ML techniques, according to preliminary evidence as per (Huck, 2009), (Takeuchi, 2013), (Moritz & Zimmermann, 2016), (Dixon et al., 2015). For the sake of comparison, this study uses the same dataset as that of (Krauss et al., 2017) and builds on his work.

In this work, focus is largely on deep learning and how it may be used to solve a large-scale financial time series forecasting problem. Three additions to the literature have been made in this regard.

- LSTM networks, for example, are one among the most sophisticated DL architectures for sequential learning applications including speech recognition, handwriting recognition, and time series prediction. More emphasis on offering a comprehensive guide to data preparation, also the creation and training for financial time series prediction tasks, or the impact of adding news for specific businesses. Finally, compare the results against a random forest, to demonstrate the value-added by the LSTM, and a standard LOG model from the literature (to establish a baseline).

- Find the stocks chosen for trading which have below-average momentum, significant volatility, extreme directional moves in the days leading up to trading and a propensity to reverse these extreme movements in upcoming time period.
- Finally, combine the data from the previous section into a simpler, rules-based trading strategy that attempts to capture the core patterns that LSTM uses to identify successful and losing stocks.

2.4.2 Research Methodology

To begin, the raw data is first divided into study periods, each of which includes training sets(used for in-sample training) and trade sets (used for out-sample training). Next, go through the essential parameter space and objectives for training and prediction.

A "study period" is defined which a training-trading split is set that consists of a 750-day training phase and a 250-day trading period. Then, from 1990 to 2015, divide the full data set into 23 research periods with non-overlapping trade periods.

For the training set, we take into account all stocks that are S&P 500 constituents on the last day of the training period that have available historical data. Some stocks show the whole 750-day training history, while others only show a fraction of it, such as when they are listed later. If a component has no price data after a specific trading day throughout the trading period, it is regarded for trade until that day.

For training, LSTMs need sequences of input parameters, or the values of the input parameters at different time-steps. The standardized one-day return is our only feature. We choose a series length of 240, which corresponds to about one trading year's worth of data. As a consequence, we have 240 standardized one-day returns in overlapping sequences.

The author has followed (Takeuchi, 2013) and designed a binary classification issue, in which the output variable for each stock s and date t can take two distinct values for the sake of comparison. To determine the two classes, the one-period returns of all stocks s in period t+1 are ordered and divided into two equal-sized classes. If the cross-sectional median return of all stocks in period t+1 is more than one-period return of stock s, class 0 is achieved.

Similarly, if the cross-sectional median is less than or equal to one-period return of stock s, class 1 is attained.

The study uses keras to train the LSTM network and has used three sophisticated techniques. RMSprop is used as an optimizer first, followed by dropout regularization in the recurrent layer. Early halting is often employed as a safeguard against overfitting.

The following is the topology of our trained LSTM network:

- One parameter and 240 time-steps in the input layer.
- an LSTM layer with dropout of 0.1 and 25 hidden neurons.
- A typical setup includes an output dense layer consisting of 2 neurons with a softmax activation function.

The LSTM model is benchmarked against a Radom Forest, DNN and a Logistic Regression Model. For two reasons, the author has chosen a random forest as a benchmark. For starters, it's a cutting-edge ML model that takes almost minimal tuning and consistently produces good results. Also, in (Krauss et al., 2017) and (Moritz & Zimmermann, 2016) - a large-scale ML application using monthly stock market data - RAF in this configuration are the best single approach and the method of choice. As a result, random forests are an excellent baseline for every new machine learning model.

To demonstrate the relative benefit of LSTM networks, a typical DNN is used. A FNN is used with 31 input neurons followed by 31 neurons in the 1st hidden layer followed by 10 neurons in the second hidden layer followed by 5 neurons in the 3^{rd} hidden layer and lastly 2 neurons in the output layer. Following (Goodfellow et al., 2013), the activation function is maxout with two channels and softmax in the output layer. L1 regularization is used with 0.00001 shrinkage, and dropout is set to 0.5.

A logistic regression model has also been used as a baseline model. Details regarding our implementation may be found in the sci-kit learn manual (Pedregosa et al., 2011) and its references. L2 regularization is chosen from a set of 100 options on a log scale ranging from 0.0001 to 10,000 using 5-fold cross-validation on the training set, and LBFGS is used to find an optimum with a maximum of 100 iterations. In compared to a conventional classifier, we

use logistic regression as a baseline to calculate the additional impact of the far more sophisticated and computationally demanding LSTM network.

2.4.3 Results

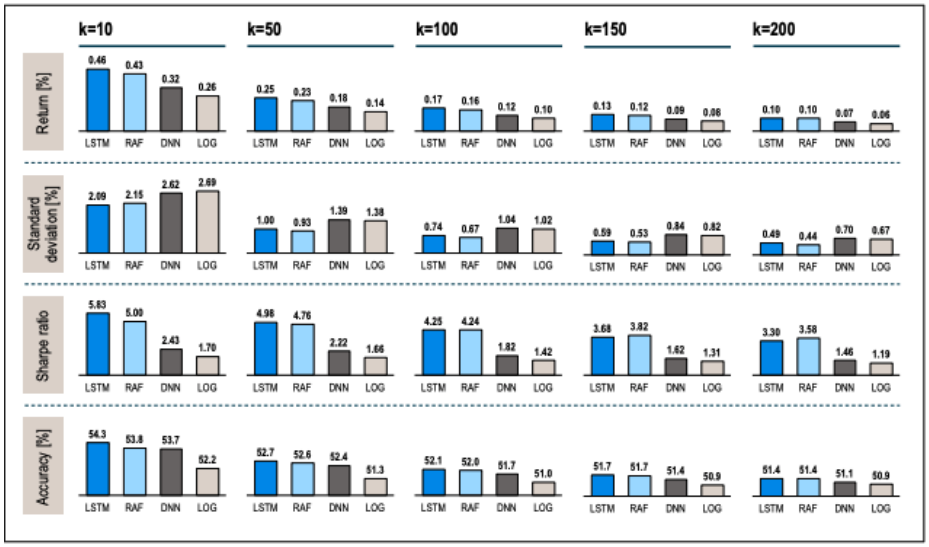


Fig 2.4.3.1 - Performance characteristics for long-short portfolios of various sizes on a daily basis(Fischer & Krauss, 2017)

k = portfolio equity size

Regardless of the portfolio size k, the LSTM outperforms the other methods. For k = 10, daily returns before transaction costs are 0.46 %, against 0.32 % for the DNN, 0.26 % for the Logistic Regression and 0.43 % for the RAF. Except for k = 200, when it is tied with the RAF, the LSTM also gets the greatest mean returns per day for bigger portfolio sizes. The LSTM and the RAF are similar in terms of standard deviation, a risk metric, with slightly smaller values for k = 10 and slightly greater values for rising portfolio sizes. Across all values of k, both LSTM and RAF have a considerably smaller standard deviation than DNN and LOG. The LSTM has the greatest return per unit of risk (Sharpe ratio), up to k = 100, and slightly less than the RAF for even bigger portfolios, when the RAF's lower standard deviation exceeds the LSTM's greater return. An essential machine learning measure is accuracy, which refers to the percentage of correct classifications. The LSTM has a clear advantage for the k = 10 portfolio, a small advantage till k = 100, and a tie for rising sizes with the RAF.

2.5 Deep Learning in Finance (Heaton et al., 2018)

2.5.1 Business Problem

Financial forecasting issues are both practical and theoretically fascinating. They're also extremely intimidating. According to theory, most information important to financial prediction issues is dispersed throughout accessible economic and other data, a concept backed up by the numerous divergent data sources that market players monitor for hints on future price movements.

It's challenging to deal with so many different data sources. Financial economic theory does not adequately specify the relevance of the data or the potentially complicated non-linear interactions in the data, thus there is a huge collection of potentially relevant data. In reality, this leads to a slew of predictive models, many of which lack theoretical support and are prone to overfitting and poor predicted out-of-sample performance.

What is required is a technique capable of learning the complicated characteristics of data inputs that result in accurate predictions of the desired output variables (such as an asset or portfolio return).

The study introduces deep learning hierarchical decision models for financial prediction and categorization in this work. The deep learning predictor provides a variety of benefits over standard predictors, including the ability to learn from experience.

2.5.2 Research Methodology

The study applies a hybrid AE + LSTM approach with Smart Indexing. The concept of applying Smart Indexing is as follows:

There are two basic methods we may use when attempting to duplicate (or approximate) a stock index using a subset of equities.

1. Identify a small set of equities that have historically performed similarly to the index under consideration.
2. Identify a small set of stocks which historically have represented an over-proportionally significant share of the total aggregate information of all the stocks the index consisted of.

While 1 and 2 may appear to be fairly similar on the surface, they represent completely distinct methods.

Many traditional techniques to index replication are based on linear regression, which belongs to group 1. We frequently try to discover a small group of stocks that in-sample offers a decent linear approximation of the considered index via trial and error.

The deep learning version of 1 allows for the translation of input data into a desired output via a hierarchical series of adaptive linear layers, which implies that even non-linear correlations may be easily detected during training. We call the resultant approximation (or prediction) method a deep feature policy since each buried layer gives a different interpretation of the input features.

The availability of customized non-linear connections in deep learning reduces the traditional goal of 1, namely good in-sample approximation, to a triviality, and shifts the focus to out-of-sample performance training.

Another flaw with 1 is that it corresponds to the final (and hence diluted) product. A deep auto-encoder solves this problem by directly (rather than indirectly) estimating the aggregate information contained in the index stock family in question.
The construction of an auto-bottleneck encoder produces a compressed data set from which all stocks are recreated. As a consequence, when it comes to indexing, the stocks that are closest to the compressed core of the index may be thought of as a non-linear foundation of the aggregate information of the considered family of firms.

During the 2014/15 financial year, all S&P500 equities were auto-encoded. We then rated the stocks based on their closeness to their own auto-encoded version; the closer a stock was to its own auto-encoded version, the greater its shared information richness.

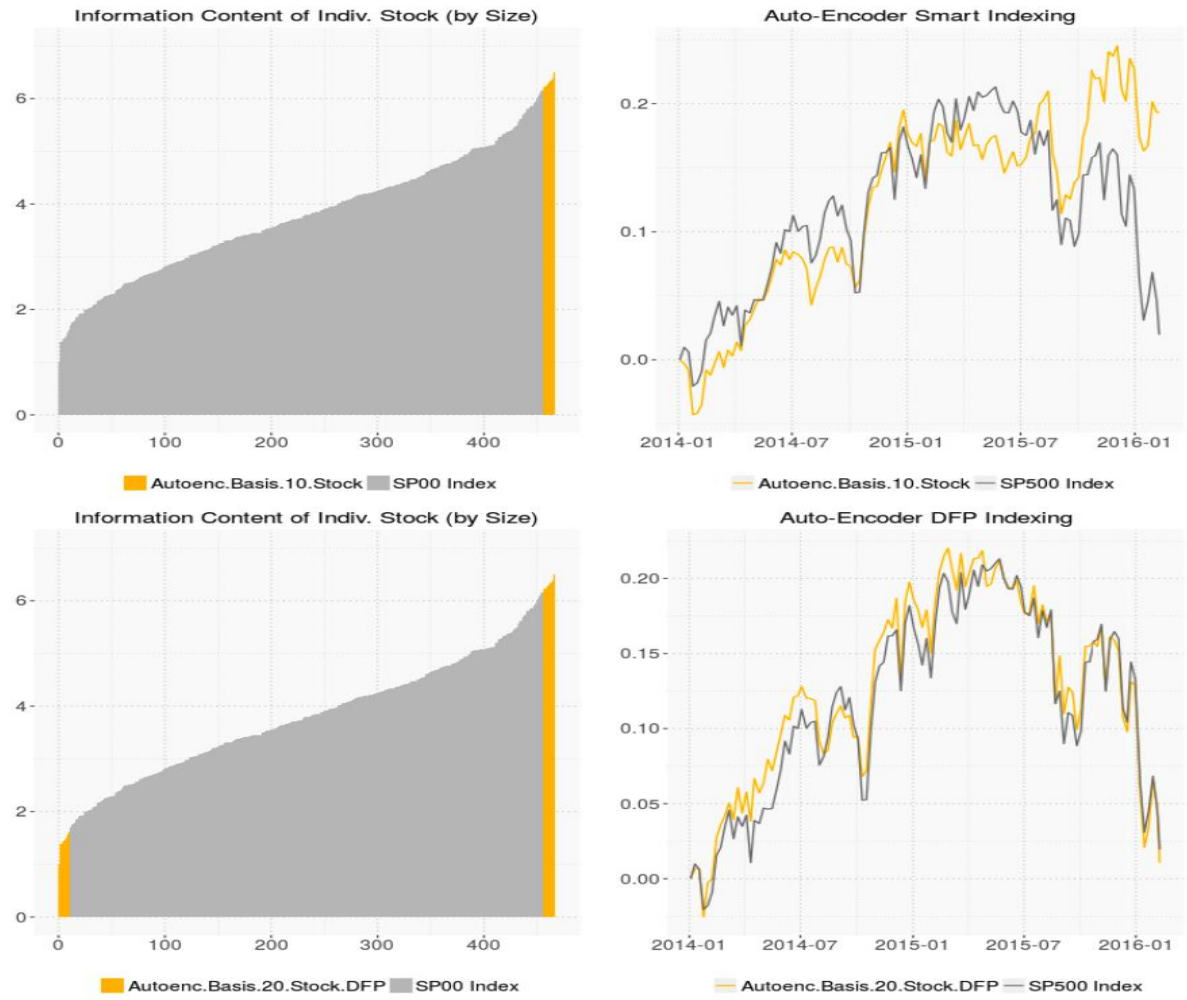


Fig 2.5.2.1: From the auto-encoder basis of the 10 top stocks, develop an equally weighted portfolio based on their proximity to the auto-encoded information. (Heaton et al., 2018)

We auto-encoded all equities in the S&P500 throughout the 2014/15 timeframe in Figure 2.5.2.1. We then rated the stocks based on how near they were to their own auto-encoded version; the closer a stock was to its own auto-encoded version, the greater its shared information richness. The S& P500 may be approximated by just investing in the 10 stocks having the largest community information content, as shown in the upper right.

While the ten stock auto-encoder foundation is fair, we note that the approximation is somewhat wrong, especially in the latter seven months of the training period. It's enlightening to see how utilizing a DFP index approximation, this divergence may be easily prevented in-sample.

We merged the two groups of 10 stocks with the greatest and lowest communal information, respectively, in Figure 2.5.2.1 at the bottom, and then trained a deep learning procedure to

approximate the S&P500 index based on this extended foundation of twenty stocks during the same period of 2014/15. The DFP generates an optimal action for approximating the target index for each combination of inputs from the selected twenty stocks, based on the hierarchical composition of non-linear characteristics derived from the input data. Given enough variation in the input data, a DFP may frequently be taught to estimate the target data to almost arbitrarily high accuracy, as seen in the bottom right chart over the last six months of the training period.

2.6 A deep learning framework for financial time series using stacked autoencoders and long-short term memory(Bao et al., 2017)

2.6.1 Business Problem

This paper offers a unique deep learning framework for stock price forecasting that combines stacked autoencoders (SAEs), wavelet transformations (WT), and long-short term memory (LSTM). This study aims at applying a deep nonlinear topology for time series prediction. By extracting strong features that capture the key information from the complex real-world data, tthe new model can successfully achieve even better performance than before, which is an improvement above traditional machine learning approaches.

2.6.2 Research Methodology

Few attempts have been made to examine if the stacked autoencoders approach might be used to financial market prediction, and this work contributes to that field. It proposes a unique stock market prediction model based on the stacked autoencoders method.

WT, SAEs, and LSTM are the three components of the suggested model in this study. The SAEs are a key component of the model, since they are utilized to learn the characteristics of financial time series in an unsupervised manner. It's a single-layer autoencoder neural network with each layer's output feature linked to the inputs of the following layer. SAEs are trained unsupervised one at a time, with output and input data compared to reduce error. As a result, the SAEs model is capable of learning both invariant and abstract features.

To assist enhance prediction accuracy, the other two approaches are used. Because it is ideal to learning from past experience and forecast time series with variable size windows, the LSTM, a kind of RNN, was used. Financial time series are believed to benefit from WT since it reduces noise. For single-dimensional data, it is a common filtering and mining approach.

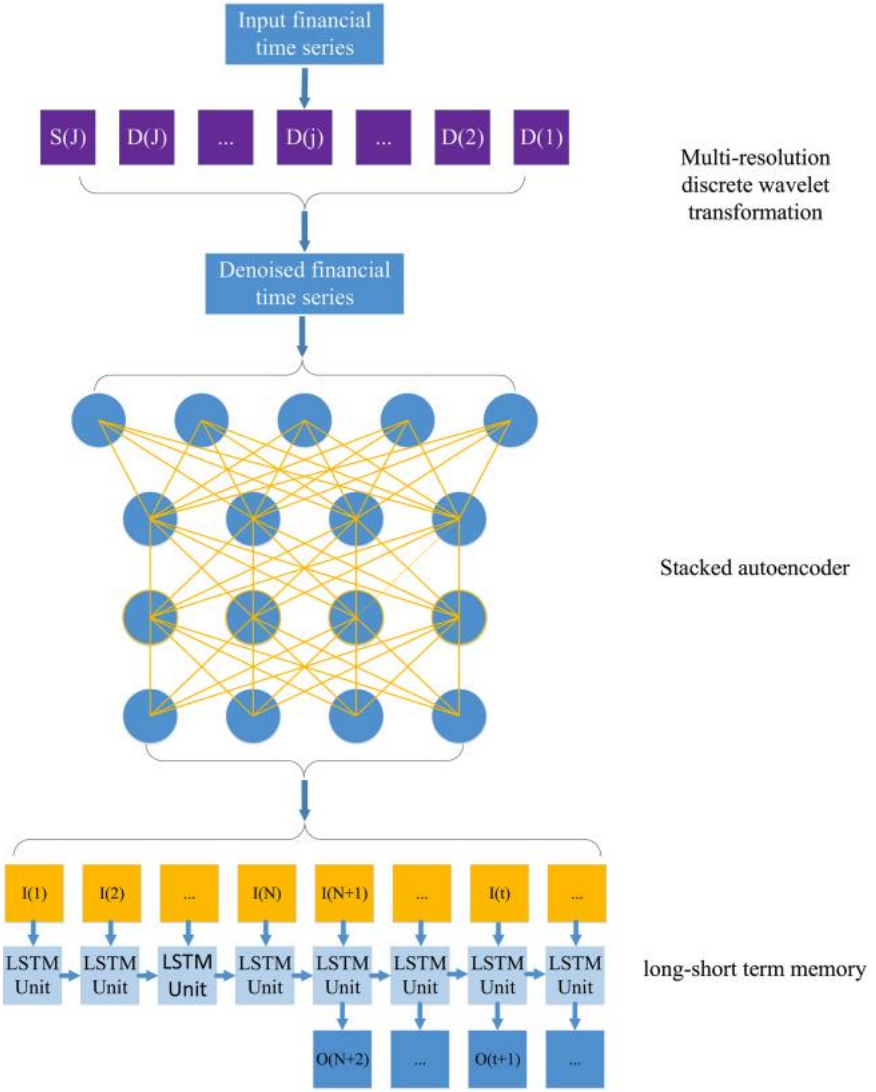


Fig 2.6.2.1 - The suggested deep learning architecture for financial time series is depicted as a flowchart. The detailed signal at the j-level is D(j). At level J, S(J) is the coarsest signal. At time step t, I(t) and O(t) represent the denoised feature and one-step-ahead output, respectively. The number of LSTM delays is N. (Bao et al., 2017)

The framework consists of three stages:

- eliminate the noise by using the wavelet transform to deconstruct the time series
- use SAEs with an unsupervised deep architecture
- generation of one-step-ahead output using long-short term memory with delays.

2.6.2.1 Data Description

CSI 300, Nifty 50, Hang Seng index, Nikkei 225, S&P500, and DJIA index are the six stock indexes used in this study. As the neural network's validity may be influenced by market conditions. Samples from various market circumstances may be useful in resolving this issue.

The NYSE, often regarded as the world's most rich and developed financial exchange, trades the S&P500 and DJIA indices. Mainland China and India's financial markets, on the other hand, are usually classified as new markets. In actuality, the majority of the country's market institutions are still in the planning stages. The Nifty 50 and CSI 300 were chosen to represent emerging markets as a consequence.

As inputs, three sets of variables have been chosen. The first set of variables is each index's historical trade data. The data contains the OHLC variables as well as the trade volume, as per prior literature. Each index's fundamental trade information is represented by these variables. Another set of inputs for each index is a collection of 12 frequently utilized technical indicators.

2.6.3 Results

According to the model findings during a 6-year time span, RNN and LSTM had larger distances and variations to the real data compared to WLSTM and WSAEs- LSTM, according to the yearly expected data and the corresponding actual data in the graph. In addition, when comparing WLSTM to WSAEs-LSTM, the latter beats the former: WSAEs-LSTM is less volatile and is similar to real-time trade data compared to WLSTM. Prediction Performance of WSAEs-LSTM appears to be more apparent in developing markets compared to established ones.

Not just on average, but year after year, WSAEs-LSTM beats the other three. To show the robustness of our findings, the study looks at the statistical relevance of discrepancies between the other three models and WSAEs-LSTM. For each accuracy parameter, we compare the 24 quarterly WSAEs-LSTM results to the three models using T-test statistic.

According to the data, the difference in predictability between two particular models in forecasting two stock indices is modest if the two stock indices are traded in markets with comparable development states, but it rises if the two stock indices are exchanged in markets with distinct development states.

2.7 A Recurrent Neural Network Approach in Predicting Daily Stock Prices(Samarawickrama & Fernando, 2017)

2.7.1 Business Problem

The purpose of this study was to create models based on the Recurrent Neural Network (RNN) Approach to forecast daily stock values of chosen listed firms on the Colombo Stock Exchange (CSE), as well as to assess the accuracy of the models generated and, if necessary, to detect model flaws. Models were built using FNN, SRNN, GRU, and LSTM architectures. For each firm, the OHLC prices from the previous two days were chosen as input variables. The biggest and lowest predicting errors are produced by feedforward networks. The finest feedforward networks have a predicting accuracy of around 99 percent. When compared to feedforward networks, SRNN and LSTM networks yield lesser mistakes on average, but on rare instances, the error is greater. GRU networks have a stronger predicting capability than the other two networks.

2.7.2 Research Methodology

Three firms were chosen for this study among the Colombo Stock Exchange's (CSE) 297 listed companies. These businesses were chosen from among 20 CSE sectors from three industries with the greatest trading volume per year. The companies were chosen based on the stock price's stability and market performance (trading volume and frequency). The following companies were chosen for this study: COM(Finance), RCL(Manufacturing) and JKH(Diversified Holding Sector). The model was built using data from these firms from January 1, 2002, to June 30, 2013.

The Low and High prices of the previous 2 days have been chosen as inputs for model building.

As a result, input params (t – current time) consist of:
C(t-1) – day t-1 closing price
C(t-2) – day t-2 closing price
L(t-1) – day t-1 lowest price
L(t-2) – day t-2 lowest price
H(t-1) – day t-1 highest price
H(t-2) – day t-2 highest price

Three prominent neural network topologies were used for model development in this study are GRU, LSTM and SRNN. The Feed Forward Neural Network (MLP) has also been utilized for comparison. Ten NN models were created for each of the neural network designs which were created with different latent/hidden units between 2 and 11. As a result, each firm had 40 NN models.

Each NN model contains a 6 input neuron input layer, a 2 to 11 hidden neuron hidden layer, and a 1 output neuron output layer. The internal architecture of each of ten models built using a single design for a single firm is described in the table 2.7.2.1.

The data was scaled using min-max scaling in this investigation. The train, validation and test data ratio was taken as 70:15:15.

On the Windows platform, the Keras was used to implementation and training of the NNs. To train neural networks, the following activation functions and features were utilized.

Parameter	MLP	SRNN	LSTM	GRU
Initialization function	uniform	glorot uniform	glorot uniform	glorot uniform
Inner Initialization function	-	orthogonal	orthogonal	orthogonal
Activation function	relu	softsign	softsign	softsign
Inner activation function	-	hard sigmoid	hard sigmoid	hard sigmoid
Error (loss) function	MSE	MSE	MSE	MSE
Optimizer/Training algorithm	adam	rmsprop	rmsprop	rmsprop

Table 2.7.2.1 - Model parameters used for training(Samarawickrama & Fernando, 2017)

All the 40 NN models underwent 5000 rounds of training. So, for all of the networks, the halting condition was 5000 iterations.

The test dataset was used to make predictions after training. MAPE and MAD were calculated to quantify the accuracy of each neural network model in order to assess its performance. The model with the lowest MAPE/MAD was deemed the best.

$$MAD = \frac{1}{|ValidationSet|} \sum_{for.all.days \in .ValidationSet} \left| price_{forecast}^{tomorrow} - price_{real}^{tomorrow} \right|$$

$$MAPE = \frac{1}{|ValidationSet|} \sum_{for.all.days \in .ValidationSet} \left| \frac{price_{forecast}^{tomorrow} - price_{real}^{tomorrow}}{price_{real}^{tomorrow}} \right|$$

To compare projected prices to actual prices, the results from the tests were converted back to raw format.

2.7.3 Results

When it comes to test error (or forecast error), we observed that MLP models yield the lowest and highest mistakes in this study. The finest feedforward networks have a predicting accuracy of around 99 percent. When compared to feedforward networks, SRNN and LSTM networks yield lesser mistakes on average, but on rare instances, the error is greater. GRU networks produce far greater predicting errors than the other two networks.

In most research, RNN models (particularly LSTM) gave the best results when incorporating the findings of prior investigations. MLP models, on the other hand, yield the greatest outcomes in this investigation. This is due to the fact that only data from the previous two days were used as inputs in this study. RNN models would give the best results if the number of previous days examined for input variable selection was increased.

2.7.4 Research Gaps

The efficacy of RNN-based LSTM is being eclipsed by Multilayer Perceptron models in this study since only data from the previous two days was used as input. If the dataset could have been expanded with more data points, the performance of RNNs should have been higher. Furthermore, this research is restricted to the three equities picked, which may have been more widely distributed.

2.8 CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets(Liu et al., 2017a)

2.8.1 Business Problem

The CNN and CNN-LSTM neural networks are used to model and analyze quantitative selection and quantitative timing strategy in stock markets in this article. The CNN is used methodically to create a quantitative stock selection approach, and then the LSTM is used to

create a quantitative timing strategy for increasing profits. Experiments have shown that the CNN-LSTM model may be utilized to build quantitative strategies that outperform the Benchmark index.

2.8.2 Research Methodology

The CNN-LSTM architecture specifics and parameter details are as follows:

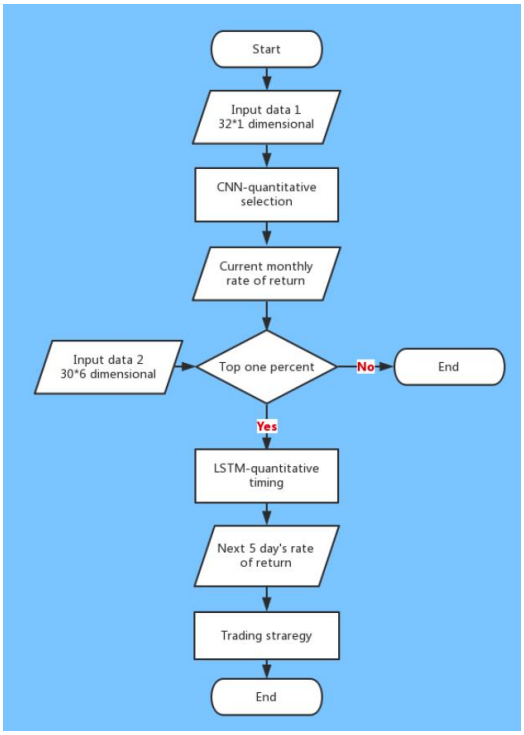


Fig 2.8.2.1 - Methodology Flowchart for CNN-LSTM(Liu et al., 2017a)

Parameters	CNN	LSTM
Input Layer	1	1
Conv/LSTM hidden Layer	2	1
FCN hidden Layer	2	1
Output Layer	1	1
Epoch	500	100
Activation	ReLU,Tanh	Tanh
Weight	Normal(0,1)	Normal(0,1)
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Objective function	cross-entropy	cross-entropy

Table 2.8.2.1 - CNN-LSTM Parameter Details(Liu et al., 2017a)

When necessary, data was subjected to z-score normalization. The training data spans the years 2007-1-1 to 2013-12-31, whereas the testing data spans the years 2014-1-1 to 2017-3-31. Only the characteristics that will be input into the neural network are chosen, and the

neural network is trained with random weights and biases. The LSTM component of the CNN-LSTM model is made up of consecutive layers, 1 LSTM layer, followed by a dense layer. Activation function used for dense layer is Tanh

The dropout parameter is introduced to the CNN-LSTM model, and to avoid overfitting the model, the regularization term is added to the weight.

We purchase and hold for 5 days when the LSTM projected value equals 1, then update the number of held days to 5 and hold for another 5 days. If the predictive value of LSTM is equal to -1, it will continue to maintain short positions; previously held shares will have their number of held days decreased by one; and if the number of held days is equal to 0, the share will be sold.

2.8.3 Results

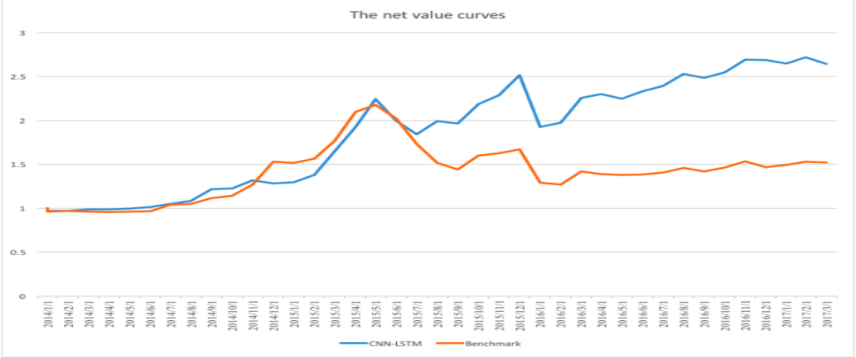


Fig 2.8.3.1 - CNN-LSTM vs Benchmark net value curves(Liu et al., 2017a)

	CNN-LSTM	Benchmark
Annualized rate of return	0.339	0.135
Maximum retracement	0.235	0.417

Table 2.8.3.1 - CNN-LSTM vs Benchmark(Liu et al., 2017a)

Our CNN-LSTM model's net value, rate of return, and maximum retracement are compared to the Benchmark index.

The net value curves show a considerable qualitative improvement in the performances. The greatest retracement of CNN-LSTM is reasonable during a stock market crisis. Our CNN-LSTM model's annual return rate is 2.5 times that of the Benchmark index's annualized rate of return. Meanwhile, the CNN-LSTM model's maximum retracement is 56% of the

Benchmark index's maximum retracement. The trials demonstrate that CNN-LSTM model is economical and that the returns are outstanding, as well as the algorithm's resilience and practicability.

2.8.4 Research Gaps

Despite the fact that the model-building process has been given, the dataset specifics have not been well clarified. In order for the study's results and conclusion to be more understandable, the details of the chosen financial asset should have been made explicit.

2.9 Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation(Yuan et al., 2018a)

2.9.1 Business Problem

DWNNs are presented in this paper, which can analyze temporal input in depth (time step) while extracting the correlation feature in the breadth direction. For our research, we use stock market data, which has an interesting correlation pattern. According to empirical data, DWNNs perform 30 percent better than regular RNNs in terms of performance.

The concept of integrating DNNs, LSTMs, CNNs and other neural models has been discussed in the past. However, the models in this research were trained individually, and the previous outputs were combined using a combination layer. The model is being developed further based on its foundation, which combines LSTMs, DNNs and CNNs into a single model that can be trained parallelly.

The merger of several neural models is based on feeding better features to LSTMs as input, which can optimize the performance of RNNs while lowering the spectrum variance of CNNs. As a consequence, the author introduces CLDNN, a system that imports input data into CNN layers for enhanced feature extraction, then implements LSTMs depending on the feature, and lastly outputs output using fully connected DNNs. The CNN-BLSTM presented in the article combines RNNs and CNNs to increase performance. According to these research, combining RNNs, CNNs and other NN models is a viable option.

2.9.2 Research Methodology

For many sets of correlated temporal data, each temporal data can be extended in time step indefinitely. If we use a generic RNN model to handle the data, we'll require m RNN models,

each of which is trained separately when dealing with such difficulties. As illustrated in Figure 2.9.2.1, the **m** RNNs have been expanded across time steps.

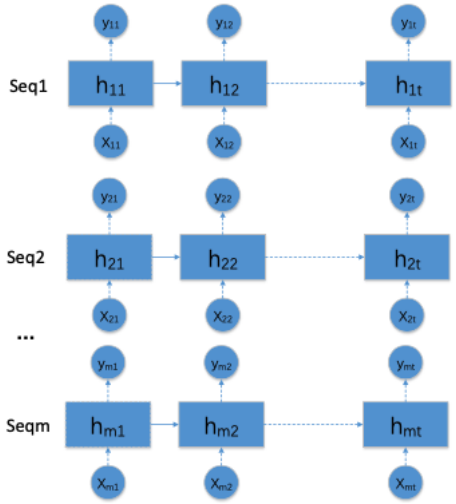


Fig 2.9.2.1: M generic RNN models were extended across timestep. (Yuan et al., 2018a)

We suggest a DWNN to improve this problem because this type of processing method obviously ignores vital information regarding the correlation between temporal data. Figure 2.9.2.2 depicts the DWNN structure of consecutive time steps. Between the RNN model's time steps **t** and **t-1**, we add CNN layers. The CNN layers are given the latent states of the **m** RNN models at time **t-1**. At time t, the CNN layer outputs are used as the state input for the RNN models. If we consider the hidden state to be the RNN's memory or history, the state processed by the CNN layers is the memory of the m RNN model. Prior to time t, this new state includes all of the memory of the m RNN.

Consider the generic RNN model, which operates with a variable-length sequence $X = (x_1,..., x_T)$ and contains a hidden state h and an output y.

The RNN model's status update formula is

$h_t = f(h_{t-1}, x_t)$

at each time step t, where f is a non-linear activation function.

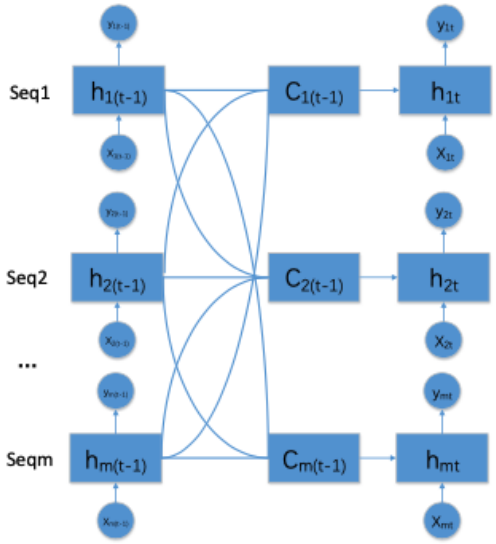


Fig 2.9.2.2: Structure of neighbouring time steps in a DWNN(Yuan et al., 2018a)

Now we consider the DWNN structure. Here, m RNNs are required to train m input sequences {Xi, 1 <= i <= m}, where Xi = (x_{i1},...,x_{iT}). Here, input m refers to sequences of different lengths, {Xi, 1 <= i <= m}. The DWNN model's status update formula at time step t is

$h\_temp_{it} = f_i(h_{i(t-1)}, x_{it})$ for 1<= i <= m

$h\_c_{it} = concat(h_{tempit}, 1 <= i <= m)$

$h_{it} = C_i(h\_c_{it})$ for1 <= I <= m

where f_i is the i_{th} RNN model's nonlinear activation function of the RNN cell;

concat is used to concatenate the hidden state of the m RNN Cell to generate a big vector

C_i refers to the CNN layer operation in the i_{th} RNN

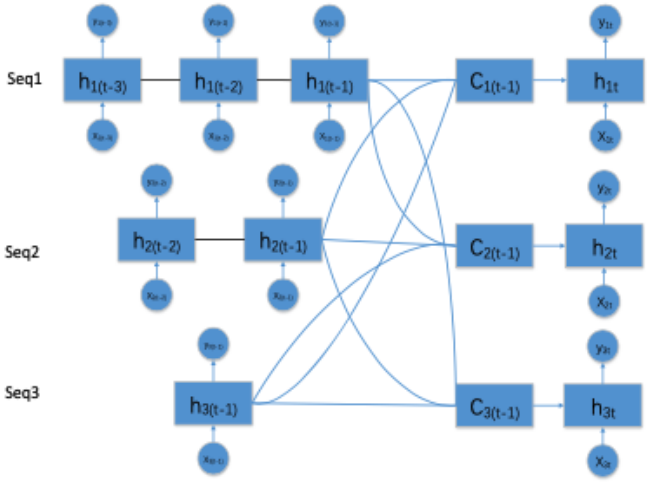


Fig 2.9.2.3: DWNN across different period(Yuan et al., 2018a)

To build single RNN chains, we use the Seq2seq model(Cho et al., 2014). The Encoder and Decoder halves of the Seq2seq model are built using two RNNs each. The Encoder part

translates the input into a vector of constant size, which is subsequently mapped to a target sequence of variable length by the Decoder component.

We employ the Seq2seq model's m groups to handle m groups of temporal data sequences at once. CNN layers are added to the encoder, while the decoder remains separate and unaffected. As per Seq2seq, to implement CNNs in DWNNs, these are the following ways:

1) The Seq2seq model in (Cho et al., 2014) employs a single layer RNN with GRU. This model is straightforward to construct since GRU only has one hidden state and just one set of CNN layers.

2) The RNN cell in the Seq2seq model in (Sutskever et al., 2014) is an LSTM. Because LSTMs have two latent states, c and h, DWNN requires the deployment of two groups of CNN layers.

3) (Sutskever et al., 2014) claims that Seq2seq model's performance can be enhanced by using reverse input data. As a result, building a DWNN with a bidirectional RNN (Graves & Schmidhuber, 2005) requires twice as many CNN layers.

4) According to (Cho et al., 2014), the Seq2seq model employs a four-layer RNN. The DWNN construction is adaptable for multi-layer RNN models. CNN layers can be built for each layer, or some.

In each RNN loop step, a CNN layer is added, but Such models are difficult to train and run the danger of overfitting. As a result, we follow the design of an interval CNN, which involves adding a CNN layer after every time window k which decreases the model's complexity while also allowing it to extract relationships. Previous RNNs struggled to deal with temporal data of varying lengths of time. However, we may create a CNN layer at the common multiple of time window, using DWNN utilizing interval CNN layer.

TensorFlow has been used to implement the DWNN network proposed in this paper. To train the algorithm, stock data from China's SSE has been used. The whole dataset contains 12 stocks which are 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679,

002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679 The dataset comprises daily k-line data from January 1, 2000, to August 16, 2017, and includes the OCHLV data. The training samples date from 2000/01/01 to 2015/12/30, whereas the testing samples date from 2016/01/01 to 2017/08/16.

The DWNN is made up of of 12 seq2seq models utilizing GRU; Encoder is set at time window of 60 days, and the length of the Decoder is set as 7, implying that 60-day historical data will be utilized to forecast data for the next week(days). To convert the dimension of the data into the form of the GRU hidden state, we use the whole connection layer. Each CNN layer has two convolution layers as well as a pooling layer.

In the experiment, instead of utilizing the order of data produced input samples, we use the random sampling technique with the Batch size as input. The samples were produced in such a way that the loss varied substantially early in the training. It does, however, have the benefit of regularization and, to a degree, improving the training performance.

2.9.3 Results

The interval CNN layer is compared to the DWNN model using the seq2seq model. Figure 2.9.2.3 shows the comparison experiment results, while The DWNN model results are shown in Figure 2.9.3.1, with the blue dots indicating the actual rate of growth in stock prices over the following 7 days and the orange plus sign showing the projected value.

When comparing the two graphs, we can observe that the DWNN model outperforms the general RNN model significantly.

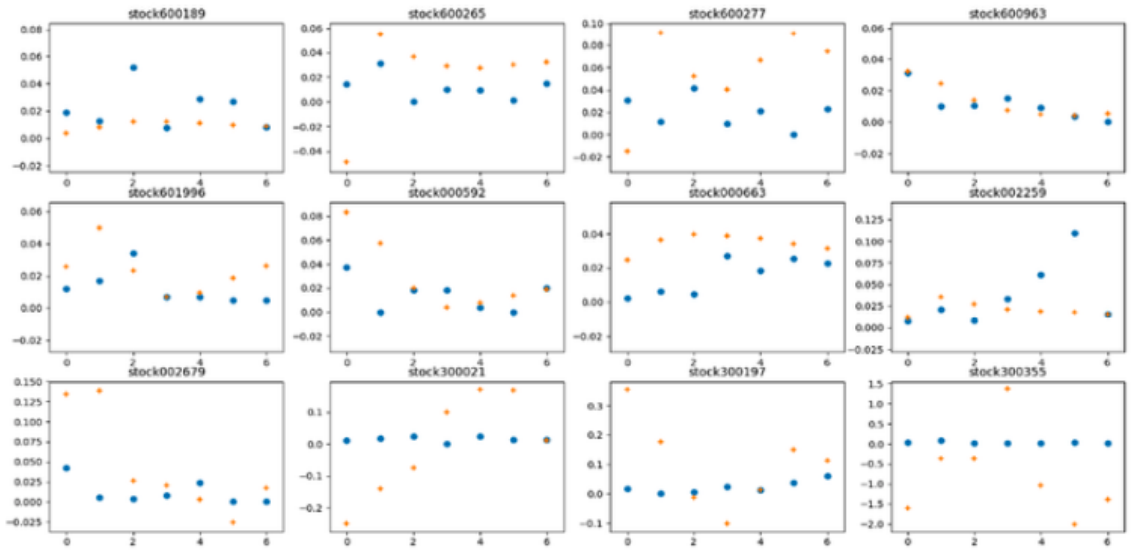


Fig 2.9.3.1 - comparison results(Yuan et al., 2018a)

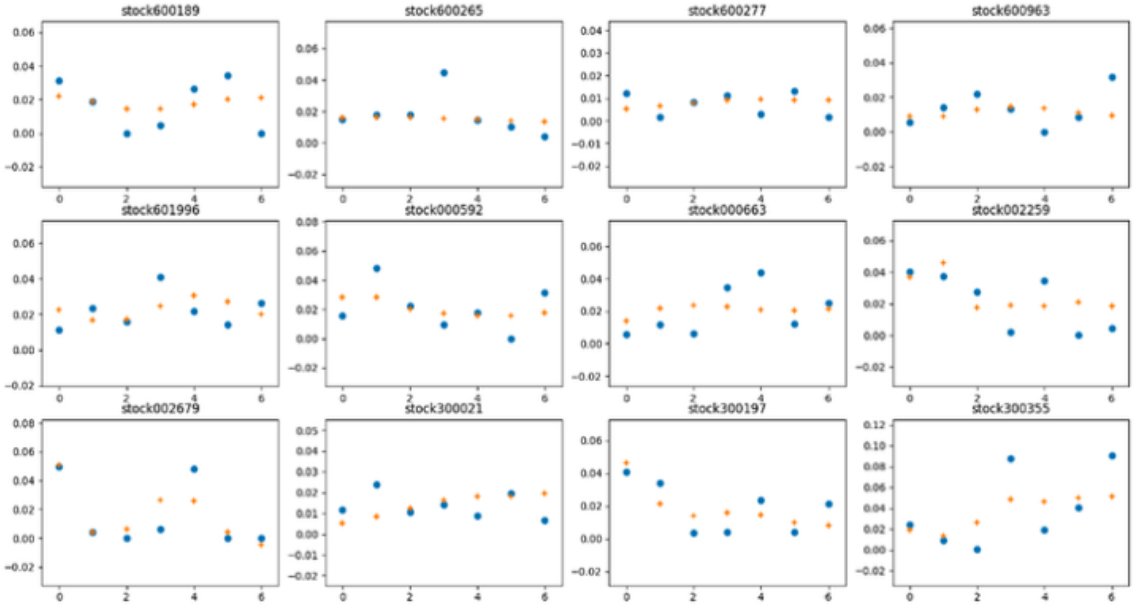


Fig - 2.9.3.2 - DWNN results(Yuan et al., 2018a)

2.10 Stock Price Prediction via Discovering Multi-Frequency Trading Patterns (Zhang et al., 2017)

2.10.1 Business Problem

Long and short term trading activity represents diverse trading patterns and frequency affect stock values. These patterns, on the other hand, are usually elusive because they are impacted in the real world by a range of unexpected political and economic factors, like corporate performance, breaking news that travels across markets and government restrictions. Furthermore, stock price time series are non-linear and non-stationary, making forecasting future price movements difficult.

To address these difficulties, this study proposes a State Frequency Memory (SFM) which is a RNN based on DFT. The memory cell hidden states are broken down by the SFM divided into multiple frequency components, each representing a distinct frequency of hidden trading patterns that drive stock price volatility. A nonlinear combination of these components is used to forecast future stock values using an IFT.

While short-term projections are generally based on trading patterns where high-frequency is observed. On the other hand, long-term projections are be based on low-frequency trading patterns.

Although, there is no method in the literature that clearly distinguishes between different frequencies of trading patterns in order to generate dynamic forecasts. Explicitly identifying and differentiating different frequencies of hidden trade patterns play a key role in developing price forecast for diverse time steps.

While there exist numerous techniques for forecasting stock prices, none of the available models explicitly deconstruct trading strategy patterns into multiple frequency components and incorporate the identified multi-frequency patterns into price forecasts, to our knowledge.

The frequency domain of input signals is found via the Discrete Fourier Transform in signal processing (DFT). It breaks down input signals into various frequency components in order to investigate their periodicity and volatility. This idea will be used to dynamically deconstruct latent trading patterns states into multiple frequency components, which will fuel SFM's growth over time. The SFM has been impacted by the LSTM's gating design and multi-frequency decomposition.

2.10.2 Research Methodology

2.10.2.1 Dataset

For the experiments, day-to-day open prices of 50 companies across 10 different sectors from year 2007 to 2016 have been collected from Yahoo Finance. Corporations with the top five market capitalizations in each industry are chosen. Before 2007, all of the companies were publicly traded. For certain stocks, there have been multiple share splits in the past. In this case, the prices have been adjusted based on the most recent split. The historical prices have a

2518-day period. Daily prices from 2007 to 2014 were used to train the models for a total of 14 days. Validation and testing are done using daily pricing from 2015 and 2016. Both the validation and test data is composed of 252 seconds. We anticipate the models to learn broad market trends over time based on the prices of 50 businesses during the last 10 years.

2.10.2.2 Price Prediction

Consider a stock's trading prices over time, $\{p_t \mid t = 1, 2,..., T\}$. The objective is to create a n-step forecast on $\mathbf{p_{t+n}}$ according to prices till $\mathbf{p_t}$, given n >= 1.

The forecast on n-step price $\mathbf{p_{t+n}}$ at the timestep t + n may be viewed as a equation given prices $\{p_t \mid t = 1, 2,..., T\}$.

$$\hat{p}_{t+n} = f(p_t, p_{t-1}, ..., p_1)$$

where f indicates the model mapping from historical prices to n-step forward prices.

Because the price scales change for various stocks, we normalize the prices $\{p_t \mid t = 1, 2,..., T\}$ per stock into $\{v_t \mid t = 1, 2,..., T\}$ within the [-1, 1] range without losing generality.

A RNN such as the LSTM or SFM is used, as a mapping for n-step prediction. At each step, the normalized price $\mathbf{v_t}$ is used as an input, with the initial latent/hidden state $\mathbf{h_0}$ and any memory state set to zero. The selected RNN version generates a hidden vector $\mathbf{h_t}$, which is utilized to forecast prices. We apply a matrix transformation to the hidden vector on the n-step prediction:

$$\hat{v}_{t+n} = \boldsymbol{w_p h_t} + b_p$$

where $\mathbf{w_p}$ stands for weight vector and $\mathbf{b_p}$ stands for bias The nonlinearity in this approach is due to the $\mathbf{h_t}$ which is the nonlinear hidden vector, despite the fact that this is a linear transformation.

This architecture varies from that of the AR model that predicts $\mathbf{p_{t+n}}$ using immediately previous data utilizing $\{ \mathbf{p_{t-r}} \mid 0 <= \mathbf{r} <= \mathbf{w} \}$ where **w** is the sliding window to forecast $\mathbf{p_{t+n}}$.

While increasing the sliding window can assist to integrate the long-term context of equity prices, it also adds to the model's complexity, raising the danger of overfitting onto previous values.

This issue does not occur in the SFM or LSTM because inherently, the memory state is capable of maintaining long-term dependencies across input prices.

Prices from numerous stocks are utilized to train the regression network in order to understand broad trading trends in the stock market. The model is then trained with M stocks while minimizing the sum of square errors in the training set between true normalized prices and predicted prices:

$$\mathcal{L} = \sum_{m=1}^{M} \sum_{t=1}^{T} (v_{t+n}^{m} - \hat{v}_{t+n}^{m})^2$$

The BPTT method is used to update all of the model parameters. In this approach, the model's weights are distributed evenly among all stocks.

Any RNN variation can be used to replace the RNN layer in the regression network. For comparison, we use the LSTM and SFM instead of the RNN. Intrinsic memory states are preserved in both designs to explain the core components of trading patterns till time t. With the aid of memory gates, these internal variables are meant to represent the dependency between stock prices.

Furthermore, the SFM's memory states split states into numerous frequency components. Within the memory cell, multi-frequency patterns are summed. It generates the hidden state $\mathbf{h_t}$ by combining these frequency memory components, which may be utilized to forecast future prices.

2.10.3 Results

The proposed SFM network has been compared with the existing state-of-the-art methods, such as the Autoregressive(AR) model and traditional LSTM, in experiments.

The AR model is a time-series forecasting approach that has been around for a long time. It forecasts future market prices based on the prices of previous stages. Its forecast is formed as

a linear combination of previous prices exposed to a Gaussian noise component. The number of prior prices used in the prediction is specified by the AR model's parameter w, which reflects the degree of reliance on historical data. By minimizing the AIC or BIC, the optimal w may be obtained. An AR model, in contrast to the suggested approach, is a linear regression model that can only represent stationary time-series. We hope to see if SFM can represent non-stationary and non-linear dependence, along with multi-frequency patterns in the stock market, by comparing it to the AR model.

We also provide a comparison to the traditional LSTM. It simply records the past prices' internal dependencies. The efficacy of the SFM can be assessed in investigating trade patterns with varied frequencies to forecast stock prices by comparing it to the traditional LSTM. The same technique is used to teach both LSTM and SFM. The models are trained using the RMSprop optimizer, with a fixed learning rate of 0.01. Orthogonal initializer and Xavier uniform initializer respectively initialize the weights U and W. The bias vectors b are set to zero by default. The weights and bias are updated every iteration using all of the training sequences. After 4K iterations, both models converge.

To evaluate the performances, we used the daily mean square error per stock as our assessment criterion. The 1-step forecast, which is a short-term prediction, suggests the trend of the next day. The half-week trend is indicated by the three-step forecast. Because the stock market is only open on weekdays, a 5-step prediction suggests the next week's trend. To put it another way, a 5-step forecast usually spans one weekend, which is a considerably more difficult assignment.

	1-step	3-step	5-step
AR	6.01	18.58	30.74
LSTM	5.93	18.38	30.02
SFM	**5.57**	**17.00**	**28.90**

Table 2.10.3.1 - mean square error of models across multiple steps (Zhang et al., 2017)

Both the LSTM and AR models are outperformed by the SFM, whereas the LSTM model beats the AR model. Each model's performance degrades as we increase the prediction step, as we expected. The SFM, on the other hand, degrades more gradually compared to the other two types. This is due to its capacity to simulate multi-frequency trading patterns in a price time-series environment that has been modified.

The number w of previous prices utilized in the AR model may be thought of as the size of the price data memory. Experiments using the AR model of various types are also carried out. No significant variations in test error is observed when the number of w increases. In fact, as **w** grows greater, the test error gets worse. This might be due to the addition of more parameters to the model, which leads to overfitting of the price data training sequence. The LSTM and proposed SFM, on the other hand, have a smaller test error than the AR model, indicating that they are better at capturing long-term relationships while avoiding the AR model's overfitting issue.

The SFM provides a more exact prediction when trade patterns with various frequencies are involved, as opposed to the LSTM, which solely models internal dependencies. It's worth noting that the frequency components account for trade operations that occur at various rates and cycles. The SFM filters the unnecessary frequency components and maintains the important ones while regulating information from multiple frequencies with gate units to offer direction for future trend prediction. By restricting the model to the local pattern of price fluctuations, it avoids states from dominating the price projection.

The three models can accurately predict stock price trends. The SFM's error curves are smaller than those of the LSTM and the AR model, as predicted. Furthermore, the AR has a tendency to merely replicate the price trend of the past with a significant delay. When the stock price moves abruptly, this results in an extremely low forecast accuracy. The LSTM and SFM, on the other hand, outperform the AR model in the short term because they are less affected by local price changes.

The planned SFM network's complexity has a direct influence on its performance. With a sufficiently sophisticated model, the training error can theoretically be arbitrarily low. A low test error does not always indicate a low prediction error on the training set.

Overfitting is more likely with a complicated model. Two metrics that may be used to measure the SFM's complexity are the number of states and the number of frequencies. It's worth mentioning that the appropriate hyper-parameters for model complexity are chosen depending on the performance of validation set. In our tests, we learn the model with prices from 2007 to 2014 and then validate it with prices from 2015.

# of states	10	20	40	50
1-step	**5.93**	6.60	6.91	6.89
3-step	**18.38**	19.05	18.60	18.68
5-step	30.30	30.58	**30.02**	31.07

Table 2.10.3.2 - In LSTM, the mean square error vs. # of states(Zhang et al., 2017)

The LSTM's complexity is a function of the number/dimension of states. In contrast, the number of states in the SFM refers to the row count in the memory state matrices, suggesting the number of patterns predicted to emerge from the price time-series. As the network learns more predicted patterns, the model grows more complex. With 10-dimensional memory states, the LSTM obtains the highest performance for 1-step and 3-step prediction. With the growing number of states, we see overfitting. Increasing the number of states will no longer reduce the test error. Worse, in LSTM, a large number of states might exacerbate the test error, resulting in an excessive fit into the training cost. The LSTM mixes the information of all frequencies, including the unneeded frequency components, without deconstruction and regulation of information over multiple frequencies. As a result, by trapping the model in the local pattern, it tends to dominate price prediction. Because the 5-step prediction is more difficult, the LSTM requires more states (40) to obtain the best results.

# of states	10	20	40	50
1-step	**5.57**	5.79	6.15	5.91
3-step	18.48	19.20	17.25	**17.00**
5-step	29.48	29.84	31.30	**28.90**

Table 2.10.3.3 - In SFM, the mean square error vs. # of states with frequencies hardcoded to ten. (Zhang et al., 2017)

The overfitting problem is mitigated by incorporating state decomposition and control, as opposed to the LSTM. The SFM, in particular, performs best with 50 states for the 3-step and 5-step. Even though the effectiveness of 1-step prediction degrades as the number of states increases, the SFM degenerates gradually compared to LSTM.

# of frequencies	5	10	15	20
1-step	6.69	5.91	5.91	**5.88**
3-step	18.39	**17.00**	19.15	19.52
5-step	30.95	**28.9**	30.57	31.22

Table 2.10.3.4 - In SFM, the mean square error is shown against the # of frequencies with states set at fifty. (Zhang et al., 2017)

The number of frequencies, in addition to the states, is an essential hyperparameter in the SFM. On [0, 2π], the frequencies are equally distributed. It is obvious that as the frequency of the memory states increases, so does the amount of information on the high frequency in

short-term memory. In addition, the components of high frequency trading are used to account for short-term trading activity. It demonstrates the SFM's ability to mimic high-frequency trading in the stock market. For 3-step and 5-step prediction, the best performance is obtained with ten frequencies. Because their volatility is smaller than the 1-step prediction, components with low frequency become more important to the forecast.

2.11 Enhancing Stock Movement Prediction with Adversarial Training(Feng et al., 2019)

2.11.1 Business Problem

This study presents a new ML approach for stock price forecasting, with the goal of predicting whether a stock's price will rise or fall in the near future. Adversarial training can be used to increase the generalization of a NN based time series prediction model, which is a unique approach. The logic of adversarial training in this case stems from the fact that stock prediction input characteristics are generally dependent on raw stock price data, which basically is a stochastic variable that changes over time. As a result, traditional price-based training (e.g., the close price) can easily overfit the data, making it impossible to generate trustworthy models. To solve this challenge, we suggest using perturbations to replicate the stochasticity of pricing variables and training the model to perform well under tiny but deliberate perturbations. Extensive trials on two real-world stock data demonstrate that our technique beats the state-of-the-art solution(Xu & Cohen, 2018) by 3.11% on average in terms of accuracy, demonstrating the utility of adversarial training for stock prediction tasks.

2.11.2 Research Methodology

Datasets. The proposed approach is evaluated against two stock movement prediction benchmarks: ACL18(Xu & Cohen, 2018) and KDD17(Zhang et al., 2017)

ACL18 provides historical data for 88 high-trade-volume equities in the NASDAQ and NYSE marketplaces from January 1, 2014 to January 1, 2016. We first align the trade days in the past, eliminating weekends and public holidays that lack historical pricing, as recommended by(Xu & Cohen, 2018). We next create candidate instances each with a lag with a duration of T along the aligned trade days (i.e., one example for a stock on every trading day). The potential instances are labelled based on the percent change in stock close prices. Positive and negative instances with movement of 0.55 percent and 0.5 percent, respectively, are discovered.

KDD17 contains a lengthier history of 50 equities in US markets from January 1, 2007 to January 1, 2016. We use the same method as ACL18 to distinguish positive and negative examples because the information was initially gathered to forecast stock prices rather than movements. The instances were then divided into three categories: training (January 1, 2007 to January 1, 2015), validation (January 1, 2015 to January 1, 2016), and testing (January 1, 2016 to January 1, 2017).

Features	Calculation
c_open, c_high, c_low	*e.g., c_open* $= open_t/close_t - 1$
n_close, n_adj_close	*e.g., n_close* $= (close_t/close_{t-1} - 1$
5-day, 10-day, 15-day, 20-day, 25-day, 30-day	*e.g., 5-day* $= \frac{\sum_{i=0}^{4} adj\_close_{t-i}/5}{adj\_close_t} - 1$

Table 2.11.2.1: temporal features to explain the trend of a stock(Feng et al., 2019)

We create 11 temporal characteristics ($\mathbf{x^s_t}$) to explain the trend of a stock s at trading day t instead of utilizing raw EOD data. The features related with computation are detailed in Fig 2.11.2.1. The purpose of establishing these characteristics is to:

1) equalize the prices of various stocks

2) record the different prices interactions clearly

- MOM is a statistical indicator that forecasts whether a sample which follows a trend in the past 10 days will be negative or positive.

- MR forecasts each example's movement in the opposite direction of the most recent price towards the 30-day moving average.

- LSTM is a neural network containing an LSTM layer and a prediction layer. Number of hidden units (U), lag size (T), and weight of regularization term (W) are three hyperparameters that we tune.

- The Attentive LSTM(Qin et al., 2017) is a optimized version of LSTM. We tune U, T, and in the same way as we tune LSTM.

- StockNet(Xu & Cohen, 2018) employs a Variational Autoencoder (VAE) to encode the stock input and a temporal attention model to represent the relevance of distinct

timesteps. Here, we use the features from Fig 2.3.2.1 as inputs to optimize the hidden size, dropout ratio, and auxiliary rate (α).

We use two measures, Accuracy and MCC (Xu & Cohen, 2018), to assess prediction performance, with ranges of [0, 100] and [1, 1]. The greater the value of the measures, the better the performance.

We use Tensorflow to construct the Adv-ALSTM and then perform optimization by the mini-batch Adam, with an initial learning rate of 0.01 and a batch size of 1024. On the validation set, we look for the best Adv-ALSTM hyper-parameters.

Adv-ALSTM inherits the optimal parameters from ALSTM for U, T, and λ, which are chosen through grid-search within hyperparameter space. We then fine-tune β and ε. Over five separate runs, we present the mean testing performance when Adv-ALSTM performs best on the validation set.

2.11.3 Results

Method	ACL18		KDD17	
	Acc	MCC	Acc	MCC
MOM	47.01±—	-0.0640±—	49.75±—	-0.0129±—
MR	46.21±—	-0.0782±—	48.46±—	-0.0366±—
LSTM	53.18±5e-1	0.0674±5e-3	51.62±4e-1	0.0183±6e-3
ALSTM	54.90±7e-1	0.1043±7e-3	51.94±7e-1	0.0261±1e-2
StockNet	54.96±—	0.0165±—	51.93±4e-1	0.0335±5e-3
Adv-ALSTM	**57.20**±—	**0.1483**±—	**53.05**±—	**0.0523**±—
RI	4.02%	42.19%	2.14%	56.12%

RI denotes the relative improvement of **Adv-ALSTM** compared to the best baseline.
The performance of **StockNet** is directly copied from [Xu and Cohen, 2018].

Table 2.11.3.1 - Model Comparison against suggested ALSTM(Feng et al., 2019)

- In every situation, Adv-ALSTM produces the best outcomes. On the ACL18 (KDD17) dataset, Adv-ALSTM improves Accuracy and MCC by 4.02 percent and 42.19 percent (2.14 percent and 56.12 percent, respectively) when compared to the baselines. This explains adversarial training's success, which may be attributable to improved model generalization by adaptively modelling disturbances during training.

- Specifically, when compared to StockNet, which uses VAE to capture stochasticity in stock inputs, Adv-ALSTM outperforms StockNet. The reason for this, we believe, is

that during the training process, since StockNet depends on Monte Carlo sampling, it is unable to explicitly describe the size and direction of stochastic perturbation.

- In terms of Accuracy and MCC, ALSTM beats LSTM by 1.93 percent and 48.69 percent on average, demonstrating the importance of attention(Qin et al., 2017). Furthermore, as predicted, MR and MOM outperform all ML-based approaches, demonstrating that historical patterns aid in stock prediction.

We compare adversarial perturbations to random perturbations to see how successful adversarial training is. Rand-ALSTM is a variant of Adv-ALSTM that adds random disturbances/noise to the clean input examples to create new instances.

Datasets	Acc	MCC
ACL18	55.08±2e0	0.1103±4e-2
KDD17	52.43±5e-1	0.0405±8e-3

Table 2.11.3.2 - Compare effect of Adversarial Training(Feng et al., 2019)

When compared to RandALSTM, Adv-ALSTM improves significantly. For example, it outperforms Rand-ALSTM by 3.95 percent when it comes to Accuracy on ACL18. It shows that adversarial perturbations are useful for stock prediction, just as they were in the original picture classification challenges(Goodfellow et al., 2013). On the two datasets, Rand-ALSTM beats ALSTM, where only clean input examples were used during training, with an average improvement of 0.64 percent vs. Accuracy. This emphasizes the need of dealing with stock characteristics' stochastic properties.

2.12 Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction(Ding et al., 2020)

2.12.1 Business Problem

To solve the stock movement prediction challenge, we present a unique method based on the Transformer in this article. In addition, we provide a number of improvements to the basic Transformer that has been presented. To begin, we suggest a Multi-Scale Gaussian Prior to improve Transformer's locality. Second, in the self-attention multi-head mechanism, we create an Orthogonal Regularization to prevent learning redundant heads. Finally, in order to

understand hierarchical characteristics of high-frequency financial data, we create a Trading Gap Splitter for Transformer. The suggested technique has the benefit of mining highly long-term dependencies from financial time series as compared to other prominent recurrent neural networks such as LSTM.

2.12.2 Research Methodology

Because predicting the precise price of a stock is exceedingly difficult, we use [Walczak, 2001]'s setup and anticipate price change instead. Typically, stock price forecasting is approached as a problem of binary classification, with the stock movement being divided into two cases i.e. Fall or Rise.

Given the parameters of the stock $\mathbf{X} = [\mathbf{x}_{T-\Delta t+1}, \mathbf{x}_{T-\Delta t+2}, \ldots, \mathbf{x}_T] \in \mathbf{R}^{\Delta t \times F}$ in the most recent time-periods $\Delta\mathbf{t}$, the prediction model $\mathbf{f}_\theta(\mathbf{X})$ with θ features predict the price movement label $\mathbf{y} = \mathbf{I}(\mathbf{p}_{T+k} > \mathbf{p}_T)$, with target trading time **T**, stock feature dimensions **F**, and $\mathbf{p}_T$ denotes the closing time at **T**. To summarize, the suggested model uses past data from stock **s** in the **lag [T - Δt + 1, T]** (with predetermined lag size Δ**t**) to forecast the price movement class **y** (1 for Rise, 0 for Fall) of next k time-periods.

We utilize two stock data sets to test the suggested methods: NASDAQ and China A-shares market. Table 2.12.2.1 shows the specifics of the two data sets. We will describe the data gathering procedure and present our empirical results from numerical experiments in the subsections that follow. We also do an incremental analysis to determine the efficacy of each suggested Transformer improvement.

Property	Data	
	Daily	15-min
Market	NASDAQ	China A-shares
Start Date	2010/07/01	2015/12/01
End Date	2019/07/01	2019/12/01
Time Frequency	1 day	15 minutes
Total Stocks	3243	500
Total Records	9749098	7928000
Rising Threshold β_{rise}	0.55%	-0.5%
Falling Threshold β_{fall}	-0.1%	0.105%

Table 2.12.2.1 - Dataset properties(Ding et al., 2020)

From July 1st, 2010 to July 1st, 2019, we collected daily quotation data for all 3243 stocks on the NASDAQ stock market, as well as 15-min quotation data for 500 CSI-500 stocks on the China A-shares market starting from December 1st, 2015 till December 1st, 2019. To create candidate examples, we slide a lag window with size of N time periods across this time series data. For the NASDAQ data, we create five datasets with different lag window widths N = 5, 10, 20, and 40 (representing five, ten, twenty, and forty days), and the lag strides are all set to one. We also create 5 datasets for China A-shares data with window sizes N = 5 * 16,10 * 16, 16 * 20, 40 * 16 (denoting five, ten, twenty, and forty days because one trading day in China A-shares market contains 16 15-minute), and the lag strides for all of them set as 16 (denoting 5-, 10-, 20-, 40-day because one trading day in China A-shares market contains 16 15-minute), and the log strides for all of them set as 16 (i.e. 1 day). Open, high, low, near, and volume are the characteristics utilized in all datasets, and they are all modified and standardized. We use the approach $\mathbf{y = I(p_{T+k} > p_T)}$, where we set k between 1 and 16 (both represent 1 day). Furthermore, we applied two threshold parameters to the labels in both datasets, β_{rise} and β_{fall}, to ensure that there is no class imbalance in each dataset:

$$y = \begin{cases} 1 & \frac{p_{T+k} - p_T}{p_T} > \beta_{rise}; \\ -1 & \frac{p_{T+k} - p_T}{p_T} < \beta_{fall}; \\ \text{abandon} & \text{otherwise.} \end{cases}$$

To eliminate data leaking, we separate training, validation, and test sets in a tight sequential manner.

Method	Accuracy (%)/MCC ($\times 10^{-2}$) with window size of K-day			
	$K = 5$	$K = 10$	$K = 20$	$K = 40$
NASDAQ Daily Data				
CNN	52.33/3.16	52.02/2.68	51.84/2.28	52.60/2.52
LSTM	53.86/7.73	53.89/7.72	53.59/7.15	53.81/7.48
ALSTM	54.06/8.35	53.94/7.92	54.05/8.11	54.19/8.56
B-TF†	54.78/8.48	54.84/8.89	54.90/9.13	56.01/9.45
MG-TF†	**55.10/8.98**	**56.18/9.74**	**56.77/10.39**	**57.30/11.46**
China A-shares 15-min Data				
CNN	53.53/2.62	52.25/1.80	52.03/1.81	51.61/1.77
LSTM	56.59/6.42	56.70/6.19	56.18/3.74	54.93/2.98
ALSTM	57.03/8.23	57.42/9.16	55.69/6.65	55.68/6.65
B-TF†	57.14/9.68	57.42/9.52	57.32/9.14	57.55/10.41
HMG-TF†	**57.36/10.52**	**57.79/9.98**	**57.90/10.33**	**58.70/14.87**

Table 2.12.2.2 - Model Comparison against suggested Transformers(Ding et al., 2020)

We use two measures to assess prediction performance: accuracy and MCC

$$\mathrm{MCC} = \frac{\mathrm{tp} \times \mathrm{tn} - \mathrm{fp} \times \mathrm{fn}}{\sqrt{(\mathrm{tp} + \mathrm{fp})(\mathrm{tp} + \mathrm{fn})(\mathrm{tn} + \mathrm{fp})(\mathrm{tn} + \mathrm{fn})}}$$

B-TF : B-TF is an encoder-only Transformer variation with L blocks of multi-head self-attention layers along with position-wise feed forward layers.

MG-TF : Multi-Gaussian Prior and Orthogonal Regularization improvements to the B-TF model

HMG-TF : Trading Gap Splitter has been added to the MG-TF model.

The Adam optimizer is used to implement the solution, with a learning rate of 1e - 4 as a starting point. The mini-batch size is set as 256 and the trade-off hyper-parameter is set at 0.05. Each of the three multi-head self-attention blocks in the TF-based models has four heads. We train the model from start to finish using only raw quotation data and no data augmentation. We exclusively use MG-TF on our NASDAQ dataset since Trading Gap Splitter is less suitable for low-frequency data like daily quotation data.

2.12.3 Results

In all situations, the suggested methods B-TF, MG-TF, and HMG-TF demonstrate considerable increases on both the Accuracy and MCC measures when compared to baselines. It demonstrates that Transformer-based techniques outperform RNN- and CNN-based approaches in terms of performance.

The ability to understand long-term dependencies is better with transformer-based methods. It is particularly difficult for RNN-based methods to discover the interdependencies across a large number of time steps on China A-shares data, where the window of 40 days comprises 640 (40*16) time-periods. While the benefits of the self-attention mechanism allow Transformer-based methods to continuously outperform other approaches as the window grows larger.

The modified Transformer models MG-TF and HMG-TF outperform the standard Transformer model.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

The main objective of this study is to first build our portfolio with high quality assets chosen based on various features extracted from raw price (OHLCV) data. Based on the momentum of the assets, we can choose the winning stocks and bet on the momentum effect and/or go with the losing stocks if we see a reversal effect trend based on historical data. Once the stocks have been finalized for our portfolio, we will evaluate and compare the DL models for price forecasting based on returns/profits achieved. The same models will be built using standard market indexes (NIFTY50, SENSEX) in order to answer our research question i.e. “Does the quality of financial assets matter if we keep the base forecasting model performance as constant?”

Fig 3.1.1. depicts the research methodology on a step-by-step manner

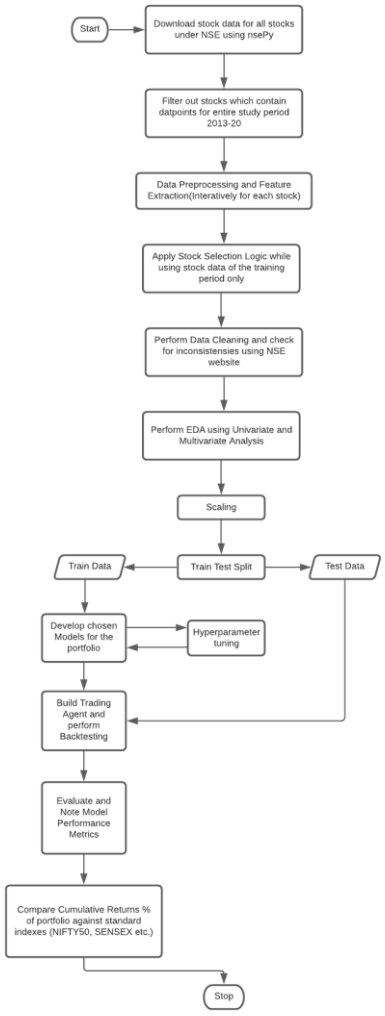


Fig 3.1.1 – Research Methodology

3.2 Dataset Description

The data is downloaded using nsepy package, an open source python package that offers various methods/functions to download financial data from the official NSE website. It also provides some daily financial information like daily top gainers/losers.

The top 10 equities from each sector from the Indian Stock Market will be utilized as the dataset for equites. We will be using the raw price data(OHLCV) to derive/extract more features which can play a significant role while deciding on the stocks for our investment portfolio.

Column Summary:

Base Attributes from dataset :

Date	Date of the record
Open	Opening Price
High	High for the day
Low	Low for the day
Close	Closing Price
Volume	Total Volume

Table 3.2.1 - Base Attributes

3.3 Data Preparation and Pre-Processing

The selected ranges of training and testing data encompassed at least one entire market cycle in order to assess the models' performance in diverse market conditions (i.e., bullish, bearish, and oscillating markets). The following are the key pre-processing processes that will be followed at this stage of the study:

- Check for missing/NaN values and treat them if necessary.
- Filter out stocks with less than required data-points based on the time window.
- Perform some basic Exploratory Data Analysis to get an idea of overall trend

3.4 Feature Extraction and Engineering

Since selection of quality assets for our portfolio is an important requirement for this study, we will be applying Feature Engineering to generate new derived features which can be potentially helpful in selection of financial assets qualitatively.

Financial data is unique and needs a methodical approach. First, in compared to datasets used in applications of ML, it is fairly restricted in availability and breadth — in fact, we don't have high-quality data for the great majority of markets prior to the 1990s. Many crucial factors that drive results can be left out of a model or don't get to see the entire range of their values during training. The significantly divergent behaviour of asset prices during high and low market volatility periods, or regimes, is a specific illustration of this difficulty; training the model on a subset of the data that spans only one of the regimes will degrade the model's capacity to generalize well on the test set. The signal-to-noise ratio of financial data is quite low. Models will overfit the data noise because the most powerful models, such as neural networks, which are low bias and high-variance learners. Surely, additional factors such as macroeconomic statistics, sentiment related to monetary policy announcement, and so on might help to describe the overall situation of the market.

Following are some of the derived features which have been identified. This list will be updated and new features may be added during feature extraction and engineering phase of the Research Plan.

Derived Attributes :	
Close Delta	Daily Change in Closing Price
Close Change %	% Daily Change in Closing Price
Market Cap	Daily Market Capitalization
Volume Delta	Daily Change in Volume
Volume Change %	% Daily Change in Volume
Weekly Volatility	Close – Weekly Close Avg.
Monthly Volatility	Close – Monthly Close Avg.
Yearly Volatility	Close – Yearly Close Avg.
Nearness to 52W High	Close – Yearly Close Max
(% positive / % negative) trading days	
% Returns over look back period	
Volatility over look back period	

Table 3.4.1 - Derived Features

3.5 Stock Sortation and Selection Logic

After feature engineering we can now use these new features to sort the stocks and choose the top XX stocks. One thing to note here is that this selection logic will be applied on training period only. The details of this logic is as follows:

Look Back Period Returns > Risk Free Rate	Operation
Stock with Highest momentum(returns %) during look back period	Descending Sort
(% positive / % negative) trading days	Descending Sort
Nearness to 52 week high	Ascending Sort
Volatility or Downside Volatility	Ascending Sort
Top XX Stocks	Filter

Table 3.5.1 - Sortation Logic

3.6 Model Development

Once we have finalized the stock pool which we will be building our portfolio with based on the previously derived features, we will be employing various DL based models to forecast the prices for the upcoming period using the extracted/derived features.

The Models which we have decided upon after the Literature Reviews are as follows:

1. **LSTM (Baseline)**

The LSTM RNN is a kind of RNN that can recall both short and long-term values. When it comes to difficult issues like automated speech recognition and handwritten character identification, many DL model developers use LSTM networks. Time-series data is the most common type of data for LSTM models. Sentiment analysis, Natural Language Processing (NLP), language modelling, speech recognition, language translation, financial time series analysis, predictive analysis and other applications use it. LSTM networks can be even more useful for time series data processing, such as language translation, using attention modules and AE structures.

The LSTM is a kind of RNN that is more specialized. As a result, the recommended optimization methods and weight updates are same. Furthermore, the number of hidden layers, the number of units in each layer, network weight initialization, activation functions, learning rate, momentum values, batch size (minibatch size), the number of epochs, decay rate, sequence length for LSTM, optimization algorithms, gradient normalization, gradient clipping

and dropout are all hyperparameters that are similar to RNN. The hyperparameter optimization approaches used for RNN may also be used to LSTM in order to discover the optimum hyperparameters.

2. **Hybrid CNN-LSTM**

As we have observed from our literature review of (Liu et al., 2017a, p.), their experiments show that the CNN-LSTM neural network model may be successfully used to create quantitative strategies and get greater returns than the Benchmark index.

3.7 Research Hypothesis Formation

Now that we have finalized the prediction models, this study will be aimed at confirming the following hypotheses:

- **H1:** In the prediction of Time Series stock data, Deep Learning-based RNN algorithms perform exceedingly well.
- **H2:** A hybrid technique, such as CNN + LSTM, can improve performance over a stacking LSTM strategy based only on RNNs.
- **H3:** Quality asset selection is equally as important as the prediction algorithm's performance, as well as the trading agent's and environment's efficacy, in maximising return on investment which is evident by the returns achieved by the standard Buy and Hold Trading Strategy.

3.8 Model Evaluation Metrics

The following measures will be used to evaluate all models under consideration:

- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Root Mean Square Error (RMSE)
- Profit/Returns %

The Cumulative Returns % for the portfolio with the top XX stocks will be compared against a portfolio with random stocks or an index like NIFTY, SENSEX etc.

In addition to the indicators listed above, the temporal complexity of the models may be taken into account while assessing them, i.e. the overall amount of features provided, as well as the

model chosen, might result in varying levels of temporal complexity, which is important when the models are deployed into production. Since some business decisions require the model response time to be as low as possible, the time complexity of the model becomes an important factor.

3.9 Resource Requirements

3.9.1 Hardware Requirement:

This study will be carried out on a desktop computer that meets the following requirements:

Processor	Intel® CoreTM i9-9900K CPU @ 5.0 GHz
Memory	32 GB DDR4
GPU	RTX 2080
Operating System	Windows 10 64-bit

Table 3.8.1

3.9.2 Software Requirements:

- Python 3.x
- Jupyter Notebook
- Libraries required for general data analysis and EDA like numpy, pandas, matplotlib
- Libraries required for pre-processing, model selection, feature selection like sklearn
- Libraries required for evaluation metrics
- Any other libraries as and when required

CHAPTER 4 - IMPLEMENTATION & ANALYSIS

4.1 Introduction

This chapter describes the pre-processing steps that will be taken with the data, including how to handle missing data and how to choose financial assets. The data will be subjected to univariate, bivariate, and multivariate analysis. Data visualization will be used to make inferences. This chapter will finish with the selection of financial assets using new derived factors that prioritize not just returns over the lookback period, but also volatility.

4.2 Dataset Description

The data set was created by a script which iteratively downloads data from Yahoo Finance using the yfinance python package. This script downloads data for all stocks which contains daily data between the period of Jan 1 2013 to Dec 31 2020 which totals 1969 datapoints and 646 stocks. We are ingoring any new stocks which got listed during this period and do not have full 1969 datapoints for our analysis. Along with the above data, we are also using India VIX data during Backtesting. The model will be developed as below.

Independent Variables:

- Closing Price data of chosen stocks pertaining to NSE.

Dependent Variables:

- Opening Price
- Low Price
- High Price
- Volume
- Open Delta (Daily change in opening price)
- Change % (Daily opening price change %)
- MarketCap
- Volume Delta (Daily change in Volume)
- Weekly Volatility
- Monthly Volatility
- Yearly Volatility
- 52W High Nearness
- Is Pos/Neg Day
- Lookback Volatility

The following observations were made after reviewing the dataset:

- The Indian stock market is only open throughout the week and is closed on weekends and public holidays. As a result, data for these dates is unavailable.
- It is easy to extract the opening values because the Indian stock market runs throughout set hours from 9:00 to 15:30.

4.3 Dataset Cleaning

Although yfinance is a popular package which has a big user base, standard data cleaning is done to find and fix errors in the dataset that would have a direct influence on the prediction models. To prepare the data, several data cleansing processes must be done in a methodical manner. For modelling purposes, these measures were taken to ensure accurate, full, and consistent data.

- Check for missing and NaN values.
- Check for duplicate records and perform deduplication if necessary.
- Check for data inconsistencies using the Indian NSE website as source of truth.

4.4 Financial Asset Selection

We have chosen stocks based on a well-defined selection criteria which uses the new attributes which were derived during Feature Engineering phase. The objective of this selection criteria is to filter out stocks which are gaining momentum while looking at potential returns as well as volatility over a chosen look back period.

The selection strategy has been applied on a total of 779 stocks with look back period of 22 days (1 month based on active trading days) which is as follows:

1. Apply descending sort on the list of stocks based on the returns over look back period.
2. Apply descending sort on the output of step 1 based on the **pos_over_neg_days**(% positive trading days/% negative trading days)
3. Apply ascending sort on the output of step 2 based on **nearness_to_52w**(Nearness to 52 week high)
4. Apply ascending sort on the output of step 3 based on **volatility_over_lookback_period**

After applying our selection criteria we filtered the following 5 stocks for our analysis:

- KSCL (Food Products)
- SOMANYCERA (Building Products)
- ORIENTBELL (Building Products)
- ESABINDIA (Machinery)
- SMLISUZU (Machinery)

4.5 Exploratory Data Analysis

This section focuses on data analysis using summary statistics and visualizations. The goal is to determine the link between the independent and dependent variables in order to forecast the closing prices of the selected stocks. Various sorts of analysis are carried out, with the results emphasised in the sections below.

4.5.1 Univariate Analysis

Univariate analysis has been used to see how each variable is distributed by drawing line graphs and looking at descriptive statistics.

Univariate analysis has been done for each chosen stock based on our filtration criteria.

Fig – 4.5.1.1 LinePlot for KSCL

Fig – 4.5.1.2 LinePlot for SOMANYCERA

Fig – 4.5.1.3 LinePlot for ORIENTBELL

Fig – 4.5.1.4 LinePlot for ESABINDIA

Fig – 4.5.1.5 LinePlot for SMLISUZU

On the basis of the Line Plots of the chosen Financial Assets, it can be shown that all of the financial assets display time series data properties such as trend and seasonality. Furthermore, by looking at the line plots, it is clear that most financial assets move in a similar way. Although the increase appears to be identical, there is a variation in the volatility noticed.

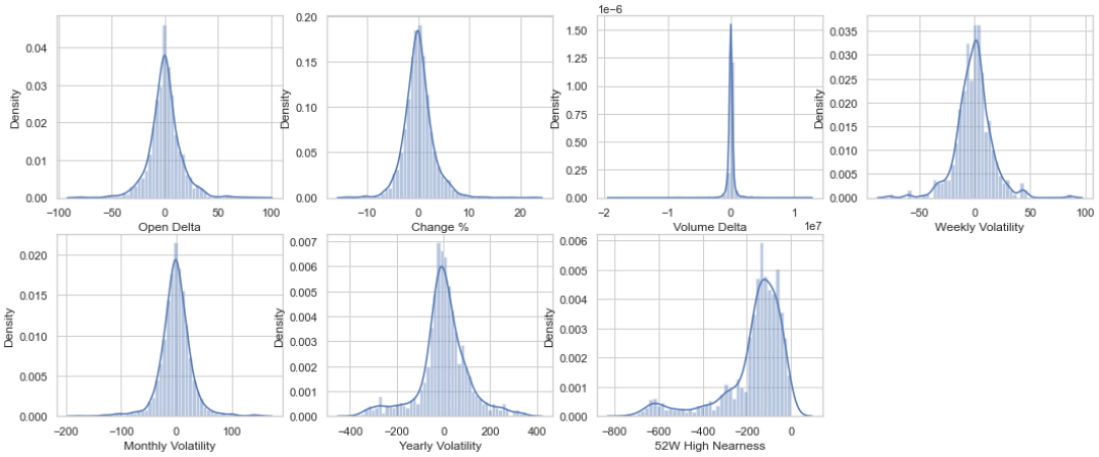


Fig – 4.5.1.6 DistPlots for KSCL

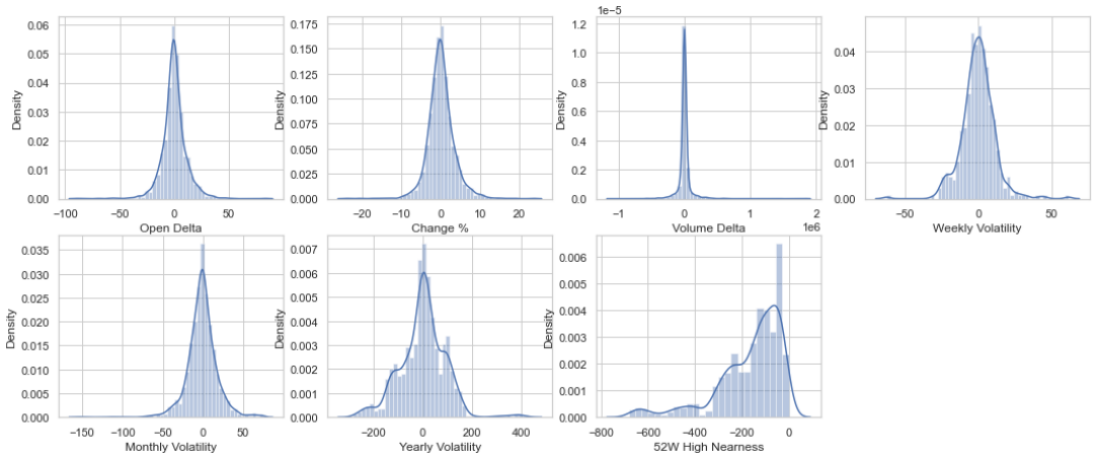


Fig – 4.5.1.7 DistPlots for SOMANYCERA

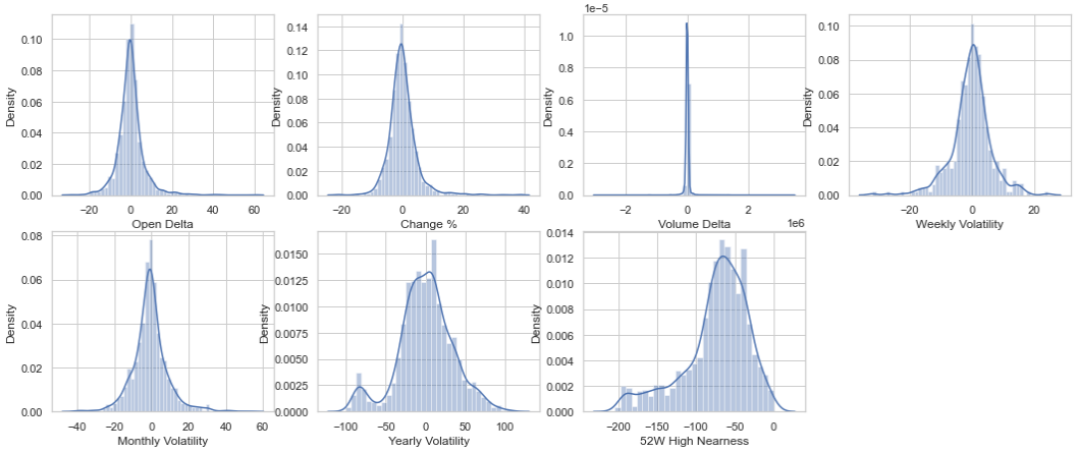


Fig – 4.5.1.8 DistPlots for ORIENTBELL

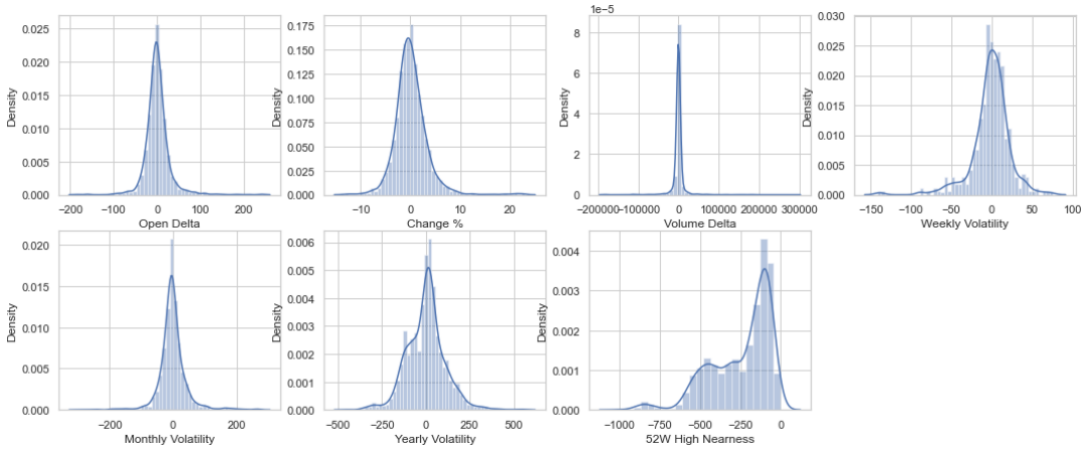


Fig – 4.5.1.9 DistPlots for ESABINDIA

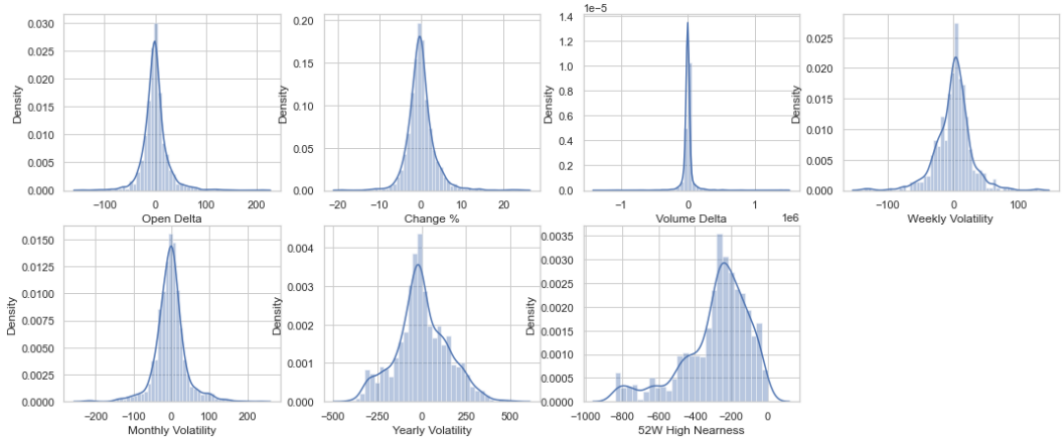


Fig – 4.5.1.10 DistPlots for SMLISUZU

Based on the distplots, it can be shown that the distribution of volatility over week, month, and year follows a normal distribution centred on zero for the selected Financial Assets. One factor to keep in mind is that the 52W High Nearness distribution for KSCL, SOMANYCERA, ORIENTBELL, and ESABINDIA is mostly between 0 and 100, indicating a less volatile distribution during the specified time period. SMLISUZU's 52W High Nearness is slightly greater than others, which range from 100 to 200. Furthermore, the normal distribution of Open Delta and Change percent, which is centred on 0, demonstrates that the daily price movement is within 100, i.e. 10%, indicating low volatility.

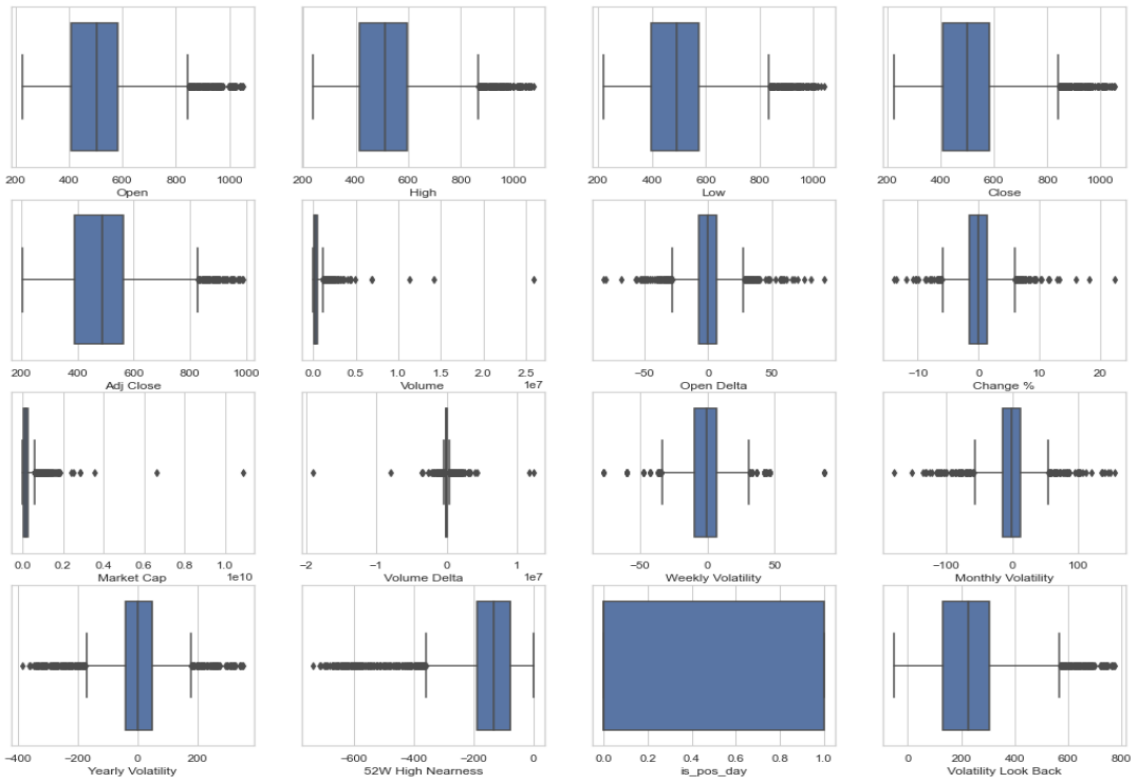


Fig – 4.5.1.11 BoxPlots for KSCL

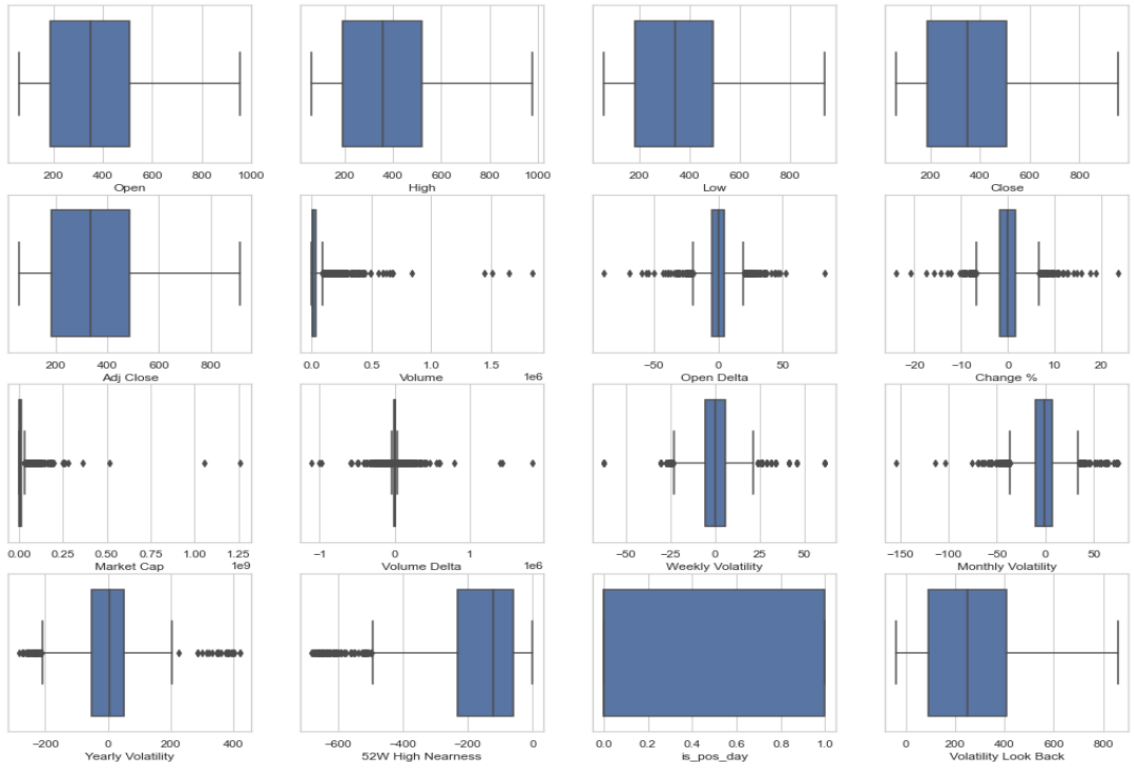


Fig – 4.5.1.12 BoxPlots for SOMANYCERA

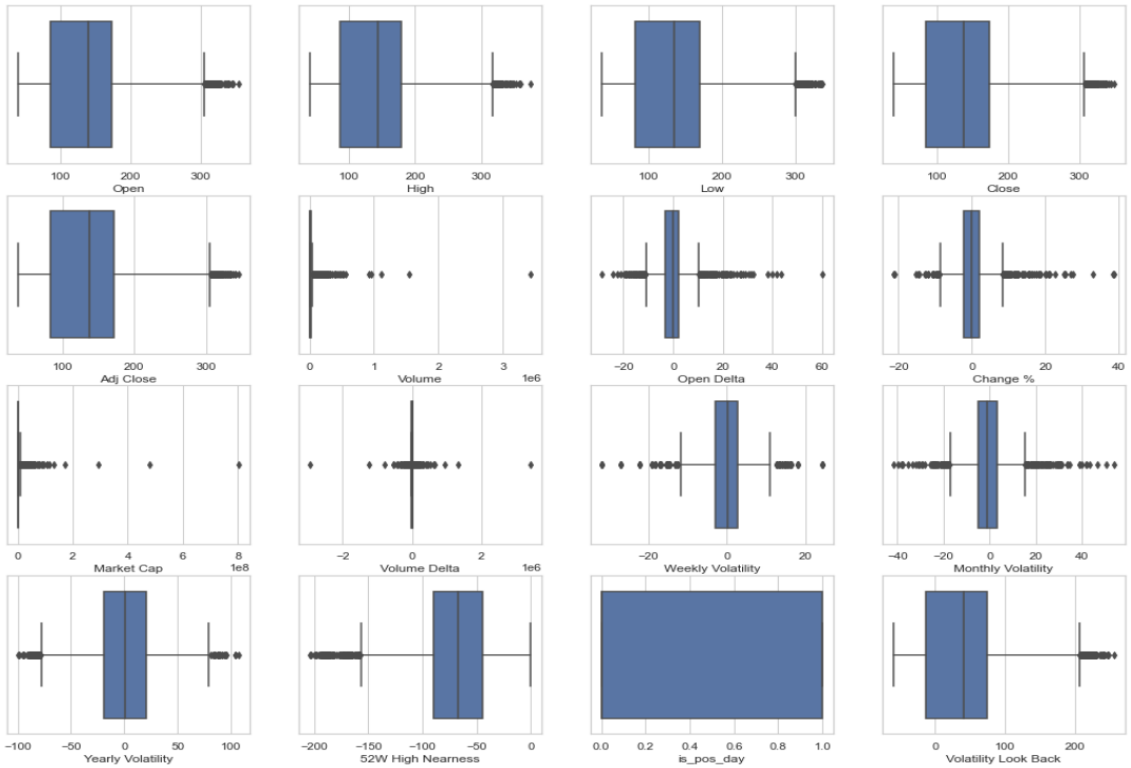


Fig – 4.5.1.13 BoxPlots for ORIENTBELL

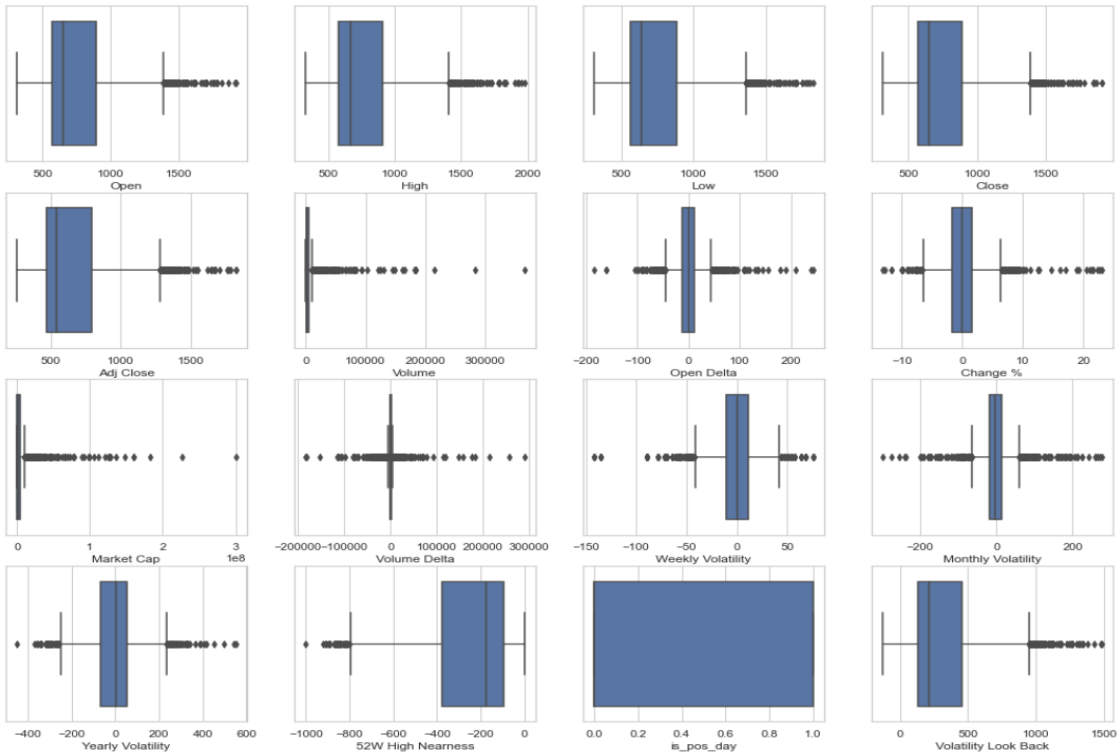


Fig – 4.5.1.14 BoxPlots for ESABINDIA

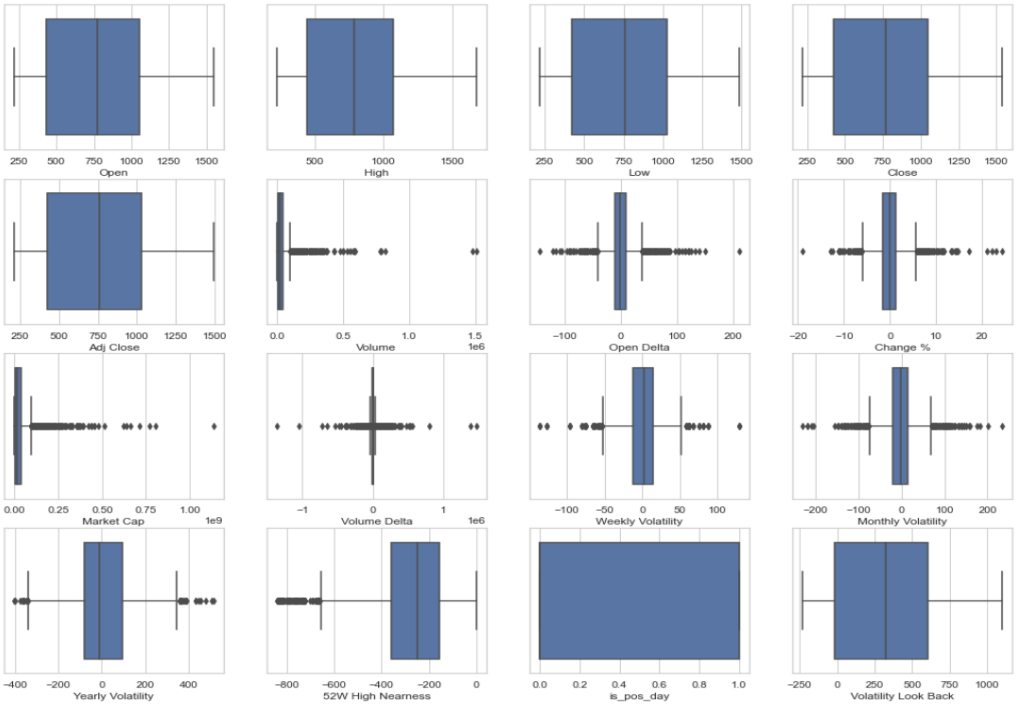


Fig – 4.5.1.15 BoxPlots for SMLISUZU

Based on the boxplots we can observe the presence of outliers for all of our chosen stocks. The cause can be attributed to the time period of our analysis which is 2013-2021 amidst which the COVID pandemic took place, which became the driving cause of a market crash observed during March 2020. But this also came as a boon for investors to enter the stock market as the market started to recover and broke through its previous all-time high before the pandemic by the end of October 2020.

4.5.2 Multivariate Analysis

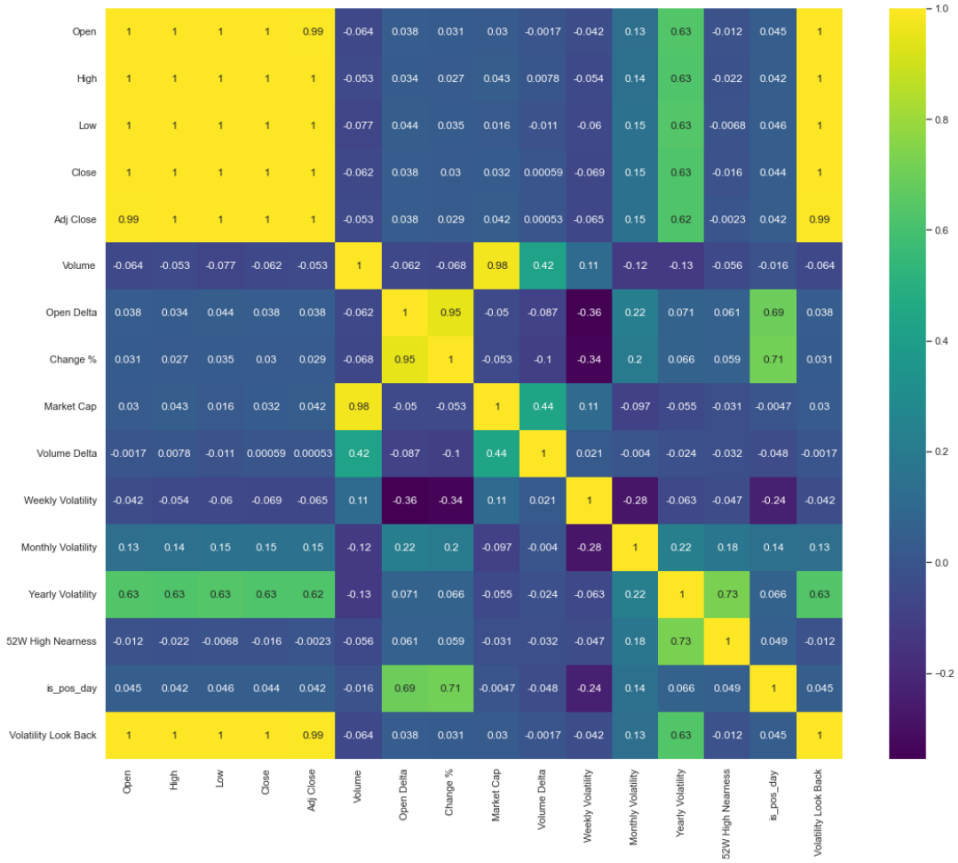


Fig – 4.5.2.1 Correlation Heat map for KSCL

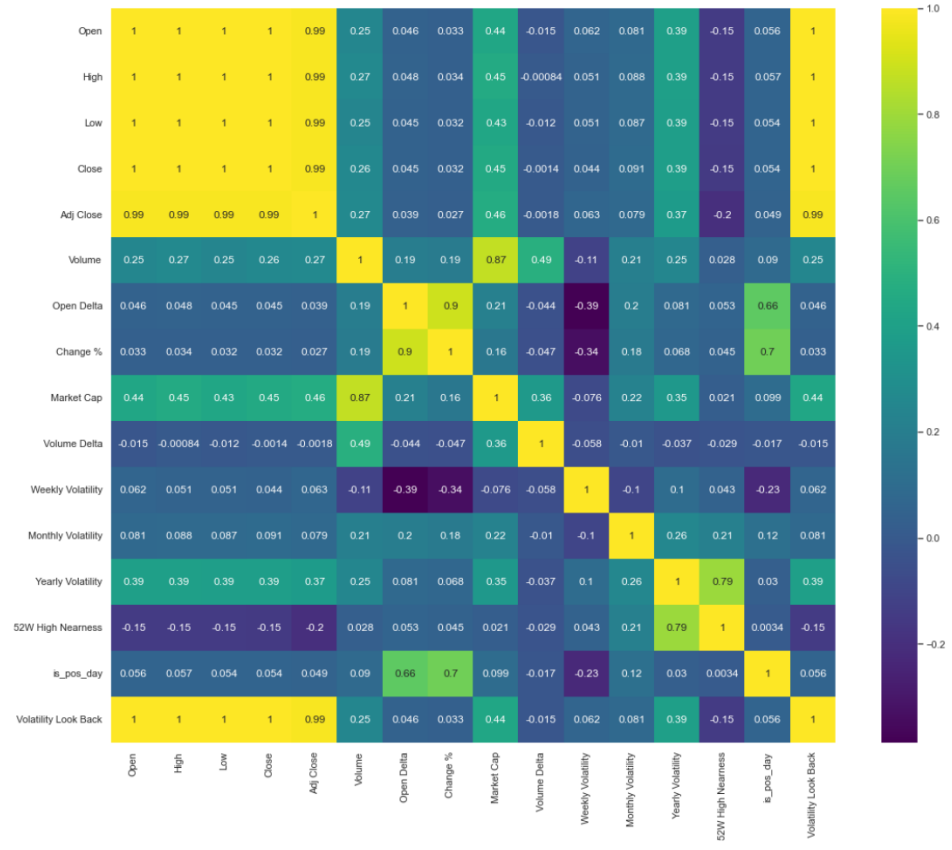


Fig – 4.5.2.2 Correlation Heat map for SOMANYCERA

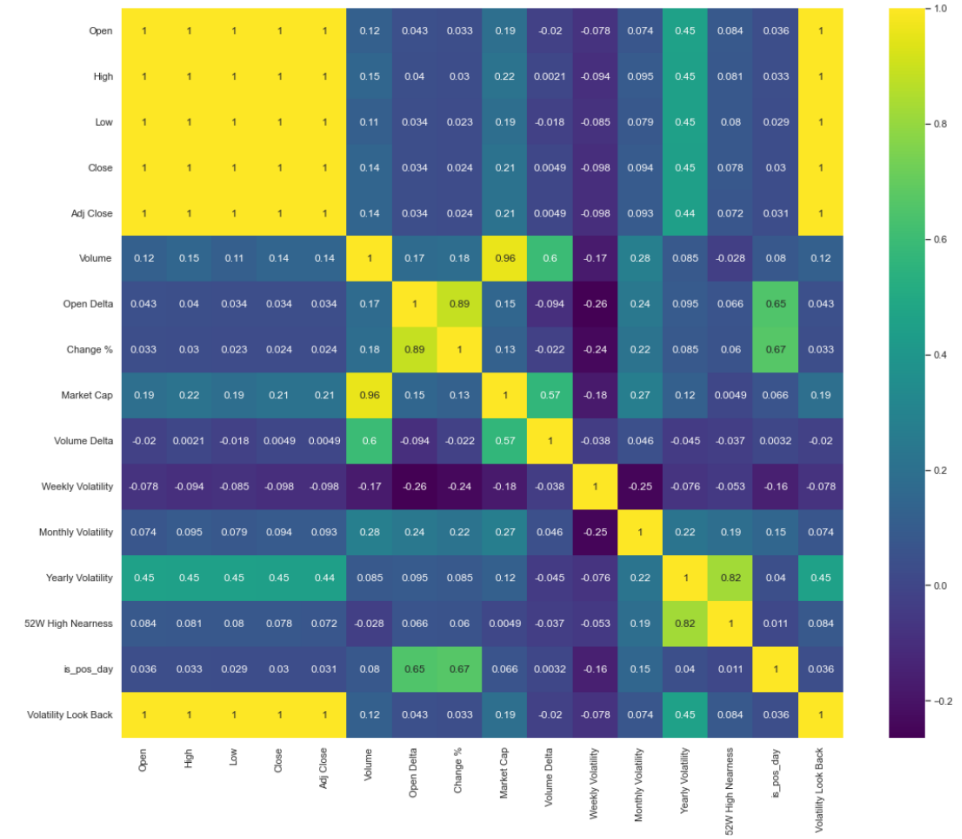


Fig – 4.5.2.3 Correlation Heat map for ORIENTBELL

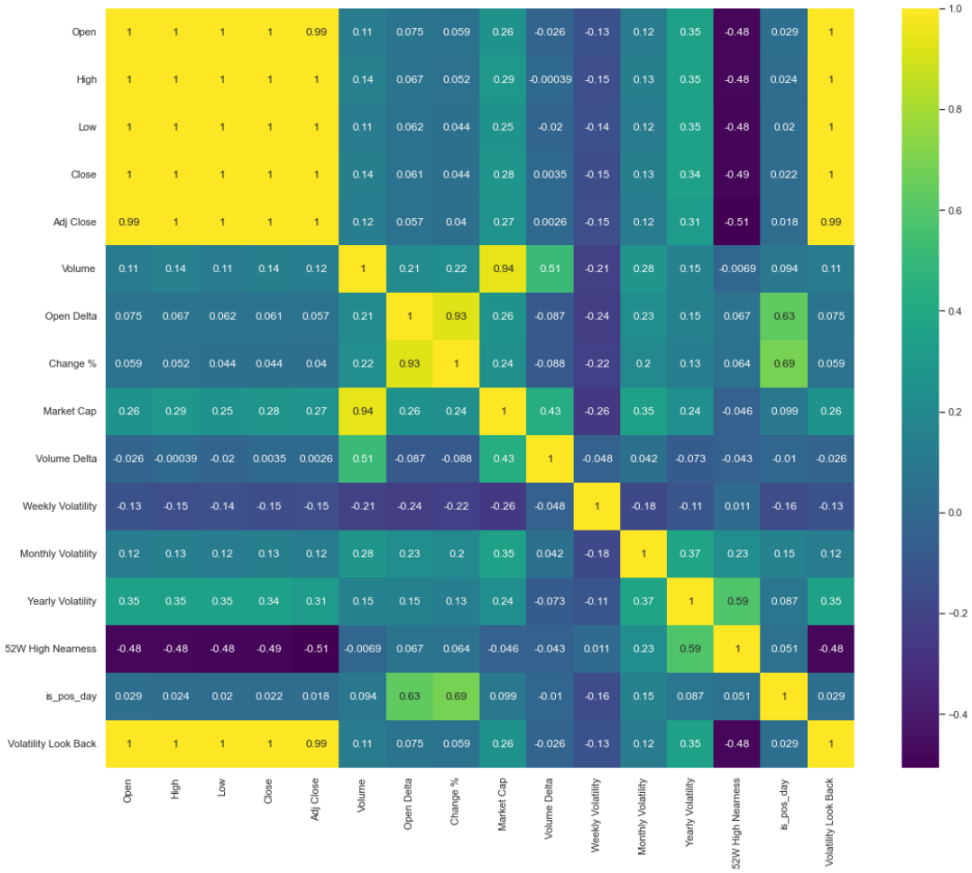


Fig – 4.5.2.4 Correlation Heat map for ESABINDIA

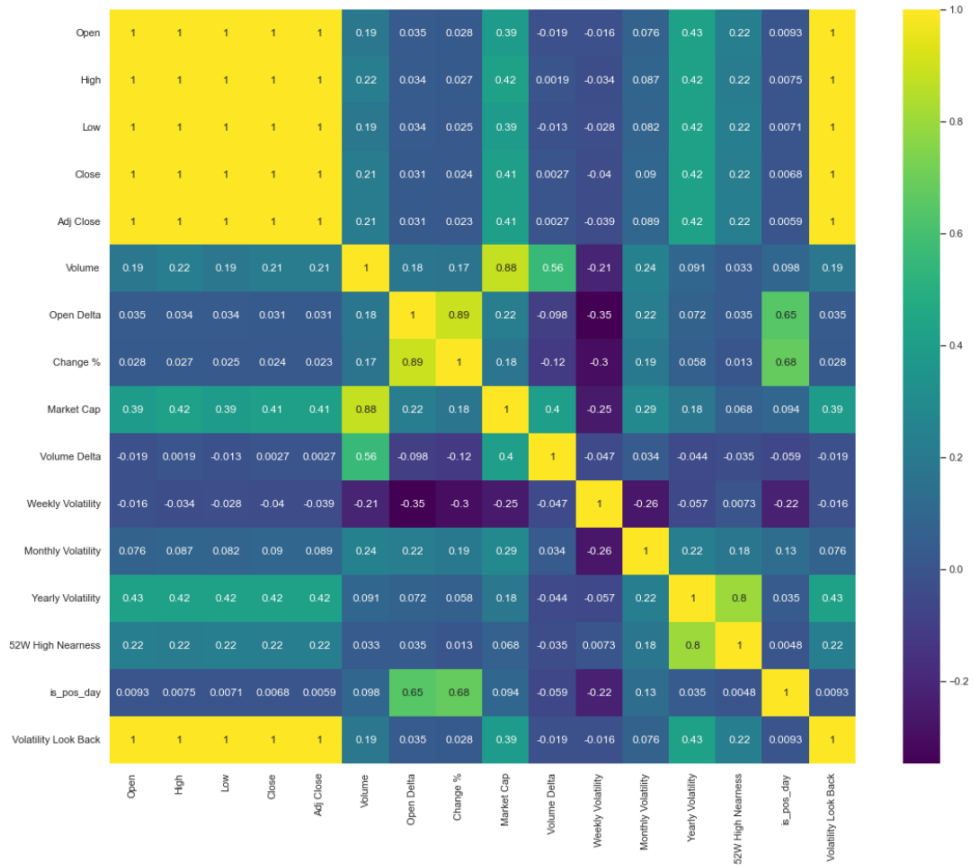


Fig – 4.5.2.5 Correlation Heat map for SMLISUZU

Open, High, Low, Close, and Adj. Close are significantly associated with each other, according to correlation heat map plots. The price of OHLC is closely associated with volatility over the Look Back Period. The plots also consolidates that volatility is inversely associated with change percent and Open Delta.

4.6 Data Transformation

4.6.1 Reviewing the values

In order to run machine learning models, all input and output variables must be numeric. The variables in use right now are all numeric. As a result, there's no need to convert numeric variables from ordinal, nominal, or interval scales.

4.6.2 Scaling

Standard scaling has been applied in order to rescale the values and bring them within the range to enhance learning performance.

CHAPTER 5 - RESULTS & DISCUSSIONS

5.1 Introduction

This section goes over the many models that are in the works. The financial assets were chosen using the criteria outlined in the preceding chapter's selection process. A two-step approach will be undertaken in model implementation based on the various financial assets specified. For a price prediction application, we are evaluating two machine learning techniques. A multi-layer recurrent neural network with Long-Short-Term Memory (LSTM) cells and a hybrid neural network with 2 convolutional neural network layers and 2 LSTM neural network layers. The models have been trained using training data from 2013-19(7 years) and data for year 2020 is being used for testing and validation.

5.2 Development of Stacked LSTM models

In this section, analysis has been done by building RNN based stacked LSTM model for each of our chosen stocks. By performing grid search on the training data, the hyper parameters such as number of epochs, number of trials, and batch were iteratively fit into the LSTM model. The optimum hyper parameters were determined using the mean absolute percentage error (MAPE) loss function. Based on the LSTM model and the test data, the projected values of the closing prices of selected stocks were determined.

The model evaluation parameters for the LSTM models for selected stocks are as follows:

5.2.1 KSCL (LSTM)

Fig 5.2.1 Original, Trained and Predicted Closing Prices for KSCL(LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1425	0.0437	0.1425	0.0292	0.0014	0.0292

Table 5.2.1 – Model Evaluation Metrics KSCL (LSTM)

5.2.2 SOMANYCERA (LSTM)

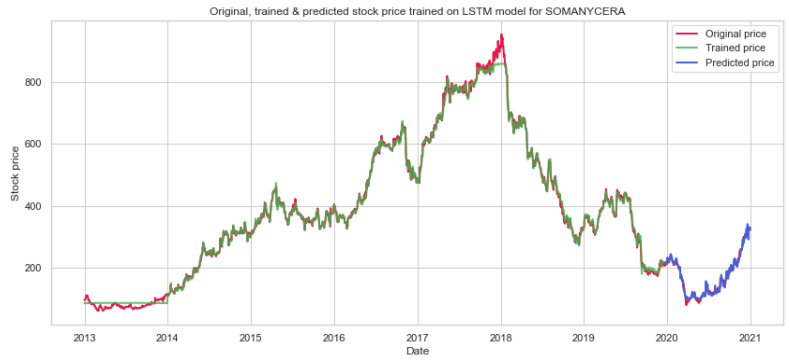


Fig 5.2.2 Original, Trained and Predicted Closing Prices for SOMANYCERA (LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1261	0.0338	0.1261	0.0271	0.0012	0.0271

Table 5.2.2 – Model Evaluation Metrics SOMANYCERA (LSTM)

5.2.3 ORIENTBELL (LSTM)

Fig 5.2.3 Original, Trained and Predicted Closing Prices for ORIENTBELL (LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1105	0.0275	0.1105	0.0203	0.00079	0.0203

Table 5.2.3 – Model Evaluation Metrics ORIENTBELL (LSTM)

5.2.4 ESABINDIA (LSTM)

Fig 5.2.4 Original, Trained and Predicted Closing Prices for ESABINDIA (LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.0495	0.0055	0.0495	0.3740	0.2944	0.3740

Table 5.2.4 – Model Evaluation Metrics ESABINDIA (LSTM)

5.2.5 SMLISUZU (LSTM)

Fig 5.2.5 Original, Trained and Predicted Closing Prices for SMLISUZU (LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1242	0.0319	0.1242	0.1661	0.0537	0.1661

Table 5.2.5 – Model Evaluation Metrics SMLISUZU (LSTM)

5.3 Development of hybrid CNN-LSTM models

In this section, we analysis has been done by building a hybrid CNN + stacked LSTM based model for each of our chosen stocks. By performing grid search on the training data, the hyper parameters such as number of epochs, number of trials, and batch were iteratively fit into the hybrid model. The optimum hyper parameters were determined using the mean absolute percentage error (MAPE) loss function. Based on the hybrid model and the test data, the projected values of the closing prices of selected stocks were determined.

The model evaluation parameters for the hybrid CNN + LSTM models for selected stocks are as follows:

5.3.1 KSCL (CNN-LSTM)

Fig 5.3.1 Original, Trained and Predicted Closing Prices for KSCL (CNN-LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1086	0.0287	0.1086	0.0308	0.0016	0.0308

Table 5.3.1 – Model Evaluation Metrics KSCL (CNN-LSTM)

5.3.2 SOMANYCERA (CNN-LSTM)

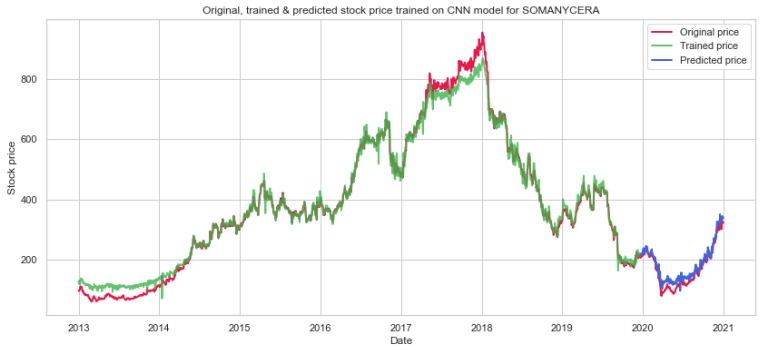


Fig 5.3.2 Original, Trained and Predicted Closing Prices for SOMANYCERA (CNN-LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.1054	0.0240	0.1054	0.0429	0.0029	0.0429

Table 5.3.2 – Model Evaluation Metrics SOMANYCERA (CNN-LSTM)

5.3.3 ORIENTBELL (CNN-LSTM)

Fig 5.3.3 Original, Trained and Predicted Closing Prices for ORIENTBELL (CNN-LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.0961	0.0212	0.0961	0.0313	0.0021	0.0313

Table 5.3.3 – Model Evaluation Metrics ORIENTBELL (CNN-LSTM)

5.3.4 ESABINDIA (CNN-LSTM)

Fig 5.3.4 Original, Trained and Predicted Closing Prices for ESABINDIA (CNN-LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.0425	0.0038	0.0425	0.3086	0.1610	0.3086

Table 5.3.4 – Model Evaluation Metrics ESABINDIA (CNN-LSTM)

5.3.5 SMLISUZU (CNN-LSTM)

Fig 5.3.5 Original, Trained and Predicted Closing Prices for SMLISUZU (CNN-LSTM)

	loss	mse	mae	val_loss	val_mse	val_mae
values	0.0943	0.0180	0.0943	0.0194	0.00068	0.0194

Table 5.3.5 – Model Evaluation Metrics SMLISUZU (CNN-LSTM)

5.4 Returns Analysis using Backtesting

The main objectives of this study was to confirm whether we can increase our return of investment (ROI). The first step was to filter out quality stocks with momentum using our specific stock selection criteria. Next step was to use the developed DL models to predict future closing price which can be further processed to generate trading signals to test whether we can achieve a respectable ROI in the period of 2020-21 which should exceed/comparable to the returns provided by the indices like NIFTY50 and BSE SENSEX.

For our analysis, we have used an existing backtesting environment to generate trading signals for the time period 2020-21 by using the predicted closing prices along with data from INDIAVIX which is a volatility index based on the NIFTY Index Option prices. The backtesting has been performed using an initial fund of ₹100000 and allocation ratio i.e. investment amount for each trade is set as 30% of the current portfolio value. The environment also takes into consideration the commission and slippage price for each trade.

Two trading approaches have been used to analyse the returns yielded by each of our models. First approach is the Buy and Hold strategy and the second is the intraday trading strategy based on the trading signals. The results of the backtesting are as follows:

5.4.1 KSCL (LSTM)

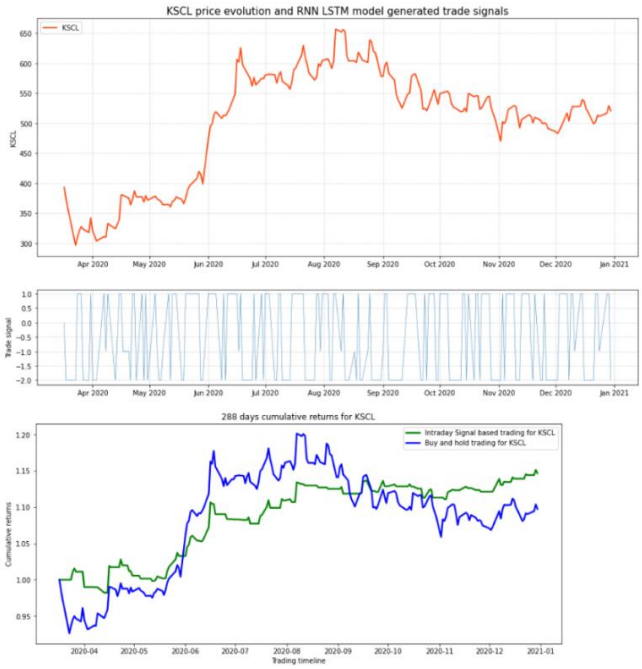


Fig 5.4.1 – Returns% for KSCL (LSTM)

	LSTM(End)	LSTM(Highest)
% Returns Buy and Hold	9.75	20.14
% Returns Intraday Trading	14.67	15.14

Table 5.4.1 – Returns% for KSCL (LSTM)

5.4.2 SOMANYCERA (LSTM)

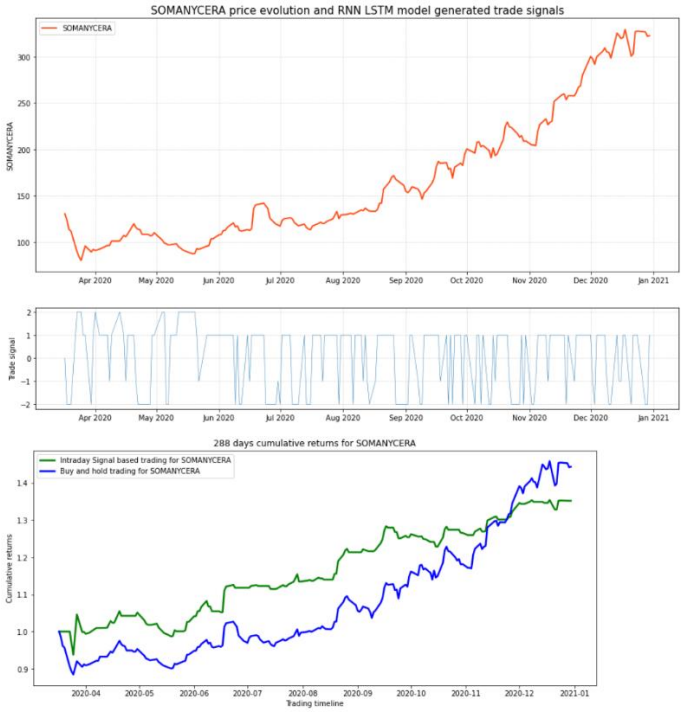


Fig 5.4.2 – Returns% for SOMANYCERA (LSTM)

	LSTM(End)	LSTM(Highest)
% Returns Buy and Hold	44.30	45.82
% Returns Intraday Trading	35.10	35.30

Table 5.4.2 – Returns% for SOMANYCERA (LSTM)

5.4.3 ORIENTBELL (LSTM)

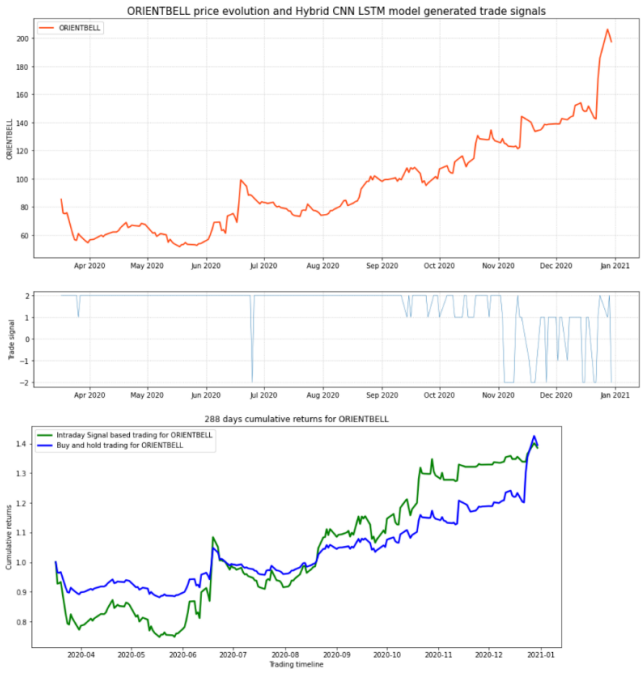


Fig 5.4.3 – Returns% for ORIENTBELL (LSTM)

	LSTM(End)	LSTM(Highest)
% Returns Buy and Hold	39.40	42.53
% Returns Intraday Trading	38.51	40.09

Table 5.4.3 – Returns% for ORIENTBELL (LSTM)

5.4.4 ESABINDIA (LSTM)

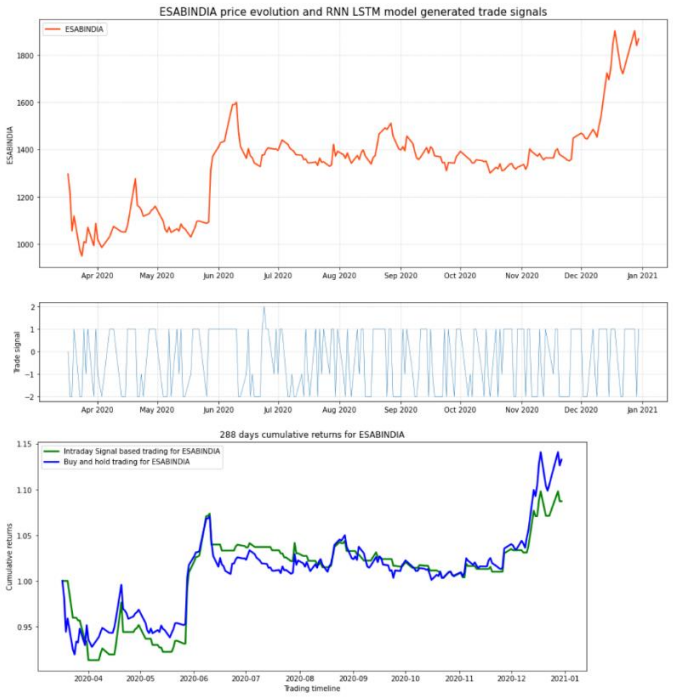


Fig 5.4.4 – Returns% for ESABINDIA(LSTM)

	LSTM(End)	LSTM(Highest)
% Returns Buy and Hold	13.28	14.10
% Returns Intraday Trading	8.72	9.81

Table 5.4.4 – Returns% for ESABINDIA (LSTM)

5.4.5 SMLISUZU (LSTM)

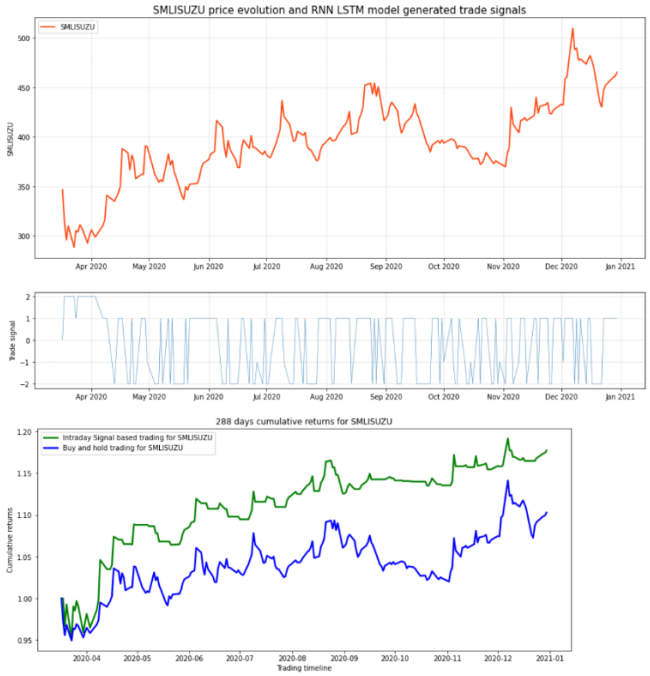


Fig 5.4.5 – Returns% for SMLISUZU (LSTM)

	LSTM(End)	LSTM(Highest)
% Returns Buy and Hold	10.28	14.13
% Returns Intraday Trading	17.71	19.15

Table 5.4.5 – Returns% for SMLISUZU (LSTM)

5.4.6 KSCL (CNN-LSTM)

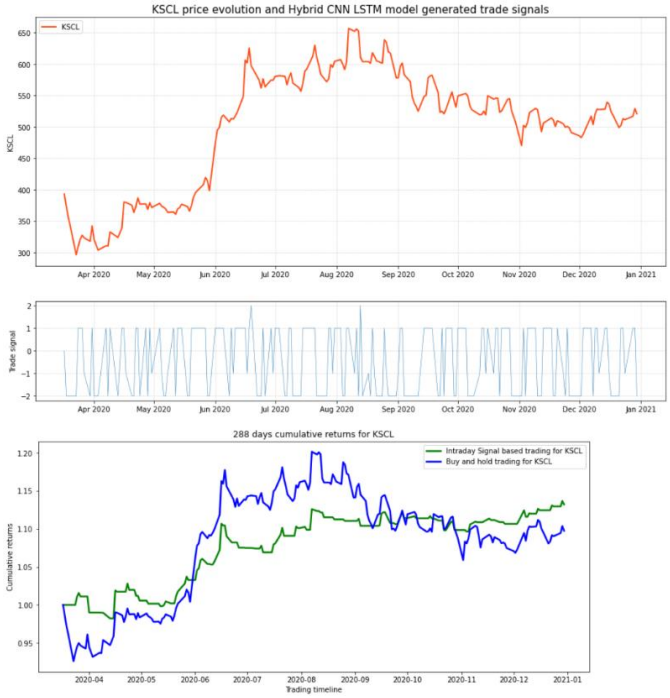


Fig 5.4.6 – Returns% for KSCL (CNN-LSTM)

	CNN+LSTM (End)	CNN+LSTM (Highest)
% Returns Buy and Hold	9.75	20.14
% Returns Intraday Trading	13.20	13.67

Table 5.4.6 – Returns% for KSCL (CNN-LSTM)

5.4.7 SOMANYCERA (CNN-LSTM)

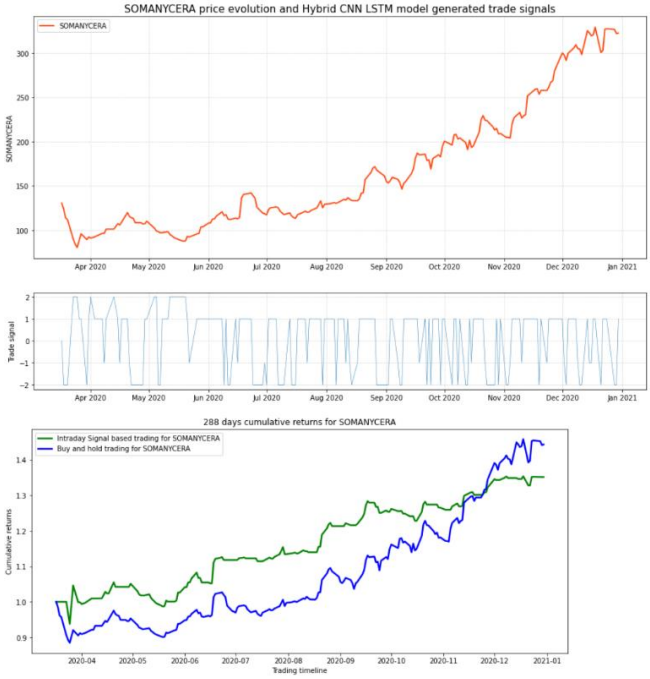


Fig 5.4.7 – Returns% for SOMANYCERA (CNN-LSTM)

	CNN+LSTM (End)	CNN+LSTM (Highest)
% Returns Buy and Hold	44.30	45.82
% Returns Intraday Trading	35.15	35.37

Table 5.4.7 – Returns% for SOMANYCERA (CNN-LSTM)

5.4.8 ORIENTBELL (CNN-LSTM)

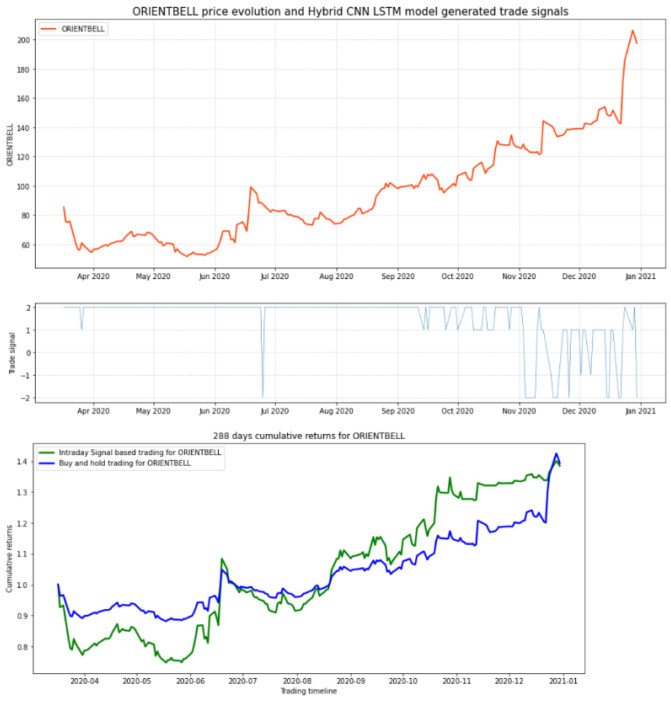


Fig 5.4.8 – Returns% for ORIENTBELL (CNN-LSTM)

	CNN+LSTM (End)	CNN+LSTM (Highest)
% Returns Buy and Hold	39.40	42.53
% Returns Intraday Trading	38.51	40.09

Table 5.4.8 – Returns% for ORIENTBELL (CNN-LSTM)

5.4.9 ESABINDIA (CNN-LSTM)

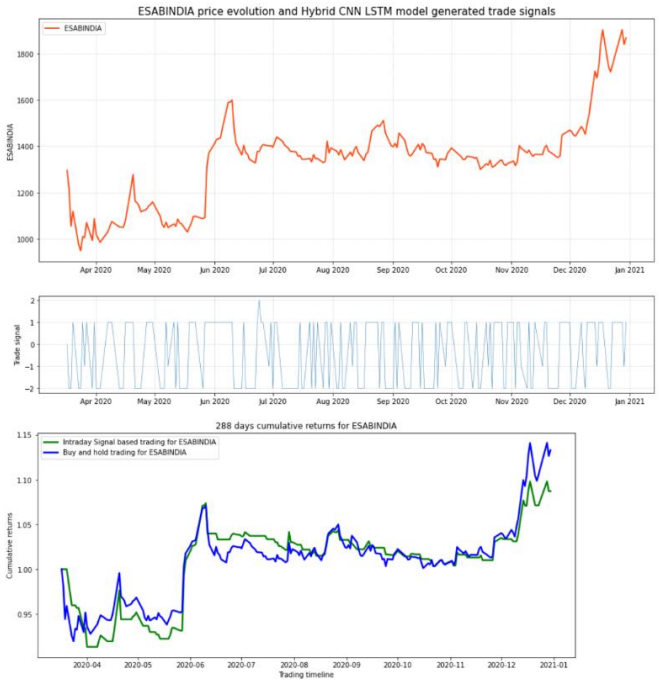


Fig 5.4.9 – Returns% for ESABINDIA (CNN-LSTM)

	CNN+LSTM (End)	CNN+LSTM (Highest)
% Returns Buy and Hold	13.28	14.10
% Returns Intraday Trading	8.72	9.81

Table 5.4.9 – Returns% for ESABINDIA (CNN-LSTM)

5.4.10 SMLISUZU (CNN-LSTM)

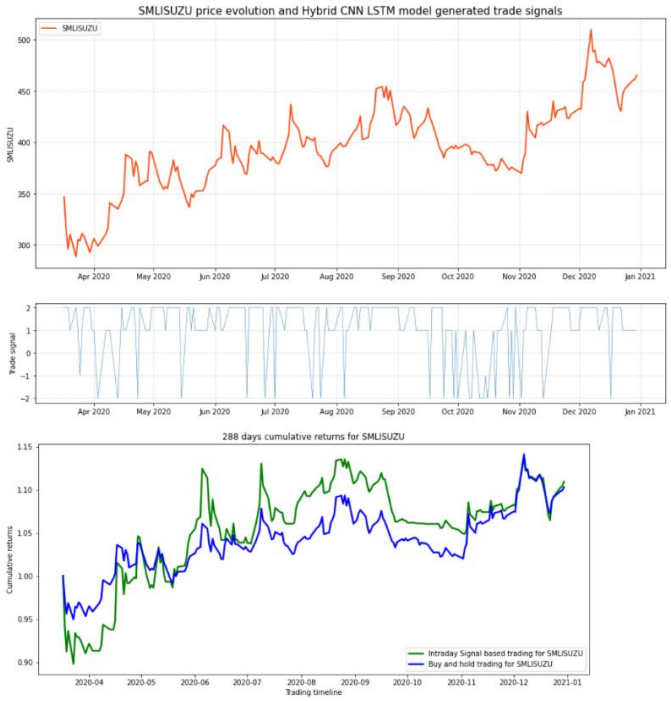


Fig 5.4.10 – Returns% for SMLISUZU (CNN-LSTM)

	CNN+LSTM (End)	CNN+LSTM (Highest)
% Returns Buy and Hold	10.28	14.13
% Returns Intraday Trading	10.94	13.76

Table 5.4.10 – Returns% for SMLISUZU (CNN-LSTM)

Based on the results of backtesting each of our models we can observe that:

- The returns yielded from KSCL (LSTM), KSCL (CNN+LSTM) and SMLISUZU (LSTM) are higher for Intraday Trading using generated signals compared to Buy and Hold approach.
- The returns yielded from SOMANYCERA (LSTM), SOMANYCERA (CNN+LSTM), ESABINDIA (LSTM) and ESABINDIA (CNN+LSTM) are higher for Buy and Hold strategy compared to Intraday Trading using generated signals.
- ORIENTBELL (LSTM), ORIENTBELL (CNN+LSTM) and SMLISUZU (CNN+LSTM) shows comparable results in both trading approaches.

Index	Returns %
NIFTY50	**16.00**
BSE SENSEX	**15.48**

Table 5.4.11 - Annual Returns of Indices for the period 2020-2021

	LSTM(B&H)	LSTM(Signal)	CNN+LSTM(B&H)	CNN+LSTM(Signal)
KSCL	9.75	14.67	9.75	13.20
SOMANYCERA	44.30	35.10	44.30	35.15
ORIENTBELL	39.40	38.51	39.40	38.51
ESABINDIA	13.28	8.72	13.28	8.72
SMLISUZU	10.28	17.71	10.28	10.94
Average Returns %	**23.40**	**22.94**	**23.40**	**21.30**
Av. Returns compared to NIFTY50	**+46.25**	**+43.37**	**+46.25**	**+33.12**
Av. Returns compared to BSE SENSEX	**+51.16**	**+48.19**	**+51.16**	**+37.59**

Table 5.4.12 - Average Returns across Trading Period for 2020-2021

The annual returns for the indices NIFTY50 and BSE SENSEX has been noted in Table 5.4.11. The data has been taken from their official website (www1.nseindia.com, www.bseindia.com). Also, we can observe from table 5.4.12 that the returns yielded from our chosen models using the stocks filtered using our stock selection criteria exceed that of the annual returns provided by NIFTY50 by at least 33.12% and BSE SENSEX by at least 37.59%

Some interesting observations to note here are:

- Buy & Hold Trading Strategy using both LSTM and CNN+LSTM approach has yielded the highest returns over the trading period 2020-21. Its ROI of 23.4 % exceed that of NIFTY50 by 46.25% and BSE SENSEX by 51.16%.
- Intraday Trading approach using generated signals for LSTM has yielded ROI of 22.94% which exceed that of NIFTY50 by 43.37% and BSE SENSEX by 48.19%.
- Intraday Trading approach using generated signals for CNN+LSTM has yielded ROI of 21.3% which exceed that of NIFTY50 by 33.12% and BSE SENSEX by 37.59%.

5.5 Resources

5.5.1 Hardware Resources

The models were implemented on a PC with the following specifications:

Processor: Intel(R) Core (TM) i9-9900K CPU

Memory: 32.0 GB

System type: 64-bit operating system, x64-based processor

GPU: NVIDIA RTX 2080

5.5.2 Software Resources

- Python 3.7 has been used to develop the Prediction Models and backtesting environment.
- nsepy python package has been used to download NSE stock dataset.
- Tensorflow-gpu 2.5.0 and its associated dependencies.

5.6 Summary

In this section, the selected financial assets were used to build Deep Learning Models for future stock closing price prediction. Two types of Deep Learning models were built for each of the five selected financial assets and hyper parameters were optimized to achieve as much prediction accuracy as possible within the specified hardware constraints. The evaluation metrics for each of the model are summarised in Table 5.5.1 – Evaluation Results Summary.

	LSTM			CNN+LSTM		
	RMSE	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**
KSCL	0.0374	9.90	0.0292	0.0400	9.50	0.0308
SOMANYCERA	0.0346	0.52	0.0271	0.0538	0.49	0.0429
ORIENTBELL	0.0281	1.91	0.0203	0.0458	1.98	0.0313
ESABINDIA	0.5425	0.83	0.3740	0.4012	0.50	0.3086
SMLISUZU	0.2317	0.39	0.1661	0.0260	0.30	0.0194

Table 5.5.1 – Evaluation Results Summary

Based on the table, we can see that the RNN-based LSTM model predicts values that are extremely near to the real or observed values, indicating that it is one of the top models for Time Series Prediction. Furthermore, the outcomes of the hybrid CNN + LSTM technique improved prediction accuracy marginally, as shown in Table 5.5.1.

5.7 Review of Hypothesis

Based on the results of this section, we can confirm the following hypothesis:

- Deep Learning based RNN approaches perform extremely well in prediction of Time Series stock data.
- A hybrid approach like CNN + LSTM can enhance the performance of what we can achieve by using only RNN based stacked LSTM approach.
- In order to maximise return on investment, quality asset selection is just as crucial as the prediction algorithm's performance, as well as the trading agent's and environment's efficacy, as evidenced by the returns produced by the basic Buy and Hold Trading Strategy.

CHAPTER 6 – CONCLUSIONS AND RECOMMENDATIONS

6.1 Discussion and Summary

Portfolio Management using Time Series Forecasting has been a prevalent field of research for past many years and many studies has been performed to analyse the performance of various Machine Learning approaches in this field. Starting from various statistical prediction methods to identify factors like trend, moving average, seasonality, relative strength index and many such indicators to traditional Machine Learning approaches like Simple Linear Regression, Tree based approaches like Decision Tree, Ensemble methods like Random Forests have been tried and tested by a great number of studies using different Financial assets belonging to different stock exchanges. The major improvements have been observed in this field since the inception of Deep Learning in the last decade. Based on our literature review, we observed that Deep Learning based approaches like Artificial Neural Networks, Multilayer Perceptron increased the prediction performance significantly. Another major boon for this field of study happened when Recurrent Neural Networks came into picture which perform extremely well for Time Series data outperform the traditional counterparts significantly. Since then various Hybrid approaches have also been proposed as well. Although many studies has been done to increase the efficacy of the prediction algorithm, not much research has been done to see how much significance the choice of financial assets have if the final objective is to maximize the return of investment.

Therefore, three different hypothesis were proposed. First hypothesis is to establish that the chosen RNN based stacked LSTM model should perform exceedingly well for prediction daily closing prices. Second hypothesis is that the hybrid approach of CNN + stacked LSTM should enhance the prediction accuracy of the standard LSTM. Lastly, the third hypothesis is focused on the significance of the choice of financial assets on the returns yielded.

In this study, major emphasis was given on the selection of stocks with momentum on the basis of returns, volatility and nearness to 52W high over look back period of 22 trading days. Data pertaining to the selected stocks were collected for the period of eight years 2013-20 and the proposed models were developed. The first two hypothesis were confirmed by evaluation of prediction models against metrics like MAE, MAPE and RMSE. The third hypothesis was confirmed by backtesting the developed model and comparing the returns yielded by the typical Buy & Hold trading approach where all of the chosen stocks yielded

returns comparable to the signal based trading approach. Furthermore, the ROI for each model were compared with the returns of indices like NIFTY50 and BSE SENSEX where we saw the models exceeded the returns of the indices by at least ~33% to a maximum of ~50%.

In conclusion, the study effectively examined each hypothesis that was first offered, and conclusions were drawn in response to each hypothesis. This section summarises the information that this research may provide to the science of stock market forecasting utilising various statistical approaches, as well as the models that enable superior forecasting. In addition, this part discusses the limits, results, and recommendations for future stock market prediction research.

6.2 Limitations

The following are some of the study's limitations:

1. The research was confined to Indian stock market equities; however, there are a variety of other financial assets, such as derivatives, bitcoin, and commodities that may have a larger potential ROI and create different outcomes.
2. While there have been a number of novel deep learning techniques presented, such as Attention models and Transformers models, the focus of this research is on RNN-based stacked LSTM and a hybrid CNN + stacked LSTM approach.
3. Rather than using a Reinforcement Learning trading agent, this study employs a static method to produce trade signals for ROI evaluation.
4. This research was conducted using equities from the Indian stock market, which is considered a developing market. In comparison to exchanges like the NYSE, which is frequently regarded as the world's most affluent and established financial market, the volatility of Indian indexes and equities is very significant. As a result, if this study is conducted using financial assets from a well-established financial exchange, the findings may differ slightly.

6.3 Final Conclusions

This research looked at all of the stocks listed on the NSE, filtered out the ones that were gaining momentum over a certain time period, and built Deep Learning models to forecast daily closing prices. In addition, the ROI of these models was assessed using a Trading agent

and various Trading techniques. Various hypotheses were also investigated, with the following results:

1. In the prediction of Time Series stock data, Deep Learning-based RNN algorithms perform exceedingly well.
2. A hybrid technique, such as CNN + LSTM, can improve performance over a stacking LSTM strategy based only on RNNs.
3. Quality asset selection is equally as important as the prediction algorithm's performance, as well as the trading agent's and environment's efficacy, in maximising return on investment which is evident by the returns achieved by the standard Buy and Hold Trading Strategy.

6.4 Contribution and Importance of Study

The study's major goal was to optimise investment return by screening out stocks with high momentum while keeping their volatility in mind and utilising them to develop effective deep learning models to forecast daily closing prices. Trading signals may then be generated using the projected prices. The prediction performance and ROI of several DL models were investigated and assessed in this study. Furthermore, their return on investment was compared to those of indices such as the NIFTY50 and the BSE SENSEX.

The major topics to which this work makes a significant contribution to understanding are as follows:

1. The research shows that financial asset closing prices may be forecasted quite effectively utilising deep learning algorithms without taking into account the stock's fundamentals.
2. Different hyperparameters may be used to fine-tune different machine learning and deep learning models.
3. These models may be used by financial and portfolio managers all over the world to anticipate the closing prices of stocks or any other financial assets in order to manage their portfolios.
4. To optimize return on investment, quality asset selection is just as critical as the prediction algorithm's performance and the trading agent's and environment's efficacy.

6.5 Future Recommendations

The method used in this study, as well as the results, provided answers to the primary hypotheses that had been suggested. However, there is more study that may be done in areas that are both scholarly and commercial in nature.

1. There are a variety of deep learning methods to choose from, and thorough study may be done to compare them. Models such as generative adversarial networks, attention models, and Transformers can be used in the study, and their efficacy can be compared.
2. Some hybrid approaches combining multiple deep learning models can be built. Combining the Attention mechanism with the LSTM to create a hybrid deep learning model is one example of this.
3. The trading agent used in this study during backtesting uses a static algorithm to generate trading signals. This can be replaced with a Reinforcement Learning approach to generate signals and improve the performance of the trading agent.
4. The research only looked at equities as a form of financial asset. Other financial assets, such as cryptocurrency and derivatives such as Futures and Options, might be included in future study because the rewards are substantially larger.

REFERENCES

Arpaci, U., & Karaoglu, S. (2017). A Deep Learning Approach for Optimization of Systematic Signal Detection in Financial Trading Systems with Big Data. *International Journal of Intelligent Systems and Applications in Engineering*, *Special Issue*(Special Issue), 31–36. https://doi.org/10.18201/ijisae.2017SpecialIssue31421

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, *12*(7), e0180944. https://doi.org/10.1371/journal.pone.0180944

Batres-Estrada, B. (2015). *Deep learning for multivariate financial time series*.

Chandra, R., & Chand, S. (2016). Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance. *Applied Soft Computing*, *49*, 462–473. https://doi.org/10.1016/j.asoc.2016.08.029

Chang, P.-C., Liao, T. W., Lin, J.-J., & Fan, C.-Y. (2011). A dynamic threshold decision system for stock trading signal detection. *Applied Soft Computing*, *11*(5), 3998–4010. https://doi.org/10.1016/j.asoc.2011.02.029

Cho, K., Merrienboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*.

Das, P., & Banerjee, A. (2011). Meta optimization and its application to portfolio selection. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '11*, 1163. https://doi.org/10.1145/2020408.2020588

De BONDT, W. F. M., & Thaler, R. (1985). Does the Stock Market Overreact? *The Journal of Finance*, *40*(3), 793–805. https://doi.org/10.1111/j.1540-6261.1985.tb05004.x

Ding, Q., Wu, S., Sun, H., Guo, J., & Guo, J. (2020). Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. *IJCAI*.

Dixon, M., Klabjan, D., & Bang, J. H. (2015). Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. *Proceedings of the 8th Workshop on High Performance Computational Finance*, 1–6. https://doi.org/10.1145/2830556.2830562

Elmsili, B., & Outtaj, B. (2018). Artificial neural networks applications in economics and management research: An exploratory literature review. *2018 4th International Conference on Optimization and Applications (ICOA)*, 1–6. https://doi.org/10.1109/ICOA.2018.8370600

Feng, F., Chen, H., He, X., Ding, J., Sun, M., & Chua, T.-S. (2019). Enhancing Stock Movement Prediction with Adversarial Training. *IJCAI*.

Fischer, T., & Krauss, C. (2017). *Deep learning with long short-term memory networks for financial market predictions* (Issue 11/2017). Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics. https://ideas.repec.org/p/zbw/iwqwdp/112017.html

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2003). Learning Precise Timing with Lstm Recurrent Networks. *J. Mach. Learn. Res.*, *3*(null), 115–143. https://doi.org/10.1162/153244303768966139

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). *Maxout Networks*.

Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. *ArXiv:1308.0850 [Cs]*. http://arxiv.org/abs/1308.0850

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Gyorfi, L., Lugosi, G., & Udina, F. (2006). NONPARAMETRIC KERNEL-BASED SEQUENTIAL INVESTMENT STRATEGIES. *Mathematical Finance*, *16*(2), 337–357. https://doi.org/10.1111/j.1467-9965.2006.00274.x

Hameed, A., & Ting, S. (2000). Trading volume and short-horizon contrarian profits: Evidence from the Malaysian market. *Pacific-Basin Finance Journal*, *8*(1), 67–84.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2018). Deep Learning in Finance. *ArXiv:1602.06561 [Cs]*. http://arxiv.org/abs/1602.06561

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In S. C. Kremer & J. F. Kolen (Eds.), *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huck, N. (2009). Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, *196*(2), 819–825.

Kearney, C., & Liu, S. (2013). Textual Sentiment Analysis in Finance: A Survey of Methods and Models. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2213801

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, *3*(3), 263–286. https://doi.org/10.1007/PL00011669

Khadjeh Nassirtoussi, A., Aghabozorgi, S., Ying Wah, T., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653–7670. https://doi.org/10.1016/j.eswa.2014.06.009

Kraus, M., & Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, *104*, 38–48. https://doi.org/10.1016/j.dss.2017.10.001

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

Lee, S. I., & Yoo, S. J. (2018). Threshold-Based Portfolio: The Role of the Threshold and Its Applications. *ArXiv:1709.09822 [q-Fin]*. http://arxiv.org/abs/1709.09822

Li, B., & Hoi, S. C. H. (2013). Online Portfolio Selection: A Survey. *ArXiv:1212.2129 [Cs, q-Fin]*. http://arxiv.org/abs/1212.2129

Li, Y., & Ma, W. (2010). Applications of Artificial Neural Networks in Financial Economics: A Survey. *2010 International Symposium on Computational Intelligence and Design*, 211–214. https://doi.org/10.1109/ISCID.2010.70

Liu, S., Zhang, C., & Ma, J. (2017a). CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets. In D. Liu, S. Xie, Y. Li, D. Zhao, & E.-S. M. El-Alfy (Eds.), *Neural Information Processing* (Vol. 10635, pp. 198–206). Springer International Publishing. https://doi.org/10.1007/978-3-319-70096-0_21

Liu, S., Zhang, C., & Ma, J. (2017b). *CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets* (p. 206). https://doi.org/10.1007/978-3-319-70096-0_21

M, H., E.A., G., Menon, V. K., & K.P., S. (2018). NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Computer Science*, *132*, 1351–1362. https://doi.org/10.1016/j.procs.2018.05.050

Minami, S. (2018). Predicting Equity Price with Corporate Action Events Using LSTM-RNN. *Journal of Mathematical Finance*, *08*(01), 58–63. https://doi.org/10.4236/jmf.2018.81005

Mittermayer, M.-A., & Knolmayer, G. (2006). *Text mining systems for market response to news: A survey*.

Moritz, B., & Zimmermann, T. (2016). Tree-Based Conditional Portfolio Sorts: The Relation between Past and Future Stock Returns. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2740751

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *ArXiv:1704.02971 [Cs, Stat]*. http://arxiv.org/abs/1704.02971

Samarawickrama, A. J. P., & Fernando, T. G. I. (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–6. https://doi.org/10.1109/ICIINFS.2017.8300345

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks*.

Takeuchi, L. (2013). *Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks*.

Tkáč, M., & Verner, R. (2016). Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, *38*, 788–804. https://doi.org/10.1016/j.asoc.2015.09.040

Xing, F. Z., Cambria, E., & Welsch, R. E. (2018). Natural language based financial forecasting: A survey. *Artificial Intelligence Review*, *50*(1), 49–73. https://doi.org/10.1007/s10462-017-9588-9

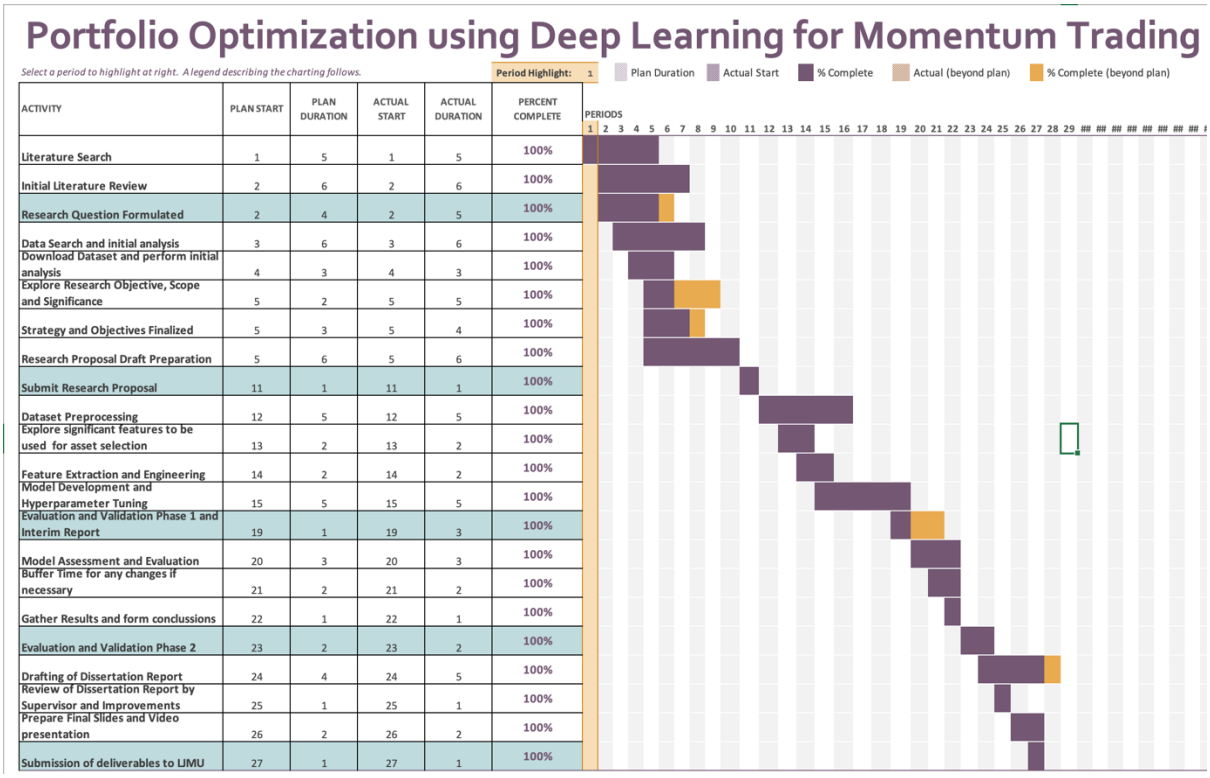
Xu, Y., & Cohen, S. B. (2018). Stock Movement Prediction from Tweets and Historical Prices. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1970–1979. https://doi.org/10.18653/v1/P18-1183

Yuan, Z., Zhang, R., & Shao, X. (2018a). Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation. *Proceedings of the 2018 International Conference on Computing and Data Engineering*, 39–43. https://doi.org/10.1145/3219788.3219793

Yuan, Z., Zhang, R., & Shao, X. (2018b). Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation. *Proceedings of the 2018 International Conference on Computing and Data Engineering*, 39–43. https://doi.org/10.1145/3219788.3219793

Zhang, L., Aggarwal, C. C., & Qi, G.-J. (2017). Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

APPENDIX A: RESEARCH PLAN



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature Search	1	5	1	5	100%
Initial Literature Review	2	6	2	6	100%
Research Question Formulated	2	4	2	5	100%
Data Search and initial analysis	3	6	3	6	100%
Download Dataset and perform initial analysis	4	3	4	3	100%
Explore Research Objective, Scope and Significance	5	2	5	5	100%
Strategy and Objectives Finalized	5	3	5	4	100%
Research Proposal Draft Preparation	5	6	5	6	100%
Submit Research Proposal	11	1	11	1	100%
Dataset Preprocessing	12	5	12	5	100%
Explore significant features to be used for asset selection	13	2	13	2	100%
Feature Extraction and Engineering	14	2	14	2	100%
Model Development and Hyperparameter Tuning	15	5	15	5	100%
Evaluation and Validation Phase 1 and Interim Report	19	1	19	3	100%
Model Assessment and Evaluation	20	3	20	3	100%
Buffer Time for any changes if necessary	21	2	21	2	100%
Gather Results and form conclussions	22	1	22	1	100%
Evaluation and Validation Phase 2	23	2	23	2	100%
Drafting of Dissertation Report	24	4	24	5	100%
Review of Dissertation Report by Supervisor and Improvements	25	1	25	1	100%
Prepare Final Slides and Video presentation	26	2	26	2	100%
Submission of deliverables to LJMU	27	1	27	1	100%

APPENDIX B: RESEARCH PROPOSAL

Abstract

Financial portfolio management refers to the continuous reassignment of funds into financial assets like cash, stocks, bonds, mutual funds, bank deposits and commodities. The financial portfolio is created based on the investors' risk tolerance, investment objectives, and time period. The price of each asset has an impact on the portfolio's risk/reward ratio. It's also known as portfolio asset allocation. Because the investing world is so volatile, portfolio management includes asset allocation and rebalancing. With the fast changes that occur throughout time, one sector may become less popular than another, and vice versa. As a result, only a nimble investor can benefit handsomely from the world's stock markets.

To tackle the challenge of Financial Time Series Forecasting, researchers devised a variety of Machine Learning models, and a lot of papers have been published as a result. Despite the increased interest in creating financial time series forecasting models, few review papers have focused on optimizing financial portfolios to maximize returns. As a result, the goal of this paper is to develop a method for managing our portfolio assets so that we may maximize profits depending on market momentum.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
OHLCV	Open-High-Low-Close-Volume
DFNN	Deep Feedforward Neural Network
FX	Forex
ML	Machine Learning
ANN	Artificial Neural Network
EC	Evolutionary Computation
GP	Genetic Programming
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
DMLP	Deep Multilayer Perceptron
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
AE	Autoencoder
DRL	Deep Reinforcement Learning
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
NLP	Natural Language Processing
ARIMA	Autoregressive Integrated Moving Average
SRNN	Stacked Recurrent Neural Network
DWNN	Deep and Wide Neural Network
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
EDA	Exploratory Data Analysis

1 Background

Portfolio management is the dynamic cycle of consistently redistributing a measure of financial assets into various diverse monetary investment products, where the objective is to boost the returns while limiting the risk as much ass possible. According to the study by (B. Li & Hoi, 2013), Traditional portfolio management strategies can be categorized into four classes, “Follow the Winner”, “Follow the Loser”, “Pattern Matching”, and “Meta Learning”

“Follow the Winner”, attempts to asymptotically accomplish a same growth rate (expected log return) as that of an ideal strategy. According to the “Follow-the-Loser” strategy, wealth should be transferred from winning assets to losing ones, which seems contradictory as per common sense but according to multiple observations, it often generates remarkable results. “Pattern Matching” algorithms anticipate the subsequent market distribution based on the sampled distribution and explicitly optimizes the portfolio using the sampled distribution as per study by (Gyorfi et al., 2006). Lastly, “Meta Learning” strategy consolidates multiple methodologies of different classifications to achieve more consistent and predictable results (Das & Banerjee, 2011).

Many Deep Learning based approaches have been explored for financial stock market prediction so far. However, the majority of them attempt to forecast price changes or trends for a selected number of assets where the focus is on the said assets only rather than the momentum of the market itself. As per design, a neural network generates an output vector of asset prices for the upcoming time period using just historical raw price data (OHLCV) of the financial assets as input. The trading agent can then take action based on the prediction. This can simply be termed as a supervised learning regression problem, and it is straightforward to implement. As a result, the accuracy of these price prediction-based algorithms is largely dependent on their performance, and it turns out that capturing future market momentum is challenging.

In order to manage the portfolio efficiently to maximize returns based on the momentum of the market, it is essential to choose quality assets themselves rather than only focus on the trading strategy itself. The quality of assets may depend on the intrinsic factor of the asset itself like Current Volume, Price Change %, Market Cap, Volatility, Nearness to 52W High etc. along with its historical price itself. These intrinsic attributes also depend on some extrinsic factors like the sentiments regarding the asset, the sector of the asset and the market momentum itself.

The most frequently researched financial application field is financial time series forecasting, specifically asset price forecasting where the major focus is always on predicting underlying asset's next movement. This was the focus of the majority of the existing DL implementations. Despite the existence of several sub-problems like price forecasting for Stocks, Indexes, Forex, Cryptocurrency, Bonds, Commodities, Volatility and many more, the fundamental elements in all of these applications are very similar.

Based on their dependent output, which can be either price prediction or price movement prediction, the studies can be divided into two groups. Although price prediction can be termed as a fundamental regression problem, in majority of its applications, correctly recognizing the directional movement or momentum is more essential than precise price predictions. Due to this, analysts regard trend forecasting, or anticipating the directional movement of the price, to be more important field of research than precise prediction of price. In this case, trend/momentum prediction turns into a classification problem. As per certain studies, only up and down price movement (2-class problem) are taken into account but neutral movement is also taken into account in others (3-class problem).

DL models like LSTM and its variations, as well as certain hybrid models, have dominated the financial time series forecasting arena. Because LSTM by design is built to exploit the temporal features of any time series data, identifying financial time series signals is a closely examined and useful application. Some analysts, on the other hand, choose to extract relevant characteristics from time series data or modify time series data such that following financial data is fixed in time. This means that the model can be trained correctly and better test performance can be achieved on unseen data regardless of the data order. DFNN and CNNs were the primarily used models in those implementations.

2 Related works

Portfolio management is one of the most significant factors when it comes to making investments into any financial assets which can include stock market, mutual funds, cryptocurrency, bonds, forex and other commodities. For a long time, traditional statistical approaches have been employed to construct an effective portfolio. Following the tremendous advancements in the area of Machine Learning over the last decade, there has been a significant movement away from traditional statistical approaches and toward ML-based strategies that are far more efficient. A large number of studies has already been done in this field and all of

these approaches try to predict price movements or trends/momentum of the chosen financial assets. A Machine Learning model can produce a projected vector of asset values for the upcoming period using past prices of the selected assets as input. The trading agent then takes action after getting the expected signal. ML models like ANNs, ECs, GP and Agent based models have been used extensively by researchers and analysts for publications and industry applications. There are also a number of survey papers which are focused solely in a single ML model. Despite being focused on a single ML technique, their implementations have spanned a number of financial and banking applications. EC and ANN were the most popular of these ML-based techniques.

(Y. Li & Ma, 2010) looked at a number of contemporary ANN implementations for stock market prediction and similar financial purposes. (Tkáč & Verner, 2016) investigated several ANN implementations in financial applications, such as stock price forecasting. (Elmsili & Outtaj, 2018) recently published a study that included applications of ANN in financial aspects and management studies, as well as financial time series estimation.

There also have been other text mining studies concentrating on asset price forecasting. (Mittermayer & Knolmayer, 2006) investigated and evaluated several text mining strategies for predicting market reaction to financial articles and news. In their work, Leela et. al. have also focused on news analytics to forecast any unexpected returns for trading techniques. Various text mining research for stock and FX market prediction were evaluated by (Khadjeh Nassirtoussi et al., 2014)

(Kearney & Liu, 2013) have also looked at sentiment based time series forecasting and trading algorithms based on sentiments extracted from texts. (Xing et al., 2018) evaluated NLP-based financial forecasting research in their latest investigations.

The financial world received a boost in the last decade thanks to the advent of DL-based models for financial forecasting studies, which resulted in a slew of new papers that heavily utilized DL models. Several new deep learning models with alternative deep architectures have been explored and performance evaluated in comparison to traditional ML models and DL-based models beat their ML counterparts in the great majority of these experiments.

Researchers have extensively investigated DL models such as DMLP, RBMs, DBN, AE, RNN, LSTM, CNN and DRL in several papers. We can see that LSTM and its variants, as well as hybrid ensemble approaches, dominate in the area of financial time series forecasting in all of the studies that have used DL models. Some research used just raw pricing data (OCHLV) for predicting and used several DL models for performance comparison. Among some noteworthy studies (Samarawickrama & Fernando, 2017) compared RNN, Stacked RNN, LSTM, and GRU in their study, which was one of the most notable one. (M et al., 2018) evaluated performances of RNN, LSTM, CNN and MLP, whilst (Selvin et al., 2017) compared CNN, LSTM, RNN and ARIMA. (Lee & Yoo, 2018) evaluated three RNN models for stock price forecasting (SRNN, LSTM and GRU), then also constructed a portfolio dependent on threshold with equities selected based on projections.

Meanwhile, a hybrid modelling technique was employed in several of the studies. In their research, (Liu et al., 2017b) used blend of CNN and LSTM in their study, (Batres-Estrada, 2015) used MLP with DBN to build an equity portfolio by forecasting log-return for each of the stock on a monthly basis and selecting those companies that were predicted to outperform the median stock. Also, several innovative techniques were used in some of the research. (Yuan et al., 2018b) presented a Deep and Wide Neural Network (DWNN) that combines RNN and CNN.

In another set of studies, some researchers employed historical price data, statistical and technical analysis, financial statements, macroeconomic data, news, investor sentiment, news and other sources of input parameters. (Kraus & Feuerriegel, 2017) used text mining with transfer learning to develop LSTM using sentiments from financial news and raw stock market price data. Using corporate action events and macroeconomic indexes, (Minami, 2018) employed LSTM to forecast the next day price of the stock.

So far, studies have used input parameters depending on the pre-selected asset, which might be raw price data, macroeconomic data, financial statements, news, investor mood, and so on. Because our primary goal is to maximize our investment returns by effectively managing our portfolio, it is critical that we select financial assets depending on market trend. Rather of focusing just on forecasts and trading technique, it is critical to select high-quality assets. The intrinsic component of the asset itself, such as current volume, price change percent, market cap, volatility, proximity to 52W high, and so on, as well as its historical price, might influence

its quality. These inherent characteristics are also influenced by extrinsic variables like as investor sentiments, the asset's sector, and market momentum.

In the stock market, momentum and reversal effects are widespread and fascinating occurrences. The momentum effect states that equities that have done well(winners) in the past (i.e., have provided greater returns) would most likely continue to keep outperforming those that have had a bad performance in the past (i.e., have provided less returns hence losers) in the future. The reverse effect, on the other hand, implies that previous losers may become winners in the future. (De BONDT & Thaler, 1985) were the first to notice the reverse effect in their studies, finding that purchasing losers and selling winners on the US stock market might result in higher profits since the US market readily overreacts to certain events, resulting in abnormal price fluctuations. Researchers found that stock markets in different regions have varied degrees of momentum and reversal impact, in addition to the US market (s). The market condition has a substantial link with the momentum impact on the Indian stock market, according to study by (Hameed & Ting, 2000).

3 Research Questions

The research problem statement is made up of the following questions to aid in describing machine learning models and the algorithms behind them to business users and management for greater acceptability.

1. What selection strategy to use to select quality assets for the portfolio?
2. Which model to use as the base forecasting model?
3. What back testing strategy to use?
4. Which Evaluation Criteria to use?
5. Does the quality of financial assets matter if we keep the base forecasting model performance as constant?

4 Aims and Objectives

This study is aimed at building a strategy to optimise the financial asset portfolio in order to achieve maximum returns based on a chosen period. More emphasis is given on choosing the assets itself based on their intrinsic attributes along with their historical price data so that depending on the momentum of the market, we always choose quality assets with positive momentum. Most of these attributes can be calculated using the historical price data.

Various Traditional and Deep Learning based approaches will be compared for this asset selection strategy to see how the returns of the selected assets compare to fixed indexes like NIFTY50 and SENSEX.

The following points form the research objective for this study:

- Perform Feature-Engineering and come up with intrinsic attributes for the assets.
- Build portfolio using the assets chosen based on the new attributes for the chosen period.
- Build different traditional and Deep Learning approaches to predict the asset price for chosen period.
- Evaluate and compare the different approaches
- Compare the returns % achieved using our portfolio against the standard indexes

5 Significance of Study

Financial portfolio management is a well-studied financial application field by both scholars and investors, with the primary goal of maximizing financial returns. We've seen the use of ML models like ANNs, ECs, GP, and Agent-based models in the financial time series forecasting space. With the advent of DL strategies for financial forecasting research, the financial community received a new push recently, and a slew of new articles resulted. We found that DL models outperformed their ML counterparts in the great majority of trials. DMLP, RNN, LSTM, CNN, RBM, DBN, AE, and DRL are some of the DL models described in the literature.

In all of these studies so far we saw that the main research objective was to increase the returns for a chosen specific financial assets which were kept constant and improving the performance of the financial price forecasting algorithm. Also, the dataset which were chosen for these studies contained only the historical price data for the assets and in some cases the user sentiment data extracted from different sources. The quality of the financial asset itself was not given much significance in these studies. One more thing to note is that majority of these studies have been done on US and Chinese Stock market and a very small number of studies have used some specifically chosen financial assets from Indian Stock Market. Based on historical data we have seen that the Indian Stock market is highly volatile compared to the volatility of the US and Chinese counterparts.

In this study, we plan to give more emphasis on the quality of the financial assets itself by deriving various intrinsic attributes for the given assets which can then be used to identify and filter quality assets on top of which we can apply the financial time series forecasting models and evaluate their performance. As we have seen that the DL models were superior to their ML counterparts, we will be using various DL models to compare and evaluate the returns. While deciding on the financial assets for investments, we have to look at the stock market's momentum and reversal impacts. To our knowledge, only a small amount of research has employed machine learning to address this problem.

6 Scope of Study

This study will focus on the financial assets which belong to the Indian Stock Market only. This includes top equities i.e. individual stocks and indices like NIFTY and SENSEX. The main objective of this study is to optimize the financial portfolio given a period of time to choose quality assets for investments. These assets will be chosen based on the intrinsic attributes which will be derived based only on the base attributes of the financial assets.The base attributes of these assets include only the historical price data of the assets including volume in market.

After these financial assets have been filtered and allocated, the financial time series price prediction model will be deployed on top of them for a certain time period to evaluate performance using evaluation metrics such as MAE, MAPE, RMSE, and Return/Profit percent. We will compare and analyse the returns using various DL models, since we have observed in prior research that DL models outperform their ML counterparts.

We will not be using any traditional ML based models like ANNs, ECs, AEs etc. for the scope of this study as we have already seen that DL models outperform their ML counterparts based on previous researches.

We will only be considering features for asset selection and model building which can be derived/extracted from the base attributes (OHLCV). Any other macroeconomic data, environmental considerations, sector information, geographical or political factors are out of the scope.

6.1 Limitations of Study

The focus of this study is limited to the stocks chosen(top 10 of each sector) as the dataset. Since this dataset is being acquired from a trusted source like yahoo finance, it has not been validated against other sources and is assumed to be valid. The data used in this study is focused on India only. Hence, the conclusions may change for the same experiments if done in different countries.

7 Research Methodology

7.1 Introduction

The main objective of this study is to first build our portfolio with high quality assets chosen based on various features extracted from raw price (OHLCV) data. Based on the momentum of the assets, we can choose the winning stocks and bet on the momentum effect and/or go with the losing stocks if we see a reversal effect trend based on historical data. Once the stocks have been finalized for our portfolio, we will evaluate and compare the DL models for price forecasting based on returns/profits achieved. The same models will be built using standard market indexes (NIFTY50, SENSEX) in order to answer our research question i.e. “Does the quality of financial assets matter if we keep the base forecasting model performance as constant?”

Fig 1. depicts the research methodology on a step-by-step manner

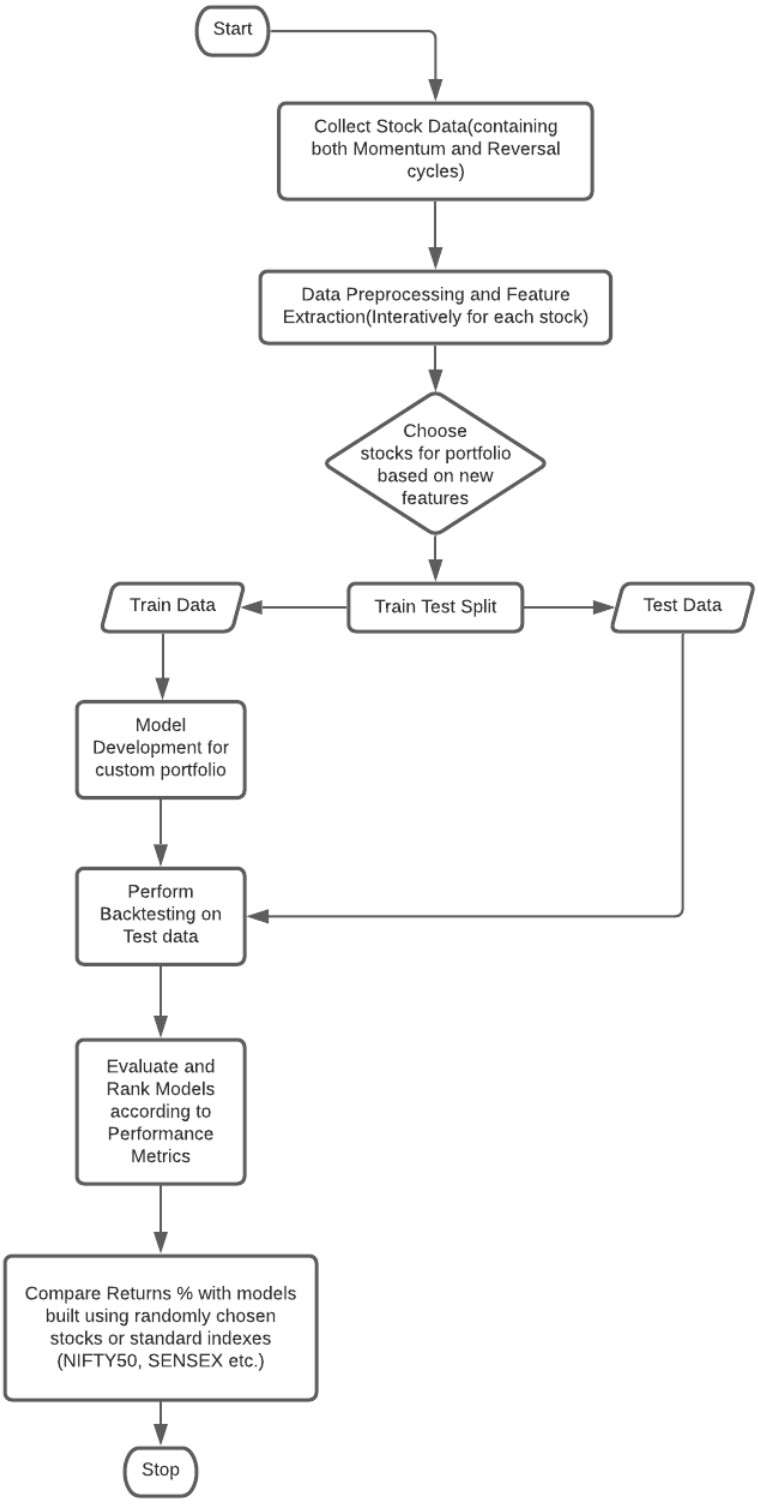

Start
Collect Stock Data(containing both Momentum and Reversal cycles)
Data Preprocessing and Feature Extraction(Interatively for each stock)
Choose stocks for portfolio based on new features
Train Test Split
Train Data
Test Data
Model Development for custom portfolio
Perform Backtesting on Test data
Evaluate and Rank Models according to Performance Metrics
Compare Returns % with models built using randomly chosen stocks or standard indexes (NIFTY50, SENSEX etc.)
Stop

Fig. 1

7.2 Dataset Description

The data came from Yahoo Finance, a website that offers financial news, statistics, and analysis, including stock quotations, press announcements, and financial reports.

The top 10 equities from each sector from the Indian Stock Market will be utilized as the dataset for equites. We will be using the raw price data(OHLCV) to derive/extract more features which can play a significant role while deciding on the stocks for our investment portfolio.

Column Summary:

Base Attributes from dataset :

Date	Date of the record
Open	Opening Price
High	High for the day
Low	Low for the day
Close	Closing Price
Volume	Total Volume

Table 1

7.3 Data Preparation and Pre-Processing

The selected ranges of training and testing data encompassed at least one entire market cycle in order to assess the models' performance in diverse market conditions (i.e., bullish, bearish, and oscillating markets). The following are the key pre-processing processes that will be followed at this stage of the study:

- Check for missing/NaN values and treat them if necessary.
- Perform feature engineering to generate new derived features using the base features.
- Perform some basic Exploratory Data Analysis using the new derived features.

7.4 Feature Extraction and Engineering

Since selection of quality assets for our portfolio is an important requirement for this study, we will be applying Feature Engineering to generate new derived features which can be potentially helpful in selection of financial assets qualitatively.

Financial data is unique and needs a methodical approach. First, in compared to datasets used in applications of ML, it is fairly restricted in availability and breadth — in fact, we don't have high-quality data for the great majority of markets prior to the 1990s. Many crucial factors that drive results can be left out of a model or don't get to see the entire range of their values during training. The significantly divergent behaviour of asset prices during high and low market volatility periods, or regimes, is a specific illustration of this difficulty; training the model on a subset of the data that spans only one of the regimes will degrade the model's capacity to generalize well on the test set. The signal-to-noise ratio of financial data is quite low. Models will overfit the data noise because the most powerful models, such as neural networks, which are low bias and high-variance learners. Surely, additional factors such as macroeconomic statistics, sentiment related to monetary policy announcement, and so on might help to describe the overall situation of the market.

Following are some of the derived features which have been identified. This list will be updated and new features may be added during feature extraction and engineering phase of the Research Plan.

Derived Attributes :	
Close Delta	Daily Change in Closing Price
Close Change %	% Daily Change in Closing Price
Market Cap	Daily Market Capitalization
Volume Delta	Daily Change in Volume
Volume Change %	% Daily Change in Volume
Weekly Volatility	Close – Weekly Close Avg.
Monthly Volatility	Close – Monthly Close Avg.
Yearly Volatility	Close – Yearly Close Avg.
Nearness to 52W High	Close – Yearly Close Max

Table 2

7.5 Model Development

Once we have finalized the stock pool which we will be building our portfolio with based on the previously derived features, we will be employing various DL based models to forecast the prices for the upcoming period using the extracted/derived features.

7.6 Model Evaluation Metrics

The following measures will be used to evaluate all models under consideration:

- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Root Mean Square Error (RMSE)
- Profit/Returns %

In addition to the indicators listed above, the temporal complexity of the models may be taken into account while assessing them, i.e. the overall amount of features provided, as well as the model chosen, might result in varying levels of temporal complexity, which is important when the models are deployed into production. Since some business decisions require the model response time to be as low as possible, the time complexity of the model becomes an important factor.

8 Resource Requirements

8.1 Hardware Requirement:

This study will be carried out on a desktop computer that meets the following requirements:

Processor	Intel® CoreTM i9-9900K CPU @ 5.0 GHz
Memory	32 GB DDR4
GPU	RTX 2080 (if required)
Operating System	Windows 10 64-bit

Table 3

8.2. Software Requirements:

- Python 3.x
- Jupyter Notebook
- Libraries required for general data analysis and EDA like numpy, pandas, matplotlib
- Libraries required for pre-processing, model selection, feature selection like sklearn
- Libraries required for evaluation metrics
- Any other libraries as and when required

9 Risk and Contingency Plan

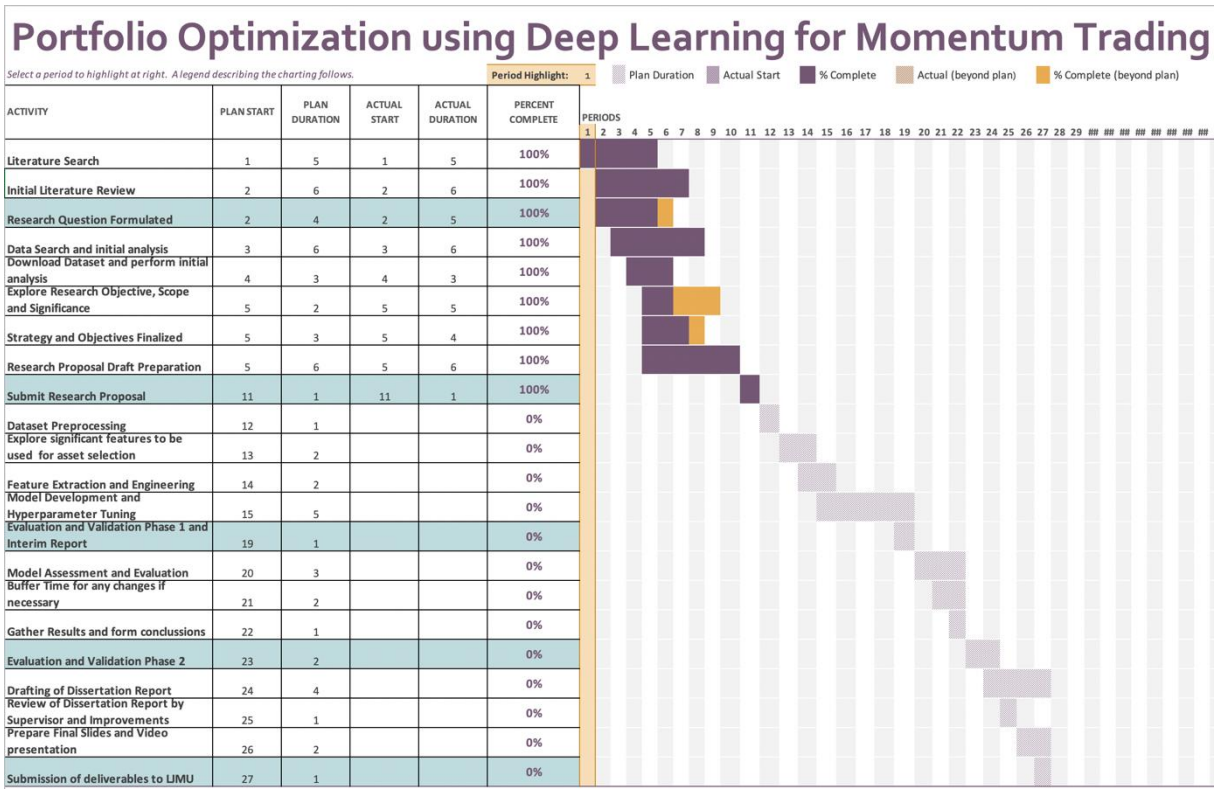
Table 4 summarizes the risks involved and contingency plan:

Risk	**Contingency**	**Severity**
Literature availability for the research process	Literature Review has been done during the creation of this proposal and same will continue throughout the Thesis Preparation. A good amount of literature has been covered in case there is any challenge during next steps	Medium
Data Quality	The dataset for this is a standard dataset which has been used in many different studies.	Low
Hardware/Software failure	Documentation and code will be maintained in a Version Control system	Low
Unforeseen Professional and Personal commitments	The Research Plan has been designed to have some buffer time before each milestone	Low
Research and Thesis quality below the expectations of the supervisor and university	Regular interactions on the research progress and review meetings for the thesis has been planned with the thesis supervisor	Low

Table 4

10 Research Plan

The research plan has been created while keeping in mind the research methodology and the submission milestones. This plan is spread across 1st June – 7th Dec which is a span of 27 weeks.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature Search	1	5	1	5	100%
Initial Literature Review	2	6	2	6	100%
Research Question Formulated	2	4	2	5	100%
Data Search and initial analysis	3	6	3	6	100%
Download Dataset and perform initial analysis	4	3	4	3	100%
Explore Research Objective, Scope and Significance	5	2	5	5	100%
Strategy and Objectives Finalized	5	3	5	4	100%
Research Proposal Draft Preparation	5	6	5	6	100%
Submit Research Proposal	11	1	11	1	100%
Dataset Preprocessing	12	1			0%
Explore significant features to be used for asset selection	13	2			0%
Feature Extraction and Engineering	14	2			0%
Model Development and Hyperparameter Tuning	15	5			0%
Evaluation and Validation Phase 1 and Interim Report	19	1			0%
Model Assessment and Evaluation	20	3			0%
Buffer Time for any changes if necessary	21	2			0%
Gather Results and form conclussions	22	1			0%
Evaluation and Validation Phase 2	23	2			0%
Drafting of Dissertation Report	24	4			0%
Review of Dissertation Report by Supervisor and Improvements	25	1			0%
Prepare Final Slides and Video presentation	26	2			0%
Submission of deliverables to LJMU	27	1			0%

Fig. 2

APPENDIX C: INTERIM REPORT

Portfolio Optimization using Deep Learning for Momentum Trading

Sourav Patel

Mid Thesis Report

Liverpool John Moores University – Master's in Data Science

October 2021

Abstract

Financial portfolio management refers to the continuous reassignment of funds into financial assets like cash, stocks, bonds, mutual funds, bank deposits and commodities. The financial portfolio is created based on the investors' risk tolerance, investment objectives, and time period. The price of each asset has an impact on the portfolio's risk/reward ratio. It's also known as portfolio asset allocation. Because the investing world is so volatile, portfolio management includes asset allocation and rebalancing. With the fast changes that occur throughout time, one sector may become less popular than another, and vice versa. As a result, only a nimble investor can benefit handsomely from the world's stock markets.

To tackle the challenge of Financial Time Series Forecasting, researchers devised a variety of Machine Learning models, and a lot of papers have been published as a result. Despite the increased interest in creating financial time series forecasting models, few review papers have focused on optimizing financial portfolios to maximize returns. As a result, the goal of this paper is to develop a method for managing our portfolio assets so that we may maximize profits depending on market momentum.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
OHLCV	Open-High-Low-Close-Volume
DFNN	Deep Feedforward Neural Network
FX	Forex
ML	Machine Learning
ANN	Artificial Neural Network
EC	Evolutionary Computation
GP	Genetic Programming
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
DMLP	Deep Multilayer Perceptron
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
AE	Autoencoder
DRL	Deep Reinforcement Learning
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
NLP	Natural Language Processing
ARIMA	Autoregressive Integrated Moving Average
SRNN	Stacked Recurrent Neural Network
DWNN	Deep and Wide Neural Network
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
EDA	Exploratory Data Analysis
VWAP	Volume Weighted Average Price
PLR	Piecewise Linear Regression
ES	Exponential Smoothening

RMSprop	Root Mean Squared Propagation
ReLU	Rectified Linear Unit
MSE	Mean Squared Error
SAE	Stacked Auto Encoder
WT	Wavelet Transform
AE	Auto Encoder
RAF	Random Forest
LOG	Logistic Regression Classifier
DFP	Deep Feature Policy
NMSE	Normalised Mean Squared Error
NL	Neuron Level
SL	Synapse Level
FNN	Feedforward Neural Network
DJIA	Dow Jones Industrial Average
NYSE	New York Stock Exchange
DWNN	Deep and Wide Neural Network
SFM	State Frequency Memory
DFT	Discrete Fourier Transform
IFT	Inverse Fourier Transform
BPTT	Back-Propagation Through Time
AR	Autoregressive
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
MAD	Mean Absolute Deviation
VAE	Variational Autoencoder
MCC	Matthews Correlation Coefficient

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Portfolio management is the dynamic cycle of consistently redistributing a measure of financial assets into various diverse monetary investment products, where the objective is to boost the returns while limiting the risk as much as possible. According to the study by (B. Li & Hoi, 2013), Traditional portfolio management strategies can be categorized into four classes, “Follow the Winner”, “Follow the Loser”, “Pattern Matching”, and “Meta Learning”

“Follow the Winner”, attempts to asymptotically accomplish a same growth rate (expected log return) as that of an ideal strategy. According to the “Follow-the-Loser” strategy, wealth should be transferred from winning assets to losing ones, which seems contradictory as per common sense but according to multiple observations, it often generates remarkable results. “Pattern Matching” algorithms anticipate the subsequent market distribution based on the sampled distribution and explicitly optimizes the portfolio using the sampled distribution as per study by (Gyorfi et al., 2006). Lastly, “Meta Learning” strategy consolidates multiple methodologies of different classifications to achieve more consistent and predictable results (Das & Banerjee, 2011).

Many Deep Learning based approaches have been explored for financial stock market prediction so far. However, the majority of them attempt to forecast price changes or trends for a selected number of assets where the focus is on the said assets only rather than the momentum of the market itself. As per design, a neural network generates an output vector of asset prices for the upcoming time period using just historical raw price data (OHLCV) of the financial assets as input. The trading agent can then take action based on the prediction. This can simply be termed as a supervised learning regression problem, and it is straightforward to implement. As a result, the accuracy of these price prediction-based algorithms is largely dependent on their performance, and it turns out that capturing future market momentum is challenging.

In order to manage the portfolio efficiently to maximize returns based on the momentum of the market, it is essential to choose quality assets themselves rather than only focus on the trading strategy itself. The quality of assets may depend on the intrinsic factor of the asset itself like Current Volume, Price Change %, Market Cap, Volatility, Nearness to 52W High etc. along

with its historical price itself. These intrinsic attributes also depend on some extrinsic factors like the sentiments regarding the asset, the sector of the asset and the market momentum itself.

The most frequently researched financial application field is financial time series forecasting, specifically asset price forecasting where the major focus is always on predicting underlying asset's next movement. This was the focus of the majority of the existing DL implementations. Despite the existence of several sub-problems like price forecasting for Stocks, Indexes, Forex, Cryptocurrency, Bonds, Commodities, Volatility and many more, the fundamental elements in all of these applications are very similar.

Based on their dependent output, which can be either price prediction or price movement prediction, the studies can be divided into two groups. Although price prediction can be termed as a fundamental regression problem, in majority of its applications, correctly recognizing the directional movement or momentum is more essential than precise price predictions. Due to this, analysts regard trend forecasting, or anticipating the directional movement of the price, to be more important field of research than precise prediction of price. In this case, trend/momentum prediction turns into a classification problem. As per certain studies, only up and down price movement (2-class problem) are taken into account but neutral movement is also taken into account in others (3-class problem).

DL models like LSTM and its variations, as well as certain hybrid models, have dominated the financial time series forecasting arena. Because LSTM by design is built to exploit the temporal features of any time series data, identifying financial time series signals is a closely examined and useful application. Some analysts, on the other hand, choose to extract relevant characteristics from time series data or modify time series data such that following financial data is fixed in time. This means that the model can be trained correctly and better test performance can be achieved on unseen data regardless of the data order. DFNN and CNNs were the primarily used models in those implementations.

1.2 Aims and Objectives

This study is aimed at building a strategy to optimise the financial asset portfolio in order to achieve maximum returns based on a chosen period. More emphasis is given on choosing the assets itself based on their intrinsic attributes along with their historical price data so that depending on the momentum of the market, we always choose quality assets with positive momentum. Most of these attributes can be calculated using the historical price data.

Various Traditional and Deep Learning based approaches will be compared for this asset selection strategy to see how the returns of the selected assets compare to fixed indexes like NIFTY50 and SENSEX.

The following points form the research objective for this study:

- Perform Feature-Engineering and come up with intrinsic attributes for the assets.
- Build portfolio using the assets chosen based on the new attributes for the chosen period.
- Build different traditional and Deep Learning approaches to predict the asset price for chosen period.
- Evaluate and compare the different approaches
- Compare the returns % achieved using our portfolio against the standard indexes

1.3 Scope of Study

This study will focus on the financial assets which belong to the Indian Stock Market only. This includes top equities i.e. individual stocks and indices like NIFTY and SENSEX. The main objective of this study is to optimize the financial portfolio given a period of time to choose quality assets for investments. These assets will be chosen based on the intrinsic attributes which will be derived based only on the base attributes of the financial assets.The base attributes of these assets include only the historical price data of the assets including volume in market.

After these financial assets have been filtered and allocated, the financial time series price prediction model will be deployed on top of them for a certain time period to evaluate performance using evaluation metrics such as MAE, MAPE, RMSE, and Return/Profit percent.

We will compare and analyse the returns using various DL models, since we have observed in prior research that DL models outperform their ML counterparts.

We will not be using any traditional ML based models like ANNs, ECs, AEs etc. for the scope of this study as we have already seen that DL models outperform their ML counterparts based on previous researches.

We will only be considering features for asset selection and model building which can be derived/extracted from the base attributes (OHLCV). Any other macroeconomic data, environmental considerations, sector information, geographical or political factors are out of the scope.

1.4 Significance of Study

Financial portfolio management is a well-studied financial application field by both scholars and investors, with the primary goal of maximizing financial returns. We've seen the use of ML models like ANNs, ECs, GP, and Agent-based models in the financial time series forecasting space. With the advent of DL strategies for financial forecasting research, the financial community received a new push recently, and a slew of new articles resulted. We found that DL models outperformed their ML counterparts in the great majority of trials. DMLP, RNN, LSTM, CNN, RBM, DBN, AE, and DRL are some of the DL models described in the literature.

In all of these studies so far we saw that the main research objective was to increase the returns for a chosen specific financial assets which were kept constant and improving the performance of the financial price forecasting algorithm. Also, the dataset which were chosen for these studies contained only the historical price data for the assets and in some cases the user sentiment data extracted from different sources. The quality of the financial asset itself was not given much significance in these studies. One more thing to note is that majority of these studies have been done on US and Chinese Stock market and a very small number of studies have used some specifically chosen financial assets from Indian Stock Market. Based on historical data we have seen that the Indian Stock market is highly volatile compared to the volatility of the US and Chinese counterparts.

In this study, we plan to give more emphasis on the quality of the financial assets itself by deriving various intrinsic attributes for the given assets which can then be used to identify and filter quality assets on top of which we can apply the financial time series forecasting models and evaluate their performance. As we have seen that the DL models were superior to their ML counterparts, we will be using various DL models to compare and evaluate the returns. While deciding on the financial assets for investments, we have to look at the stock market's momentum and reversal impacts. To our knowledge, only a small amount of research has employed machine learning to address this problem.

1.5 Structure of the Study

In this study, we start with the literature review of the currently existing Deep Learning Solutions for solving the problem of Stock market prediction. Then we will proceed with explaining our methodology. Based on the Literature Review, we will be finalizing on the models to go forward with for our use case. We will then proceed with implementing the chosen models on our chosen portfolio and evaluating them according to our chosen evaluation metrics. We will also use the models to evaluate the returns achieved in a portfolio with random stocks or standard indexes like NIFTY, SENSEX etc.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

After doing literature review extensively, we have filtered out the studies which we felt were relevant and suggested solutions comparable to the current state-of-the-art solutions. A brief overview of these studies are as follows:

2.2 Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance(Chandra & Chand, 2016)

2.2.1 Business Problem

For time series prediction issues, the combination of soft computing approaches such as evolutionary algorithms and NNs has yielded extremely promising results.
Cooperative coevolution breaks down a problem into isolated subcomponents, and collaboration usually requires fitness evaluation. Developing efficient subcomponent decomposition algorithms for various neural network topologies has been a problem. The Two decomposition approaches have been compared and evaluated for training feedforward and RNNs for chaotic time series in this study. We also use them to solve financial forecasting challenges from NASDAQ. The constraints of real-time implementation are discussed, as well as a mobile application architecture for predicting financial time series. In general, the results demonstrate that for real-world time series issues, recurrent neural networks outperform feedforward networks in terms of generalisation.

2.2.2 Research Methodology

This paper examines the effectiveness of coevolutionary techniques for training FNNs and RNNs to predict chaotic time series data. Elman recurrent networks are described in detail initially, followed by cooperative neuro-evolution and time series issues.

As illustrated in Fig. 1, Elman RNN is made up of an input layer followed by a context layer that provides state information followed by a hidden layer and finally an output layer. Each

layer has one or more neurons that use a non-linear function of their weighted sum of inputs to transmit information from one layer to the next.

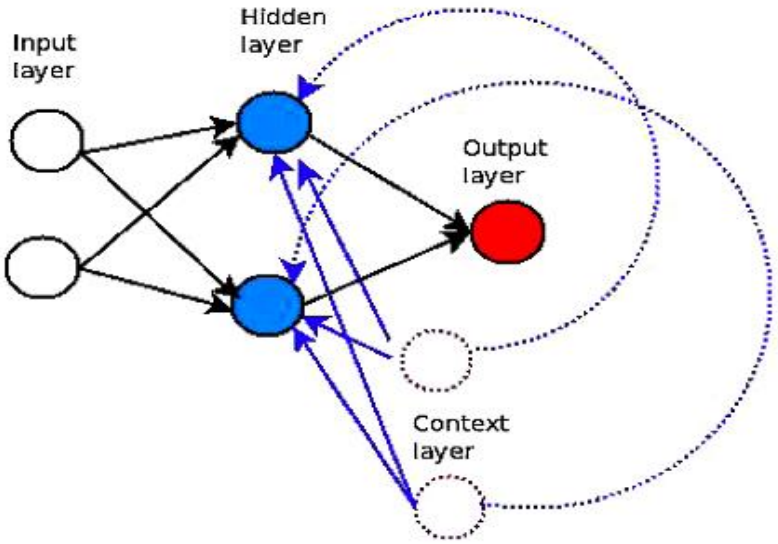


Fig. 1: Elman RNN

Elman RNNs have been trained utilizing cooperative neuro-evolution and back-propagation over time which have proved to be extremely promising for time series prediction issues.

Cooperative coevolution utilizes issue decomposition techniques that deconstruct the network architecture into multiple small components. It also specifies how weights of subcomponents are represented that are implemented as sub-populations. NL and SL issue decomposition are the two most used problem break down approaches.

The neural network is decomposed to its simplest level in SL problem decomposition, with each weight link (synapse) becoming a subcomponent. The neurons in the network serve as the decomposition's reference point in NL problem decomposition. Enforced sub-populations and neuron-based sub-populations are two examples.

A huge number of data points are collected regularly at same intervals for time series prediction. We must alter them in order to use soft computing approaches such as NNs to construct prediction models. The time series must be split/sliced down into smaller chunks that are acquired at regular intervals in order to do the fundamental transformation or reconstruction. Following is a detailed description of the procedure.

An embedded phase space Y(t)=[(x(t), x(t - T),..., x(t(D - 1)T)] can be generated from an observed time series x(t), where T represents time delay, D represents embedding dimension, t=0, 1, 2,..., N - D_T - 1 and N = original time series length. The vector series reproduces many important properties of the original time series, according to Taken's theorem. To use Taken's theorem effectively, the proper values for D and T must be chosen. It has been demonstrated that if the original attractor had dimension d, then D = 2d + 1 will suffice to recreate it.

To train the FNN and RNN the reconstructed vector is used for one-step-ahead prediction where 1 neuron is employed in the output layer. The number of input neurons in a feedforward network is denoted by D. In recurrent networks, 1 represents the number of input neurons. The embedding dimension D corresponds to the number of steps taken by the recurrent network in time.

The recurrent and feed-forward networks' prediction performance is measured using the RMSE. For further comparison with literature data, the NMSE is employed.

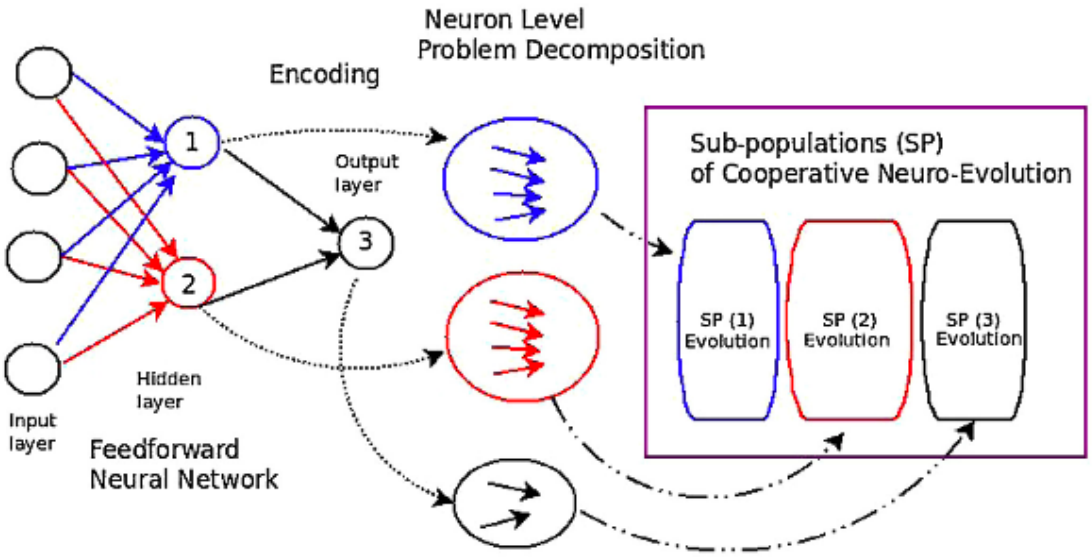


Fig 2 : Decomposition of NL problems using FNN. It's worth noting that the four input neurons indicate time series reconstruction with a four-dimensional embedding dimension.

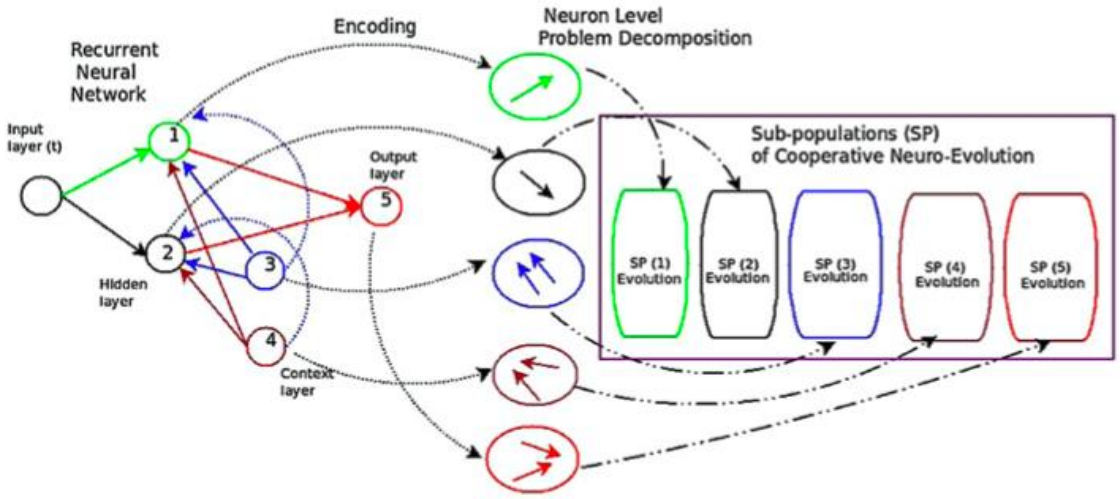


Fig 3 : With the NL problem decomposition, Elman RNN is used. It's worth noting that one input neuron can represent any embedding dimension for time series reconstruction. The network expands to the extent that the embedding dimension allows.

The NASDAQ stock exchange provided the financial time series data collection. It includes, , Staples Inc., Seagate Technology Holdings and ACI Worldwide daily closing prices. Each company's data set included closing stock prices from December 2006 to October 2010, totaling roughly 800 data points. Because stock prices varied over a three-year period, the dataset also includes data from the 2008 financial crisis in the United States. To create the training and testing datasets, we utilized Taken's theorem to rebuild the time series data with time lag T = 2 and embedding dimension D = 5.

The stock prices at the end of the day were scaled using min-max scaling. The algorithm's maximum function evaluations was set to 50,000, which served as a terminal condition. We utilized a population size of 300 people, same as the prior data sets. In the hidden and output layers of the corresponding neural network designs, sigmoid units were used.

2.2.3 Results

Their abilities are assessed in terms of training and generalization. The Mackey–Glass time series and the Lorenz time series are the two simulated time series in the benchmark chaotic time series issues. The Sunspot time series is a real-world situation.

For both issue decomposition approaches in the Mackey–Glass problem, the FNN was able to provide better generalization than the RNN. We note that the RNN showed higher

generalisation compared to FNN for the Lorenz issue. Furthermore, NL failed to outperform SL. When compared to the RNN, the FNN performed worse in the Sunspot issue. When NL was compared against SL for both network designs, NL performed better. In general, the results demonstrate that the NN with the lowest training error does not necessarily perform the best in terms of generalization. This is due to the dataset's over-fitting and noise. With fewer hidden neurons, performance in the Lorenz and Sunspot tasks was typically better. In the instance of the Mackey–Glass issue, however, this was not the case.

In two of the three benchmark problems, the RNN outperformed the FNN. Furthermore, for both neural network architectures, NL has given better performance when compared to SL in the majority of cases.

For both network designs, generalisation performance was extremely competitive. In terms of training and generalization, the RNN outperformed the FNN marginally. In addition, we found that the NL decomposition performed better than the SL decomposition. As the number of hidden neurons rose, so did the performance on NL.

RNN beat FNN in both generalization and training performance in several situations. We also found that the SL decomposition performed better than the NL decomposition for both neural network designs.

On both network designs, the RNN outperformed the FNN and the NL outperformed the SL. Because the training and generalization performance had a very significant variance, the offered approaches showed evidence of over-fitting with this problem. We also point out that training performance is far superior to generalization performance. This could be owing to a significant difference in the stock market's trajectory, which distinguishes the training dataset from the testing dataset.

In two out of the three experiments, NL outperformed SL. In a comparison of the two network topologies, we find that the RNN outperforms the FNN in terms of generalization. In all three experiments, the RNN outperformed the others. As a result, we can conclude that the RNN is better suited to financial prediction in general.

2.3 A Deep Learning Approach for Optimization of Systematic Signal Detection in Financial Trading Systems with Big Data(Arpaci & Karaoglu, 2017)

2.3.1 Business problem

The purpose of this research is to create an intelligent trading decision support system by applying recurrent neural network technique to financial trading subsystems in order to discover systematic signals in Big Data. The model was evaluated using data from the Istanbul Stock Exchange in this study.

2.3.2 Research Methodology

Amazon (S3) is used to store the daily stock market data compressed in gzip format. Because the algorithms demand a variety of types and forms, the data is in a raw format; it is their responsibility to format certain day intervals correctly.
We utilized the stock data of Istanbul Stock Exchange-Nasdaq suite's standard raw format. We used the full stock trading data from the year of 2016 in order to train our model for this study's assessments.

In the construction of successful machine learning models, choosing input parameters is crucial. Market microstructure features should be used to train machine learning models that have a substantial influence on the market which in turn allows learning the policy to condition and enhance forecasting accuracy. We discovered that the properties stated in Table 1 are beneficial in our model, despite the fact that there are many possible parameters that might be employed in ML models.

Feature	Description
TradeCount15, TradeCount30, TradeCount60	trade volume within 15,30,60 sec time-step
Spread	Bid vs Ask price difference
PriceVwapDiff, PricePeriodicVwapDiff	differences of the price vs Real-time VWAP, differences between price vs periodic VWAP
BidAskTotalTradeVolumeDiff	Difference of trades happening at Ask price and trades happening at Bid price
TradeVol15, TradeVol30, TradeVol60	volume in 15,30,60 sec time-step
lambda15, lambda30, lambda60	Kyle’s Lambda values under

	15,30,60 sec time-step
FirstLevelImbalance, SecondLevelImbalance, ThirthLevelImbalance, ForthLevelImbalance, FifthLevelImbalance	5 Levels of Orderbook Imbalance (BidVolume-AskVolume) / (BidVolume+AskVolume)
Volatility	Daily highest - lowest price
Pt15, Pt30, Pt60	difference of sum of trades happening at the Ask vs Bid prices under 15, 30, 60 sec time-step x Kyle's Lambda values
DifTotalTurnoverPercent, DifTotalTurnover	Bid Turnover – Ask Turnover and Turnover %
BidOrAskDirection	Bid or Ask Trade relization
OpenChangePercent, LowChangePercent, HighChangePercent, PriceOpenDiff, PriceHighDiff, PriceLowDiff	previous closing price - current day lowest and highest opening prices, day change %
TotalVolume	total volume per day
DailyChangePercent	closing price – yesterday's closing price

Table 1: Features used for model

PLR is a basic modelling approach that allows for dynamic incremental updates in a time series. The number of line segments may be adjusted dynamically to alter the granularity of PLR approximation (Keogh et al., 2001). The rapid changes in trade points might make it difficult to identify and utilize in prediction systems. As a result, we employ PLR as a sliding window to detect falls and spikes in stock market data. Our method uses PLR to identify turning moments as trading signals.

A complete gradient and bidirectional version of LSTM, Graves LSTM (Graves & Schmidhuber, 2005), (Graves, 2014), is used in the system for data modelling. LSTM (Hochreiter & Schmidhuber, 1997), (Gers et al., 2003) is an effective approach for analyzing serial data with gaps of uncertain size between key occurrences as a form of recurrent neural network (RNN). Compared to standard hidden Markov models and RNNs, the major benefit

of LSTM is its relative insensitivity to lag duration. By adding a network design incorporating memory blocks, LSTM overcomes the problem of backpropagation faults (Hochreiter et al., 2001) in standard RNNs with lengthy time delays. The input, output, and forget gates make up the RNN memory blocks. With the activation of the input gate, each cell doubles the input. The forget gate multiplies the preceding cell values. The output gate multiplies the output to the network. As a result, the network's cells communicate through gates (Graves & Schmidhuber, 2005). Graves LSTM is used in our system since it performs better than the original LSTM (Graves, 2014).

Furthermore, our system employs the Exponential Smoothing method. This is a common technique for applying low-pass filters in order to eliminate short-term oscillations while looking at time series data. ES maintains the longer-term trend by flattening noise. In contrast to the moving average, which assigns equal weights to prior data, it reduces weights exponentially with time.

2.3.3 Results

We used trade data of Garanti Bank from September 2016 to January 2017 to analyze the performance of our model, which corresponds to the fourth quarter of 2016. During the time period, the data comprised about 400.000 transactions. To feed each trade into LSTM, we used a sliding window method. In a sliding window method, we employed a total of 20 timesteps for 36 characteristics. PLR with a price limit of 0.06 points were our goal values.

Following parameters were discovered to be a good fit for our three-layer model after examining multiple different parameter combinations inside the hyperparameter set. We used the Xavier init technique to set their weights. Graves LSTM is the first two layers, followed by RNN output layer. With width of the Graves LSTM as 32, and the RNN layer has single output per 20 timesteps. The learning rate was set to 0.1, taking RMSProp as updater function, and ReLU as activation function. The training phase started with a MSE of 14 which decreased down to 0.001.

When we used a batch size of 20, we found that the error rate fell regularly. The error function became noisy when the number was less than 20. When evaluating the standard deviations of gradients, all layers looked to have almost similar learning rate. Subsequently, the model's accuracy was enhanced by first performing standardization and then using min-

max normalization. We selected the first quarter of 2017 as the test period, resulting in an almost 50 percent train-test split.

2.4 Deep learning with long short-term memory networks for financial market predictions(Fischer & Krauss, 2017)

2.4.1 Business Problem

Financial time series prediction problems are notoriously tough, owing to the high noise level and the semi-strong form of market efficiency. The financial models used to create a link between these return prediction signals (features) and future returns (targets) are often transparent and incapable of capturing complicated non-linear relationships. Non-linear patterns in financial time series data can be recognized using ML techniques, according to preliminary evidence as per (Huck, 2009), (Takeuchi, 2013), (Moritz & Zimmermann, 2016), (Dixon et al., 2015). For the sake of comparison, this study uses the same dataset as that of (Krauss et al., 2017) and builds on his work.

In this work, focus is largely on deep learning and how it may be used to solve a large-scale financial time series forecasting problem. Three additions to the literature have been made in this regard.

- LSTM networks, for example, are one among the most sophisticated DL architectures for sequential learning applications including speech recognition, handwriting recognition, and time series prediction. More emphasis on offering a comprehensive guide to data preparation, also the creation and training for financial time series prediction tasks, or the impact of adding news for specific businesses. Finally, compare the results against a random forest, to demonstrate the value-added by the LSTM, and a standard LOG model from the literature (to establish a baseline).
- Find the stocks chosen for trading which have below-average momentum, significant volatility, extreme directional moves in the days leading up to trading and a propensity to reverse these extreme movements in upcoming time period.
- Finally, combine the data from the previous section into a simpler, rules-based trading strategy that attempts to capture the core patterns that LSTM uses to identify successful and losing stocks.

2.4.2 Research Methodology

To begin, the raw data is first divided into study periods, each of which includes training sets(used for in-sample training) and trade sets (used for out-sample training). Next, go through the essential parameter space and objectives for training and prediction.

A "study period" is defined which is a training-trading set that consists of a 750-day training phase and a 250-day trading period. Then, from 1990 to 2015, divide the full data set into 23 research periods with non-overlapping trade periods.

For the training set, we take into account all stocks that are S&P 500 constituents on the last day of the training period that have available historical data. Some stocks show the whole 750-day training history, while others only show a fraction of it, such as when they are listed later. If a component has no price data after a specific trading day throughout the trading period, it is regarded for trade until that day.

For training, LSTMs need sequences of input parameters, or the values of the input parameters at different time-steps. The standardized one-day return is our only feature. We choose a series length of 240, which corresponds to about one trading year's worth of data. As a consequence, we have 240 standardized one-day returns in overlapping sequences.

The author has followed (Takeuchi, 2013) and designed a binary classification issue, in which the output variable for each stock s and date t can take two distinct values for the sake of comparison. To determine the two classes, the one-period returns of all stocks s in period t+1 are ordered and divided into two equal-sized classes. If the cross-sectional median return of all stocks in period t+1 is more than one-period return of stock s, class 0 is achieved. Similarly, if the cross-sectional median is less than or equal to one-period return of stock s, class 1 is attained.

The study uses keras to train the LSTM network and has used three sophisticated techniques. RMSprop is used as an optimizer first, followed by dropout regularization in the recurrent layer. Early halting is often employed as a safeguard against overfitting.

The following is the topology of our trained LSTM network:

- One parameter and 240 time-steps in the input layer.
- an LSTM layer with dropout of 0.1 and 25 hidden neurons.
- A typical setup includes an output dense layer consisting of 2 neurons with a softmax activation function.

The LSTM model is benchmarked against a Radom Forest, DNN and a Logistic Regression Model. For two reasons, the author has chosen a random forest as a benchmark. For starters, it's a cutting-edge ML model that takes almost minimal tuning and consistently produces good results. Also, in (Krauss et al., 2017) and (Moritz & Zimmermann, 2016) - a large-scale ML application using monthly stock market data - RAF in this configuration are the best single approach and the method of choice. As a result, random forests are an excellent baseline for every new machine learning model.

To demonstrate the relative benefit of LSTM networks, a typical DNN is used. A FNN is used with 31 input neurons followed by 31 neurons in the 1st hidden layer followed by 10 neurons in the second hidden layer followed by 5 neurons in the 3rd hidden layer and lastly 2 neurons in the output layer. Following (Goodfellow et al., 2013), the activation function is maxout with two channels and softmax in the output layer. L1 regularization is used with 0.00001 shrinkage, and dropout is set to 0.5.

A logistic regression model has also been used as a baseline model. Details regarding our implementation may be found in the sci-kit learn manual (Pedregosa et al., 2011) and its references. L2 regularization is chosen from a set of 100 options on a log scale ranging from 0.0001 to 10,000 using 5-fold cross-validation on the training set, and LBFGS is used to find an optimum with a maximum of 100 iterations. In compared to a conventional classifier, we use logistic regression as a baseline to calculate the additional impact of the far more sophisticated and computationally demanding LSTM network.

2.4.3 Results

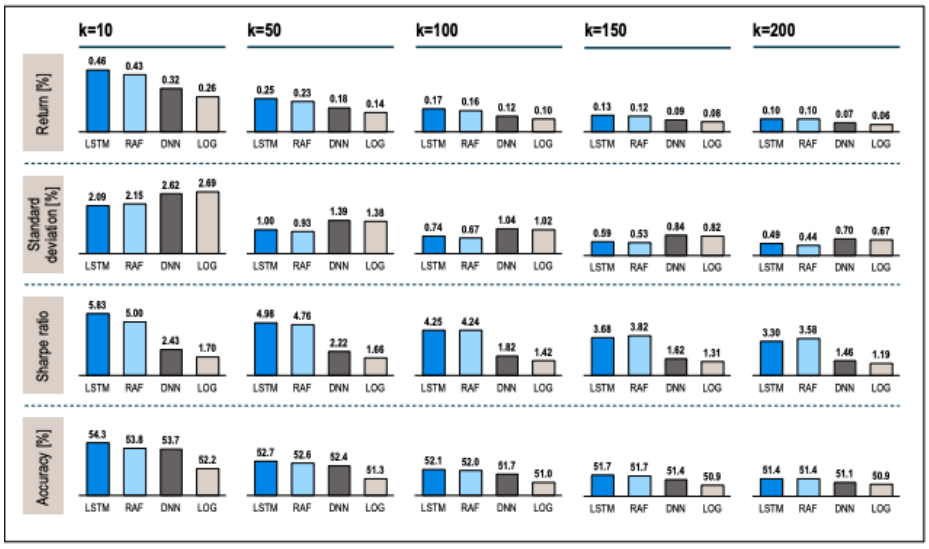


Fig 4: Performance characteristics for long-short portfolios of various sizes on a daily basis

k = portfolio equity size

Regardless of the portfolio size k, the LSTM outperforms the other methods. For k = 10, daily returns before transaction costs are 0.46 %, against 0.32 % for the DNN, 0.26 % for the Logistic Regression and 0.43 % for the RAF. Except for k = 200, when it is tied with the RAF, the LSTM also gets the greatest mean returns per day for bigger portfolio sizes. The LSTM and the RAF are similar in terms of standard deviation, a risk metric, with slightly smaller values for k = 10 and slightly greater values for rising portfolio sizes. Across all values of k, both LSTM and RAF have a considerably smaller standard deviation than DNN and LOG. The LSTM has the greatest return per unit of risk(Sharpe ratio), up to k = 100, and slightly less than the RAF for even bigger portfolios, when the RAF's lower standard deviation exceeds the LSTM's greater return. An essential machine learning measure is accuracy, which refers to the percentage of correct classifications. The LSTM has a clear advantage for the k = 10 portfolio, a small advantage till k = 100, and a tie for rising sizes with the RAF.

2.5 Deep Learning in Finance (Heaton et al., 2018)

2.5.1 Business Problem

Financial forecasting issues are both practical and theoretically fascinating. They're also extremely intimidating. According to theory, most information important to financial

prediction issues is dispersed throughout accessible economic and other data, a concept backed up by the numerous divergent data sources that market players monitor for hints on future price movements.

It's challenging to deal with so many different data sources. Financial economic theory does not adequately specify the relevance of the data or the potentially complicated non-linear interactions in the data, thus there is a huge collection of potentially relevant data. In reality, this leads to a slew of predictive models, many of which lack theoretical support and are prone to overfitting and poor predicted out-of-sample performance.

What is required is a technique capable of learning the complicated characteristics of data inputs that result in accurate predictions of the desired output variables (such as an asset or portfolio return).

The study introduces deep learning hierarchical decision models for financial prediction and categorization in this work. The deep learning predictor provides a variety of benefits over standard predictors, including the ability to learn from experience.

2.5.2 Research Methodology

The study applies a hybrid AE + LSTM approach with Smart Indexing. The concept of applying Smart Indexing is as follows:

There are two basic methods we may use when attempting to duplicate (or approximate) a stock index using a subset of equities.

3. Identify a small set of equities that have historically performed similarly to the index under consideration.
4. Identify a small set of stocks which historically have represented an over-proportionally significant share of the total aggregate information of all the stocks the index consisted of.

While 1 and 2 may appear to be fairly similar on the surface, they represent completely distinct methods.

Many traditional techniques to index replication are based on linear regression, which belongs to group 1. We frequently try to discover a small group of stocks that in-sample offers a decent linear approximation of the considered index via trial and error.

The deep learning version of 1 allows for the translation of input data into a desired output via a hierarchical series of adaptive linear layers, which implies that even non-linear correlations may be easily detected during training. We call the resultant approximation (or prediction) method a deep feature policy since each buried layer gives a different interpretation of the input features.

The availability of customized non-linear connections in deep learning reduces the traditional goal of 1, namely good in-sample approximation, to a triviality, and shifts the focus to out-of-sample performance training.

Another flaw with 1 is that it corresponds to the final (and hence diluted) product. A deep auto-encoder solves this problem by directly (rather than indirectly) estimating the aggregate information contained in the index stock family in question.
The construction of an auto-bottleneck encoder produces a compressed data set from which all stocks are recreated. As a consequence, when it comes to indexing, the stocks that are closest to the compressed core of the index may be thought of as a non-linear foundation of the aggregate information of the considered family of firms.

During the 2014/15 financial year, all S&P500 equities were auto-encoded. We then rated the stocks based on their closeness to their own auto-encoded version; the closer a stock was to its own auto-encoded version, the greater its shared information richness.

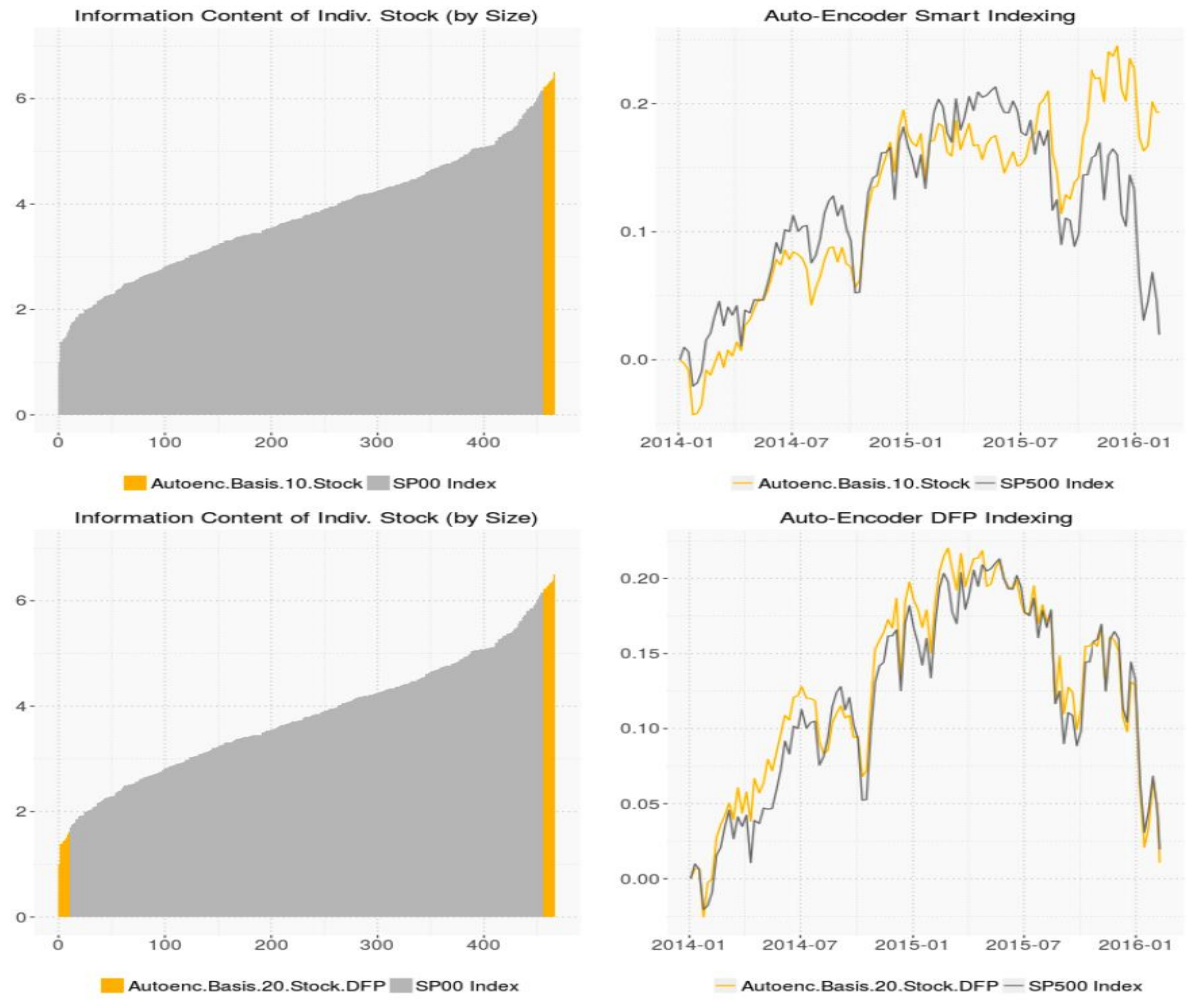


Fig 5: From the auto-encoder basis of the 10 top stocks, develop an equally weighted portfolio based on their proximity to the auto-encoded information.

We auto-encoded all equities in the S&P500 throughout the 2014/15 timeframe in Figure 5. We then rated the stocks based on how near they were to their own auto-encoded version; the closer a stock was to its own auto-encoded version, the greater its shared information richness. The S& P500 may be approximated by just investing in the 10 stocks having the largest community information content, as shown in the upper right.

While the ten stock auto-encoder foundation is fair, we note that the approximation is somewhat wrong, especially in the latter seven months of the training period. It's enlightening to see how utilizing a DFP index approximation, this divergence may be easily prevented in-sample.

We merged the two groups of 10 stocks with the greatest and lowest communal information, respectively, in Figure 5 at the bottom, and then trained a deep learning procedure to

approximate the S&P500 index based on this extended foundation of twenty stocks during the same period of 2014/15. The DFP generates an optimal action for approximating the target index for each combination of inputs from the selected twenty stocks, based on the hierarchical composition of non-linear characteristics derived from the input data. Given enough variation in the input data, a DFP may frequently be taught to estimate the target data to almost arbitrarily high accuracy, as seen in the bottom right chart over the last six months of the training period.

2.6 A deep learning framework for financial time series using stacked autoencoders and long-short term memory(Bao et al., 2017)

2.6.1 Business Problem

This paper offers a unique deep learning framework for stock price forecasting that combines stacked autoencoders (SAEs), wavelet transformations (WT), and long-short term memory (LSTM). This study aims at applying a deep nonlinear topology for time series prediction. By extracting strong features that capture the key information from the complex real-world data, tthe new model can successfully achieve even better performance than before, which is an improvement above traditional machine learning approaches.

2.6.2 Research Methodology

Few attempts have been made to examine if the stacked autoencoders approach might be used to financial market prediction, and this work contributes to that field. It proposes a unique stock market prediction model based on the stacked autoencoders method.

WT, SAEs, and LSTM are the three components of the suggested model in this study. The SAEs are a key component of the model, since they are utilized to learn the characteristics of financial time series in an unsupervised manner. It's a single-layer autoencoder neural network with each layer's output feature linked to the inputs of the following layer. SAEs are trained unsupervised one at a time, with output and input data compared to reduce error. As a result, the SAEs model is capable of learning both invariant and abstract features.

To assist enhance prediction accuracy, the other two approaches are used. Because it is ideal to learning from past experience and forecast time series with variable size windows, the LSTM, a kind of RNN, was used. Financial time series are believed to benefit from WT since it reduces noise. For single-dimensional data, it is a common filtering and mining approach.

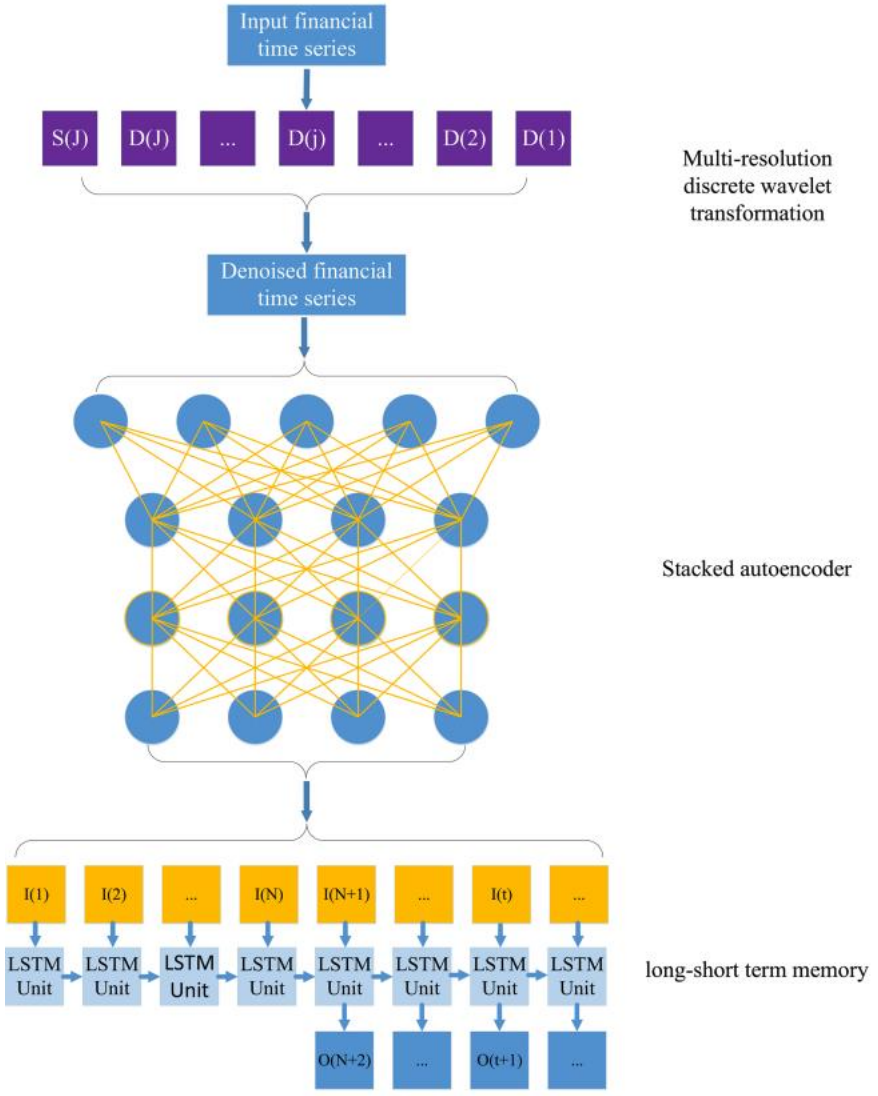


Fig 6: The suggested deep learning architecture for financial time series is depicted as a flowchart. The detailed signal at the j-level is D(j). At level J, S(J) is the coarsest signal. At time step t, I(t) and O(t) represent the denoised feature and one-step-ahead output, respectively. The number of LSTM delays is N.

The framework consists of three stages:

- eliminate the noise by using the wavelet transform to deconstruct the time series
- use SAEs with an unsupervised deep architecture
- generation of one-step-ahead output using long-short term memory with delays.

2.6.2.1 Data Description

CSI 300, Nifty 50, Hang Seng index, Nikkei 225, S&P500, and DJIA index are the six stock indexes used in this study. As the neural network's validity may be influenced by market conditions. Samples from various market circumstances may be useful in resolving this issue. The NYSE, often regarded as the world's most rich and developed financial exchange, trades the S&P500 and DJIA indices. Mainland China and India's financial markets, on the other hand, are usually classified as new markets. In actuality, the majority of the country's market institutions are still in the planning stages. The Nifty 50 and CSI 300 were chosen to represent emerging markets as a consequence.

As inputs, three sets of variables have been chosen. The first set of variables is each index's historical trade data. The data contains the OHLC variables as well as the trade volume, as per prior literature. Each index's fundamental trade information is represented by these variables. Another set of inputs for each index is a collection of 12 frequently utilized technical indicators.

2.6.3 Results

According to the model findings during a 6-year time span, RNN and LSTM had larger distances and variations to the real data compared to WLSTM and WSAEs- LSTM, according to the yearly expected data and the corresponding actual data in the graph. In addition, when comparing WLSTM to WSAEs-LSTM, the latter beats the former: WSAEs-LSTM is less volatile and is similar to real-time trade data compared to WLSTM. Prediction Performance of WSAEs-LSTM appears to be more apparent in developing markets compared to established ones.

Not just on average, but year after year, WSAEs-LSTM beats the other three. To show the robustness of our findings, the study looks at the statistical relevance of discrepancies between the other three models and WSAEs-LSTM. For each accuracy parameter, we compare the 24 quarterly WSAEs-LSTM results to the three models using T-test statistic.

According to the data, the difference in predictability between two particular models in forecasting two stock indices is modest if the two stock indices are traded in markets with

comparable development states, but it rises if the two stock indices are exchanged in markets with distinct development states.

2.7 A Recurrent Neural Network Approach in Predicting Daily Stock Prices(Samarawickrama & Fernando, 2017)

2.7.1 Business Problem

The purpose of this study was to create models based on the Recurrent Neural Network (RNN) Approach to forecast daily stock values of chosen listed firms on the Colombo Stock Exchange (CSE), as well as to assess the accuracy of the models generated and, if necessary, to detect model flaws. Models were built using FNN, SRNN, GRU, and LSTM architectures. For each firm, the OHLC prices from the previous two days were chosen as input variables. The biggest and lowest predicting errors are produced by feedforward networks. The finest feedforward networks have a predicting accuracy of around 99 percent. When compared to feedforward networks, SRNN and LSTM networks yield lesser mistakes on average, but on rare instances, the error is greater. GRU networks have a stronger predicting capability than the other two networks.

2.7.2 Research Methodology

Three firms were chosen for this study among the Colombo Stock Exchange's (CSE) 297 listed companies. These businesses were chosen from among 20 CSE sectors from three industries with the greatest trading volume per year. The companies were chosen based on the stock price's stability and market performance (trading volume and frequency). The following companies were chosen for this study: COM(Finance), RCL(Manufacturing) and JKH(Diversified Holding Sector). The model was built using data from these firms from January 1, 2002, to June 30, 2013.

The Low and High prices of the previous 2 days have been chosen as inputs for model building.

As a result, input params (t – current time) consist of:
C(t-1) – day t-1 closing price

C(t-2) – day t-2 closing price

L(t-1) – day t-1 lowest price

L(t-2) – day t-2 lowest price

H(t-1) – day t-1 highest price

H(t-2) – day t-2 highest price

Three prominent neural network topologies were used for model development in this study are GRU, LSTM and SRNN. The Feed Forward Neural Network (MLP) has also been utilized for comparison. Ten NN models were created for each of the neural network designs which were created with different latent/hidden units between 2 and 11. As a result, each firm had 40 NN models.

Each NN model contains a 6 input neuron input layer, a 2 to 11 hidden neuron hidden layer, and a 1 output neuron output layer. The internal architecture of each of ten models built using a single design for a single firm is described in the table below.

The data was scaled using min-max scaling in this investigation. The train, validation and test data ratio was taken as 70:15:15.

On the Windows platform, the Keras was used to implementation and training of the NNs. To train neural networks, the following activation functions and features were utilized.

Parameter	MLP	SRNN	LSTM	GRU
Initialization function	uniform	glorot uniform	glorot uniform	glorot uniform
Inner Initialization function	-	orthogonal	orthogonal	orthogonal
Activation function	relu	softsign	softsign	softsign
Inner activation function	-	hard sigmoid	hard sigmoid	hard sigmoid
Error (loss) function	MSE	MSE	MSE	MSE
Optimizer/Training algorithm	adam	rmsprop	rmsprop	rmsprop

Table 2: Model parameters used for training

All the 40 NN models underwent 5000 rounds of training. So, for all of the networks, the halting condition was 5000 iterations.

The test dataset was used to make predictions after training. MAPE and MAD were calculated to quantify the accuracy of each neural network model in order to assess its performance. The model with the lowest MAPE/MAD was deemed the best.

$$MAD = \frac{1}{|ValidationSet|} \sum_{for.all.days \in .ValidationSet} \left| price_{forecast}^{tomorrow} - price_{real}^{tomorrow} \right|$$

$$MAPE = \frac{1}{|ValidationSet|} \sum_{for.all.days \in .ValidationSet} \left| \frac{price_{forecast}^{tomorrow} - price_{real}^{tomorrow}}{price_{real}^{tomorrow}} \right|$$

To compare projected prices to actual prices, the results from the tests were converted back to raw format.

2.7.3 Results

When it comes to test error (or forecast error), we observed that MLP models yield the lowest and highest mistakes in this study. The finest feedforward networks have a predicting accuracy of around 99 percent. When compared to feedforward networks, SRNN and LSTM networks yield lesser mistakes on average, but on rare instances, the error is greater. GRU networks produce far greater predicting errors than the other two networks.

In most research, RNN models (particularly LSTM) gave the best results when incorporating the findings of prior investigations. MLP models, on the other hand, yield the greatest outcomes in this investigation. This is due to the fact that only data from the previous two days were used as inputs in this study. RNN models would give the best results if the number of previous days examined for input variable selection was increased.

2.8 CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets(Liu et al., 2017a)

2.8.1 Business Problem

The CNN and CNN-LSTM neural networks are used to model and analyze quantitative selection and quantitative timing strategy in stock markets in this article. The CNN is used methodically to create a quantitative stock selection approach, and then the LSTM is used to create a quantitative timing strategy for increasing profits. Experiments have shown that the CNN-LSTM neural network model may be utilized to build quantitative strategies that outperform the Benchmark index.

2.8.2 Research Methodology

The CNN-LSTM framework's specifics are as follows:

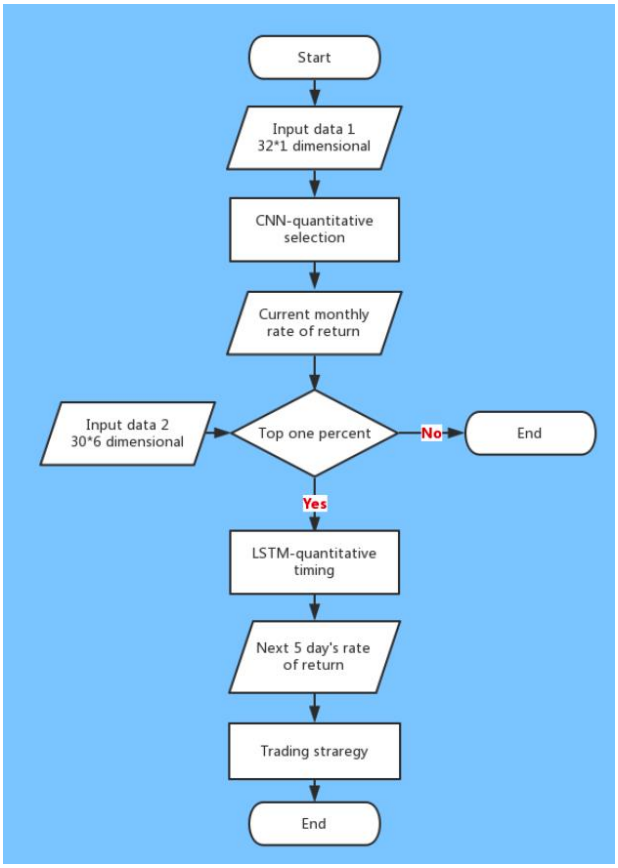


Fig 7: Methodology Flowchart for CNN-LSTM

Parameters	CNN	LSTM
Input Layer	1	1
Conv/LSTM hidden Layer	2	1
FCN hidden Layer	2	1
Output Layer	1	1
Epoch	500	100
Activation	ReLU,Tanh	Tanh
Weight	Normal(0,1)	Normal(0,1)
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Objective function	cross-entropy	cross-entropy

Table 3: CNN-LSTM Parameter Details

When necessary, data was subjected to z-score normalization. The training data spans the years 2007-1-1 to 2013-12-31, whereas the testing data spans the years 2014-1-1 to 2017-3-31. Only the characteristics that will be input into the neural network are chosen, and the neural network is trained with random weights and biases. The LSTM component of the CNN-LSTM model is made up of consecutive layers, 1 LSTM layer, followed by a dense layer. Activation function used for dense layer is Tanh

The dropout parameter is introduced to the CNN-LSTM model, and to avoid overfitting the model, the regularization term is added to the weight.

We purchase and hold for 5 days when the LSTM projected value equals 1, then update the number of held days to 5 and hold for another 5 days. If the LSTM predictive value is equal to -1, it will continue to maintain short positions; previously held shares will have their number of held days decreased by one; and if the number of held days is equal to 0, the share will be sold.

2.8.3 Results

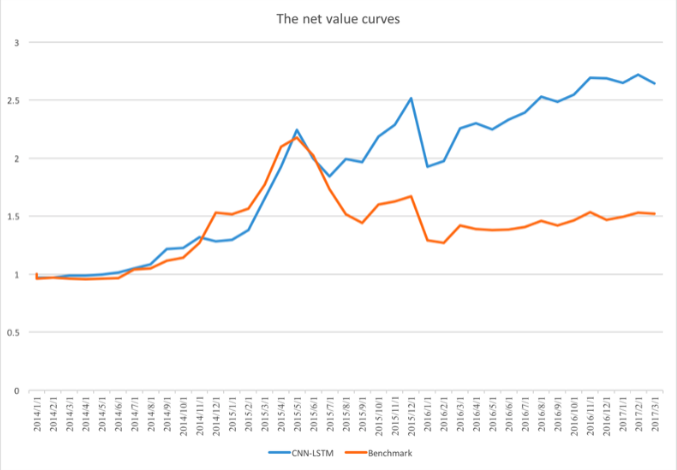


Fig 8: CNN-LSTM vs Benchmark net value curves

	CNN-LSTM	Benchmark
Annualized rate of return	0.339	0.135
Maximum retracement	0.235	0.417

Table 4: CNN-LSTM vs Benchmark

Our CNN-LSTM model's net value, rate of return, and maximum retracement are compared to the Benchmark index.

The net value curves show a considerable qualitative improvement in the performances. The greatest retracement of CNN-LSTM is reasonable during a stock market crisis. Our CNN-LSTM neural network model's annual return rate is 2.5 times that of the Benchmark index's annualized rate of return. Meanwhile, the CNN-LSTM model's maximum retracement is 56% of the Benchmark index's maximum retracement. The trials demonstrate that CNN-LSTM model is economical and that the returns are outstanding, as well as the algorithm's resilience and practicability.

2.9 Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation(Yuan et al., 2018a)

2.9.1 Business Problem

DWNNs are presented in this paper, which can analyze temporal input in depth (time step) while extracting the correlation feature in the breadth direction. For our research, we use stock market data, which has an interesting correlation pattern. According to empirical data, DWNNs perform 30 percent better than regular RNNs in terms of performance.

The concept of integrating DNNs, LSTMs, CNNs and other neural models has been discussed in the past. However, the models in this research were trained individually, and the previous outputs were combined using a combination layer. The model is being developed further based on its foundation, which combines LSTMs, DNNs and CNNs into a single model that can be trained parallelly.

The merger of several neural models is based on feeding better features to LSTMs as input, which can optimize the performance of RNNs while lowering the spectrum variance of CNNs. As a consequence, the author introduces CLDNN, a system that imports input data into CNN layers for enhanced feature extraction, then implements LSTMs depending on the feature, and lastly outputs output using fully connected DNNs. The CNN-BLSTM presented in the article combines RNNs and CNNs to increase performance. According to these research, combining RNNs, CNNs and other NN models is a viable option.

2.9.2 Research Methodology

For many sets of correlated temporal data, each temporal data can be extended in time step indefinitely. If we use a generic RNN model to handle the data, we'll require m RNN models, each of which is trained separately when dealing with such difficulties. As illustrated in Figure 9, the **m** RNNs have been expanded across time steps.

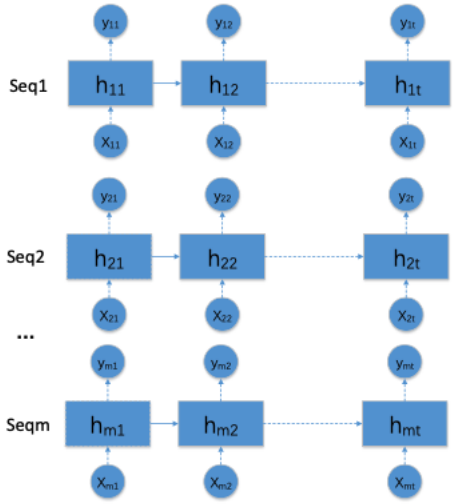


Fig 9: M generic RNN models were extended across timestep.

We suggest a DWNN to improve this problem because this type of processing method obviously ignores vital information regarding the correlation between temporal data. Figure 10 depicts the DWNN structure of consecutive time steps. Between the RNN model's time steps **t** and **t-1**, we add CNN layers. The CNN layers are given the latent states of the **m** RNN models at time **t-1**. At time t, the CNN layer outputs are used as the state input for the RNN models. If we consider the hidden state to be the RNN's memory or history, the state processed by the CNN layers is the memory of the m RNN model. Prior to time t, this new state includes all of the memory of the m RNN.

Consider the generic RNN model, which operates with a variable-length sequence $X = (x_1,..., x_T)$ and contains a hidden state h and an output y.

The RNN model's status update formula is

$h_t = f(h_{t-1}, x_t)$

at each time step t, where f is a non-linear activation function.

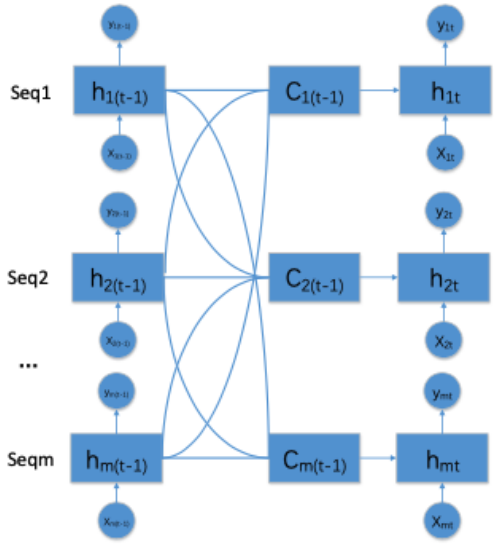


Fig 10: Structure of neighbouring time steps in a DWNN

Now we consider the DWNN structure. Here, m RNNs are required to train m input sequences {Xi, 1 <= i <= m}, where Xi = (x_{i1},...,x_{iT}). Here, input m refers to sequences of different lengths, {Xi, 1 <= i <= m}. The DWNN model's status update formula at time step t is

$h\_temp_{it} = f_i(h_{i(t-1)}, x_{it})$ for 1<= i <= m

$h\_c_{it} = concat(h_{tempit}, 1 <= i <= m)$

$h_{it} = C_i(h\_c_{it})$ for1 <= I <= m

where f_i is the i_{th} RNN model's nonlinear activation function of the RNN cell;
concat is used to concatenate the hidden state of the m RNN Cell to generate a big vector
C_i refers to the CNN layer operation in the i_{th} RNN

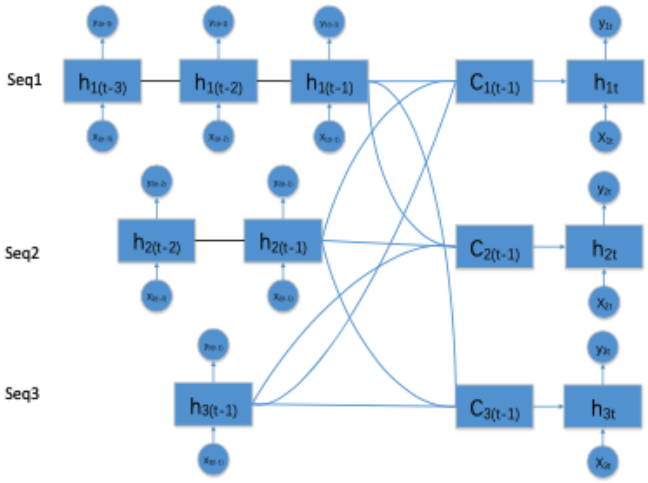


Fig 11: DWNN across different period

To build single RNN chains, we use the Seq2seq model(Cho et al., 2014). The Encoder and Decoder halves of the Seq2seq model are built using two RNNs each. The Encoder part translates the input into a vector of constant size, which is subsequently mapped to a target sequence of variable length by the Decoder component.

We employ the Seq2seq model's m groups to handle m groups of temporal data sequences at once. CNN layers are added to the encoder, while the decoder remains separate and unaffected. As per Seq2seq, to implement CNNs in DWNNs, these are the following ways:

1) The Seq2seq model in (Cho et al., 2014) employs a single layer RNN with GRU. This model is straightforward to construct since GRU only has one hidden state and just one set of CNN layers.

2) The RNN cell in the Seq2seq model in (Sutskever et al., 2014) is an LSTM. Because LSTMs have two latent states, c and h, DWNN requires the deployment of two groups of CNN layers.

3) (Sutskever et al., 2014) claims that Seq2seq model's performance can be enhanced by using reverse input data. As a result, building a DWNN with a bidirectional RNN (Graves & Schmidhuber, 2005) requires twice as many CNN layers.

4) According to (Cho et al., 2014), the Seq2seq model employs a four-layer RNN. The DWNN construction is adaptable for multi-layer RNN models. CNN layers can be built for each layer, or some.

In each RNN loop step, a CNN layer is added, but Such models are difficult to train and run the danger of overfitting. As a result, we follow the design of an interval CNN, which involves adding a CNN layer after every time window k which decreases the model's complexity while also allowing it to extract relationships. Previous RNNs struggled to deal with temporal data of varying lengths of time. However, we may create a CNN layer at the common multiple of time window, using DWNN utilizing interval CNN layer.

TensorFlow has been used to implement the DWNN network proposed in this paper. To train the algorithm, stock data from China's SSE has been used. The whole dataset contains 12 stocks which are 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679, 002259, 002679 The dataset comprises daily k-line data from January 1, 2000, to August 16, 2017, and includes the OCHLV data. The training samples date from 2000/01/01 to 2015/12/30, whereas the testing samples date from 2016/01/01 to 2017/08/16.

The DWNN is made up of of 12 seq2seq models utilizing GRU; Encoder is set at time window of 60 days, and the length of the Decoder is set as 7, implying that 60-day historical data will be utilized to forecast data for the next week(days). To convert the dimension of the data into the form of the GRU hidden state, we use the whole connection layer. Each CNN layer has two convolution layers as well as a pooling layer.

In the experiment, instead of utilizing the order of data produced input samples, we use the random sampling technique with the Batch size as input. The samples were produced in such a way that the loss varied substantially early in the training. It does, however, have the benefit of regularization and, to a degree, improving the training performance.

2.9.3 Results

The interval CNN layer is compared to the DWNN model using the seq2seq model. Figure 11 shows the comparison experiment results, while The DWNN model results are shown in Figure 12, with the blue dots indicating the actual rate of growth in stock prices over the following 7 days and the orange plus sign showing the projected value.

When comparing the two graphs, we can observe that the DWNN model outperforms the general RNN model significantly.

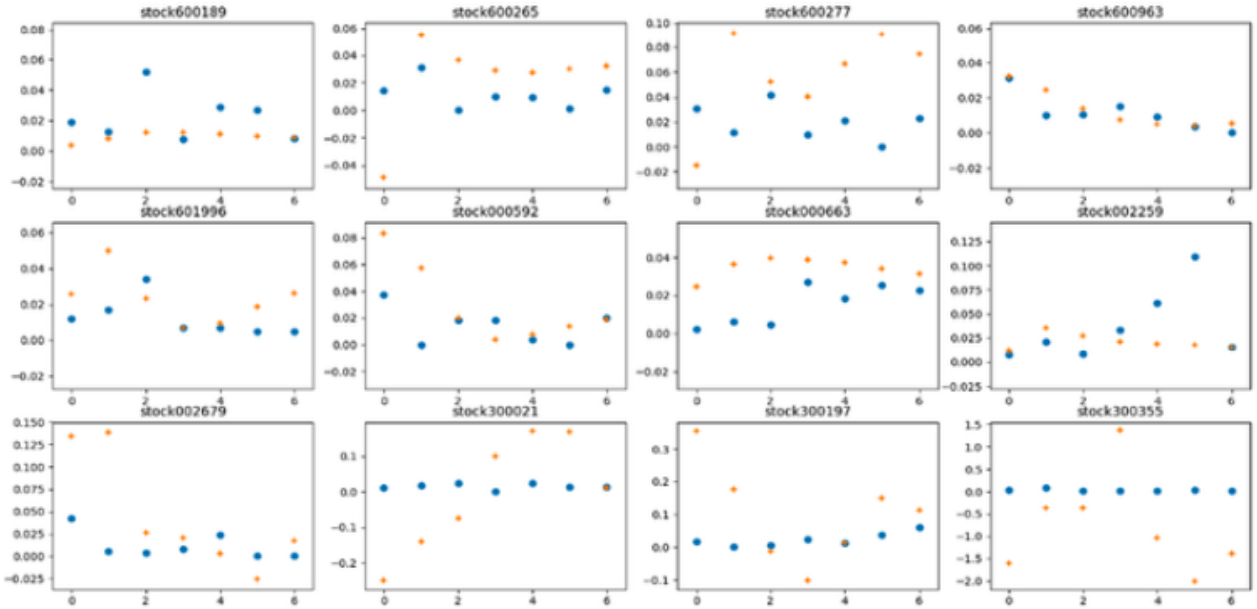


Fig 12: comparison results

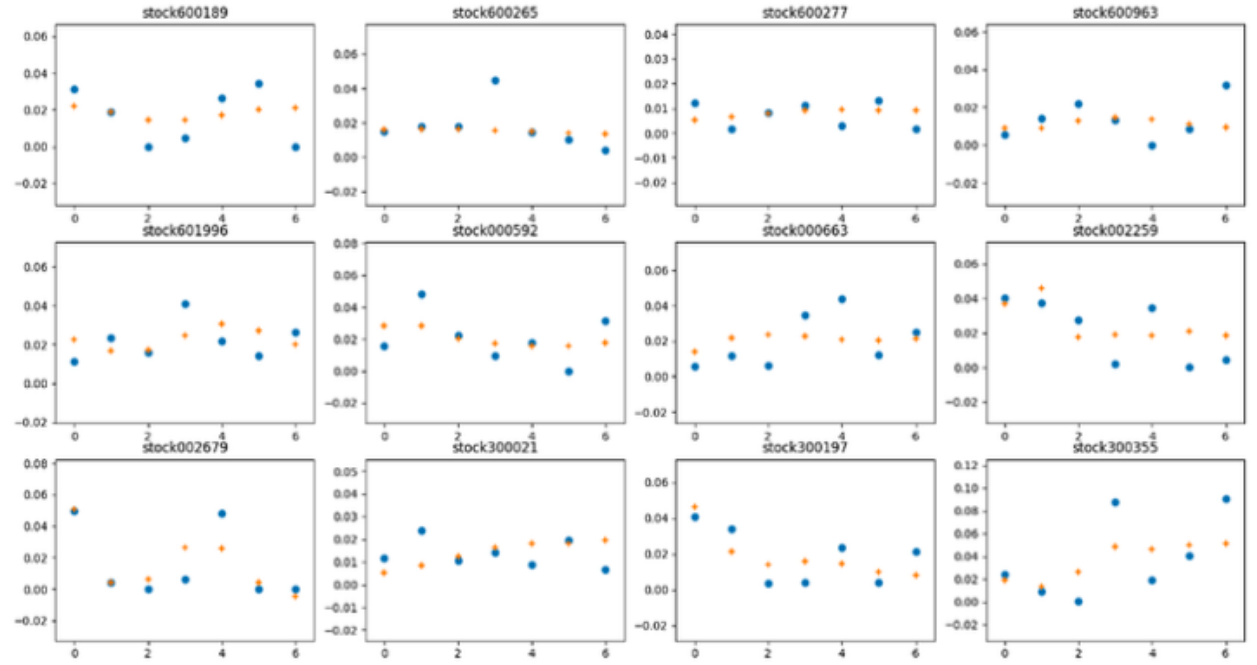


Fig : 13 DWNN results

2.10 Stock Price Prediction via Discovering Multi-Frequency Trading Patterns (Zhang et al., 2017)

2.10.1 Business Problem

Long and Short term trading activity represents diverse trading patterns and frequency affect stock values. These patterns, on the other hand, are usually elusive because they are impacted in the real world by a range of unexpected political and economic factors, like corporate performance, breaking news that travels across markets and government restrictions. Furthermore, stock price time series are non-linear and non-stationary, making forecasting future price movements difficult.

To address these difficulties, this study proposes a State Frequency Memory (SFM) recurrent neural network based on DFT. The memory cell hidden states are broken down by the SFM divides into multiple frequency components, each representing a distinct frequency of hidden trading patterns that drive stock price volatility. A nonlinear combination of these components is used to forecast future stock values using an IFT.

While short-term projections are generally based on trading patterns where high-frequency is observed. On the other hand, long-term projections are be based on low-frequency trading patterns.

Although, there is no method in the literature that clearly distinguishes between different frequencies of trading patterns in order to generate dynamic forecasts. Explicitly identifying and differentiating different frequencies of hidden trade patterns play a key role in developing price forecast for diverse time steps.

While there exist numerous techniques for forecasting stock prices, none of the available models explicitly deconstruct trading strategy patterns into multiple frequency components and incorporate the identified multi-frequency patterns into price forecasts, to our knowledge.

The frequency domain of input signals is found via the Discrete Fourier Transform in signal processing (DFT). It breaks down input signals into various frequency components in order to investigate their periodicity and volatility. This idea will be used to dynamically deconstruct

latent trading patterns states into multiple frequency components, which will fuel SFM's growth over time. The SFM has been impacted by the LSTM's gating design and multi-frequency decomposition.

2.10.2 Research Methodology

2.10.2.1 Dataset

For the experiments, day-to-day open prices of 50 companies across 10 different sectors from year 2007 to 2016 have been collected from Yahoo Finance. Corporations with the top five market capitalizations in each industry are chosen. Before 2007, all of the companies were publicly traded. For certain stocks, there have been multiple share splits in the past. In this case, the prices have been adjusted based on the most recent split. The historical prices have a 2518-day period. Daily prices from 2007 to 2014 were used to train the models for a total of 14 days. Validation and testing are done using daily pricing from 2015 and 2016. Both the validation and test data is composed of 252 seconds. We anticipate the models to learn broad market trends over time based on the prices of 50 businesses during the last 10 years.

2.10.2.2 Price Prediction

Consider a stock's trading prices over time, $\{p_t \mid t = 1, 2,..., T\}$. The objective is to create a n-step forecast on $\mathbf{p_{t+n}}$ according to prices till $\mathbf{p_t}$, given n >= 1.

The forecast on n-step price $\mathbf{p_{t+n}}$ at the timestep t + n may be viewed as a equation given prices $\{p_t \mid t = 1, 2,..., T\}$.

$$\hat{p}_{t+n} = f(p_t, p_{t-1}, ..., p_1)$$

where f indicates the model mapping from historical prices to n-step forward prices.

Because the price scales change for various stocks, we normalize the prices $\{p_t \mid t = 1, 2,..., T\}$ per stock into $\{v_t \mid t = 1, 2,..., T\}$ within the [-1, 1] range without losing generality.

A RNN such as the LSTM or SFM is used, as a mapping for n-step prediction. At each step, the normalized price $\mathbf{v_t}$ is used as an input, with the initial latent/hidden state $\mathbf{h_0}$ and any memory state set to zero. The selected RNN version generates a hidden vector $\mathbf{h_t}$, which is utilized to forecast prices. We apply a matrix transformation to the hidden vector on the n-step prediction:

$$\hat{v}_{t+n} = \boldsymbol{w_p h_t} + b_p$$

where $\mathbf{w_p}$ stands for weight vector and $\mathbf{b_p}$ stands for bias The nonlinearity in this approach is due to the $\mathbf{h_t}$ which is the nonlinear hidden vector, despite the fact that this is a linear transformation.

This architecture varies from that of the AR model that predicts $\mathbf{p_{t+n}}$ using immediately previous data utilizing { $\mathbf{p_{t-r}}$ | 0 <= **r** <= **w** } where **w** is the sliding window to forecast $\mathbf{p_{t+n}}$.

While increasing the sliding window can assist to integrate the long-term context of equity prices, it also adds to the model's complexity, raising the danger of overfitting onto previous values.

This issue does not occur in the SFM or LSTM because inherently, the memory state is capable of maintaining long-term dependencies across input prices.

Prices from numerous stocks are utilized to train the regression network in order to understand broad trading trends in the stock market. The model is then trained with M stocks while minimizing the sum of square errors in the training set between true normalized prices and predicted prices:

$$\mathcal{L} = \sum_{m=1}^{M} \sum_{t=1}^{T} (v_{t+n}^m - \hat{v}_{t+n}^m)^2$$

The BPTT method is used to update all of the model parameters. In this approach, the model's weights are distributed evenly among all stocks.

Any RNN variation can be used to replace the RNN layer in the regression network. For comparison, we use the LSTM and SFM instead of the RNN. Intrinsic memory states are preserved in both designs to explain the core components of trading patterns till time t. With

the aid of memory gates, these internal variables are meant to represent the dependency between stock prices.

Furthermore, the SFM's memory states split states into numerous frequency components. Within the memory cell, multi-frequency patterns are summed. It generates the hidden state $\mathbf{h}_t$ by combining these frequency memory components, which may be utilized to forecast future prices.

2.10.3 Results

The proposed SFM network has been compared with the existing state-of-the-art methods, such as the Autoregressive(AR) model and traditional LSTM, in experiments.

The AR model is a time-series forecasting approach that has been around for a long time. It forecasts future market prices based on the prices of previous stages. Its forecast is formed as a linear combination of previous prices exposed to a Gaussian noise component. The number of prior prices used in the prediction is specified by the AR model's parameter w, which reflects the degree of reliance on historical data. By minimizing the AIC or BIC, the optimal w may be obtained. An AR model, in contrast to the suggested approach, is a linear regression model that can only represent stationary time-series. We hope to see if SFM can represent non-stationary and non-linear dependence, along with multi-frequency patterns in the stock market, by comparing it to the AR model.

We also provide a comparison to the traditional LSTM. It simply records the past prices' internal dependencies. The efficacy of the SFM can be assessed in investigating trade patterns with varied frequencies to forecast stock prices by comparing it to the traditional LSTM. The same technique is used to teach both LSTM and SFM. The models are trained using the RMSprop optimizer, with a fixed learning rate of 0.01. Orthogonal initializer and Xavier uniform initializer respectively initialize the weights U and W. The bias vectors b are set to zero by default. The weights and bias are updated every iteration using all of the training sequences. After 4K iterations, both models converge.

To evaluate the performances, we used the daily mean square error per stock as our assessment criterion. The 1-step forecast, which is a short-term prediction, suggests the trend

of the next day. The half-week trend is indicated by the three-step forecast. Because the stock market is only open on weekdays, a 5-step prediction suggests the next week's trend. To put it another way, a 5-step forecast usually spans one weekend, which is a considerably more difficult assignment.

	1-step	3-step	5-step
AR	6.01	18.58	30.74
LSTM	5.93	18.38	30.02
SFM	**5.57**	**17.00**	**28.90**

Table 5 : mean square error of models across multiple steps

Both the LSTM and AR models are outperformed by the SFM, whereas the LSTM model beats the AR model. Each model's performance degrades as we increase the prediction step, as we expected. The SFM, on the other hand, degrades more gradually compared to the other two types. This is due to its capacity to simulate multi-frequency trading patterns in a price time-series environment that has been modified.

The number w of previous prices utilized in the AR model may be thought of as the size of the price data memory. Experiments using the AR model of various types are also carried out. No significant variations in test error is observed when the number of w increases. In fact, as **w** grows greater, the test error gets worse. This might be due to the addition of more parameters to the model, which leads to overfitting of the price data training sequence. The LSTM and proposed SFM, on the other hand, have a smaller test error than the AR model, indicating that they are better at capturing long-term relationships while avoiding the AR model's overfitting issue.

The SFM provides a more exact prediction when trade patterns with various frequencies are involved, as opposed to the LSTM, which solely models internal dependencies. It's worth noting that the frequency components account for trade operations that occur at various rates and cycles. The SFM filters the unnecessary frequency components and maintains the important ones while regulating information from multiple frequencies with gate units to offer direction for future trend prediction. By restricting the model to the local pattern of price fluctuations, it avoids states from dominating the price projection.

The three models can accurately predict stock price trends. The SFM's error curves are smaller than those of the LSTM and the AR model, as predicted. Furthermore, the AR has a tendency to merely replicate the price trend of the past with a significant delay. When the stock price moves abruptly, this results in an extremely low forecast accuracy. The LSTM and SFM, on the other hand, outperform the AR model in the short term because they are less affected by local price changes.

The planned SFM network's complexity has a direct influence on its performance. With a sufficiently sophisticated model, the training error can theoretically be arbitrarily low. A low test error does not always indicate a low prediction error on the training set.

Overfitting is more likely with a complicated model. Two metrics that may be used to measure the SFM's complexity are the number of states and the number of frequencies. It's worth mentioning that the appropriate hyper-parameters for model complexity are chosen depending on the performance of validation set. In our tests, we learn the model with prices from 2007 to 2014 and then validate it with prices from 2015.

# of states	10	20	40	50
1-step	**5.93**	6.60	6.91	6.89
3-step	**18.38**	19.05	18.60	18.68
5-step	30.30	30.58	**30.02**	31.07

Table 6 : In LSTM, the mean square error vs. # of states

The LSTM's complexity is a function of the number/dimension of states. In contrast, the number of states in the SFM refers to the number of rows in the memory state matrices, suggesting the number of patterns predicted to emerge from the price time-series. As the network learns more predicted patterns, the model grows more complex. With 10-dimensional memory states, the LSTM obtains the highest performance for 1-step and 3-step prediction. With the growing number of states, we see overfitting. Increasing the number of states will no longer reduce the test error. Worse, in LSTM, a large number of states (more than 50) might exacerbate the test error, resulting in an excessive fit into the training cost. The LSTM mixes the information of all frequencies, including the irrelevant frequency components, without deconstruction and regulation of information over multiple frequencies. As a result, by trapping the model in the local pattern, it tends to dominate price prediction.

Because the 5-step prediction is more difficult, the LSTM requires more states (40) to obtain the best results.

# of states	10	20	40	50
1-step	**5.57**	5.79	6.15	5.91
3-step	18.48	19.20	17.25	**17.00**
5-step	29.48	29.84	31.30	**28.90**

Table 7: In SFM, the mean square error vs. # of states with frequencies hardcoded to ten.

The overfitting problem is mitigated by incorporating state decomposition and control, as opposed to the LSTM. The SFM, in particular, performs best with 50 states for the 3-step and 5-step. Although the effectiveness of 1-step prediction degrades as the number of states increases, the SFM degenerates more slowly than the LSTM.

# of frequencies	5	10	15	20
1-step	6.69	5.91	5.91	**5.88**
3-step	18.39	**17.00**	19.15	19.52
5-step	30.95	**28.9**	30.57	31.22

Table 8: In SFM, the mean square error is shown against the # of frequencies with states set at fifty.

The number of frequencies, in addition to the states, is an essential hyperparameter in the SFM. On [0, 2π], the frequencies are equally distributed. It is obvious that as the frequency of the memory states increases, so does the amount of information on the high frequency in short-term memory. In addition, the components of high frequency trading are used to account for short-term trading activity. It demonstrates the SFM's ability to mimic high-frequency trading in the stock market. For 3-step and 5-step prediction, the best performance is obtained with ten frequencies. Because their volatility is smaller than the 1-step prediction, components with low frequency become more important to the forecast.

2.11 Enhancing Stock Movement Prediction with Adversarial Training(Feng et al., 2019)

2.11.1 Business Problem

This study presents a new ML approach for stock price forecasting, with the goal of predicting whether a stock's price will rise or fall in the near future. Adversarial training can be used to increase the generalization of a NN based time series prediction model, which is a unique approach. The logic of adversarial training in this case stems from the fact that stock prediction input characteristics are generally dependent on raw stock price data, which basically is a stochastic variable that changes over time. As a result, traditional price-based training (e.g., the close price) can easily overfit the data, making it impossible to generate trustworthy models. To solve this challenge, we suggest using perturbations to replicate the stochasticity of pricing variables and training the model to perform well under tiny but deliberate perturbations. Extensive trials on two real-world stock data demonstrate that our technique beats the state-of-the-art solution(Xu & Cohen, 2018) by 3.11% on average in terms of accuracy, demonstrating the utility of adversarial training for stock prediction tasks.

2.11.2 Research Methodology

Datasets. The proposed approach is evaluated against two stock movement prediction benchmarks: ACL18(Xu & Cohen, 2018) and KDD17(Zhang et al., 2017)

ACL18 provides historical data for 88 high-trade-volume equities in the NASDAQ and NYSE marketplaces from January 1, 2014 to January 1, 2016. We first align the trade days in the past, eliminating weekends and public holidays that lack historical pricing, as recommended by(Xu & Cohen, 2018). We next create candidate instances each with a lag with a duration of T along the aligned trade days (i.e., one example for a stock on every trading day). The potential instances are labelled based on the percent change in stock close prices. Positive and negative instances with movement of 0.55 percent and 0.5 percent, respectively, are discovered.

KDD17 contains a lengthier history of 50 equities in US markets from January 1, 2007 to January 1, 2016. We use the same method as ACL18 to distinguish positive and negative examples because the information was initially gathered to forecast stock prices rather than movements. The instances were then divided into three categories: training (January 1, 2007 to January 1, 2015), validation (January 1, 2015 to January 1, 2016), and testing (January 1, 2016 to January 1, 2017).

Features	Calculation
c_open, c_high, c_low	*e.g.,* $c\_open = open_t/close_t - 1$
n_close, n_adj_close	*e.g.,* $n\_close = (close_t/close_{t-1} - 1$
5-day, 10-day, 15-day, 20-day, 25-day, 30-day	*e.g., 5-day* $= \frac{\sum_{i=0}^{4} adj\_close_{t-i}/5}{adj\_close_t} - 1$

Table 9: temporal features to explain the trend of a stock

We create 11 temporal characteristics ($\mathbf{x}^s_t$) to explain the trend of a stock s at trading day t instead of utilizing raw EOD data. The features related with computation are detailed in Table 9. The purpose of establishing these characteristics is to:

1) equalize the prices of various stocks

2) record the different prices interactions clearly

- MOM is a statistical indicator that forecasts whether a sample which follows a trend in the past 10 days will be negative or positive.

- MR forecasts each example's movement in the opposite direction of the most recent price towards the 30-day moving average.

- LSTM is a neural network containing an LSTM layer and a prediction layer. Number of hidden units (U), lag size (T), and weight of regularization term (W) are three hyperparameters that we tune.

- The Attentive LSTM(Qin et al., 2017) is a optimized version of LSTM. We tune U, T, and in the same way as we tune LSTM.

- StockNet(Xu & Cohen, 2018) employs a Variational Autoencoder (VAE) to encode the stock input and a temporal attention model to represent the relevance of distinct timesteps. Here, we use the features from Table 1 as inputs to optimize the hidden size, dropout ratio, and auxiliary rate (α).

We use two measures, Accuracy and MCC (Xu & Cohen, 2018), to assess prediction performance, with ranges of [0, 100] and [1, 1]. The greater the value of the measures, the better the performance.

We use Tensorflow to construct the Adv-ALSTM and then perform optimization by the mini-batch Adam, with an initial learning rate of 0.01 and a batch size of 1024. On the validation set, we look for the best Adv-ALSTM hyper-parameters.

Adv-ALSTM inherits the optimal parameters from ALSTM for U, T, and λ, which are chosen through grid-search within hyperparameter space. We then fine-tune β and ε. Over five separate runs, we present the mean testing performance when Adv-ALSTM performs best on the validation set.

2.11.3 Results

Method	ACL18		KDD17	
	Acc	MCC	Acc	MCC
MOM	47.01±—	-0.0640±—	49.75±—	-0.0129±—
MR	46.21±—	-0.0782±—	48.46±—	-0.0366±—
LSTM	53.18±5e-1	0.0674±5e-3	51.62±4e-1	0.0183±6e-3
ALSTM	54.90±7e-1	0.1043±7e-3	51.94±7e-1	0.0261±1e-2
StockNet	54.96±—	0.0165±—	51.93±4e-1	0.0335±5e-3
Adv-ALSTM	**57.20**±—	**0.1483**±—	**53.05**±—	**0.0523**±—
RI	4.02%	42.19%	2.14%	56.12%

RI denotes the relative improvement of **Adv-ALSTM** compared to the best baseline.
The performance of **StockNet** is directly copied from [Xu and Cohen, 2018].

Table 10: Model Comparison against suggested ALSTM

- In every situation, Adv-ALSTM produces the best outcomes. On the ACL18 (KDD17) dataset, Adv-ALSTM improves Accuracy and MCC by 4.02 percent and 42.19 percent (2.14 percent and 56.12 percent, respectively) when compared to the baselines. This explains adversarial training's success, which may be attributable to improved model generalization by adaptively modelling disturbances during training.

- Specifically, when compared to StockNet, which uses VAE to capture stochasticity in stock inputs, Adv-ALSTM outperforms StockNet. The reason for this, we believe, is that during the training process, since StockNet depends on Monte Carlo sampling, it is unable to explicitly describe the size and direction of stochastic perturbation.

- In terms of Accuracy and MCC, ALSTM beats LSTM by 1.93 percent and 48.69 percent on average, demonstrating the importance of attention(Qin et al., 2017). Furthermore, as predicted, MR and MOM outperform all ML-based approaches, demonstrating that historical patterns aid in stock prediction.

We compare adversarial perturbations to random perturbations to see how successful adversarial training is. Rand-ALSTM is a variant of Adv-ALSTM that adds random disturbances/noise to the clean input examples to create new instances.

Datasets	Acc	MCC
ACL18	55.08±2e0	0.1103±4e-2
KDD17	52.43±5e-1	0.0405±8e-3

Table 11: Compare effect of Adversarial Training

When compared to RandALSTM, Adv-ALSTM improves significantly. For example, it outperforms Rand-ALSTM by 3.95 percent when it comes to Accuracy on ACL18. It shows that adversarial perturbations are useful for stock prediction, just as they were in the original picture classification challenges(Goodfellow et al., 2013). On the two datasets, Rand-ALSTM beats ALSTM, where only clean input examples were used during training, with an average improvement of 0.64 percent vs. Accuracy. This emphasizes the need of dealing with stock characteristics' stochastic properties.

2.12 Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction(Ding et al., 2020)

2.12.1 Business Problem

To solve the stock movement prediction challenge, we present a unique method based on the Transformer in this article. In addition, we provide a number of improvements to the basic Transformer that has been presented. To begin, we suggest a Multi-Scale Gaussian Prior to improve Transformer's locality. Second, in the self-attention multi-head mechanism, we create an Orthogonal Regularization to prevent learning redundant heads. Finally, in order to understand hierarchical characteristics of high-frequency financial data, we create a Trading Gap Splitter for Transformer. The suggested technique has the benefit of mining highly long-term dependencies from financial time series as compared to other prominent recurrent neural networks such as LSTM.

2.12.2 Research Methodology

Because predicting the precise price of a stock is exceedingly difficult, we use [Walczak, 2001]'s setup and anticipate price change instead. Typically, stock price forecasting is approached as a problem of binary classification, with the stock movement being divided into two cases i.e. Fall or Rise.

Given the parameters of the stock $\mathbf{X} = [\mathbf{x}_{T-\Delta t+1}, \mathbf{x}_{T-\Delta t+2}, \ldots, \mathbf{x}_T] \in \mathbf{R}^{\Delta t \times F}$ in the most recent time-periods $\Delta\mathbf{t}$, the prediction model $\mathbf{f}_\theta(\mathbf{X})$ with θ features predict the price movement label $\mathbf{y} = \mathbf{I}(\mathbf{p}_{T+k} > \mathbf{p}_T)$, with target trading time **T**, stock feature dimensions **F**, and $\mathbf{p}_T$ denotes the closing time at **T**. To summarize, the suggested model uses past data from stock **s** in the **lag [T - Δt + 1, T]** (with predetermined lag size Δ**t**) to forecast the price movement class **y** (1 for Rise, 0 for Fall) of next k time-periods.

We utilize two stock data sets to test the suggested methods: NASDAQ and China A-shares market. Table 12 shows the specifics of the two data sets. We will describe the data gathering procedure and present our empirical results from numerical experiments in the subsections that follow. We also do an incremental analysis to determine the efficacy of each suggested Transformer improvement.

Property	Data	
	Daily	15-min
Market	NASDAQ	China A-shares
Start Date	2010/07/01	2015/12/01
End Date	2019/07/01	2019/12/01
Time Frequency	1 day	15 minutes
Total Stocks	3243	500
Total Records	9749098	7928000
Rising Threshold β_{rise}	0.55%	-0.5%
Falling Threshold β_{fall}	-0.1%	0.105%

Table 12: Dataset properties

From July 1st, 2010 to July 1st, 2019, we collected daily quotation data for all 3243 stocks on the NASDAQ stock market, as well as 15-min quotation data for 500 CSI-500 stocks on the China A-shares market starting from December 1st, 2015 till December 1st, 2019. To create candidate examples, we slide a lag window with size of N time periods across this time series data. For the NASDAQ data, we create five datasets with different lag window widths N = 5, 10, 20, and 40 (representing five, ten, twenty, and forty days), and the lag strides are all set to one. We also create 5 datasets for China A-shares data with window sizes N = 5 * 16,10 * 16, 16 * 20, 40 * 16 (denoting five, ten, twenty, and forty days because one trading day in China

A-shares market contains 16 15-minute), and the lag strides for all of them set as 16 (denoting 5-, 10-, 20-, 40-day because one trading day in China A-shares market contains 16 15-minute), and the log strides for all of them set as 16 (i.e. 1 day). Open, high, low, near, and volume are the characteristics utilized in all datasets, and they are all modified and standardized. We use the approach $\mathbf{y = I(p_{T+k} > p_T)}$, where we set k between 1 and 16 (both represent 1 day). Furthermore, we applied two threshold parameters to the labels in both datasets, β_{rise} and β_{fall}, to ensure that there is no class imbalance in each dataset:

$$y = \begin{cases} 1 & \frac{p_{T+k}-p_T}{p_T} > \beta_{rise}; \\ -1 & \frac{p_{T+k}-p_T}{p_T} < \beta_{fall}; \\ \text{abandon} & \text{otherwise.} \end{cases}$$

To eliminate data leaking, we separate training, validation, and test sets in a tight sequential manner.

Method	**Accuracy (%)/MCC** ($\times 10^{-2}$) with window size of K-day			
	$K = 5$	$K = 10$	$K = 20$	$K = 40$
NASDAQ Daily Data				
CNN	52.33/3.16	52.02/2.68	51.84/2.28	52.60/2.52
LSTM	53.86/7.73	53.89/7.72	53.59/7.15	53.81/7.48
ALSTM	54.06/8.35	53.94/7.92	54.05/8.11	54.19/8.56
B-TF†	54.78/8.48	54.84/8.89	54.90/9.13	56.01/9.45
MG-TF†	**55.10/8.98**	**56.18/9.74**	**56.77/10.39**	**57.30/11.46**
China A-shares 15-min Data				
CNN	53.53/2.62	52.25/1.80	52.03/1.81	51.61/1.77
LSTM	56.59/6.42	56.70/6.19	56.18/3.74	54.93/2.98
ALSTM	57.03/8.23	57.42/9.16	55.69/6.65	55.68/6.65
B-TF†	57.14/9.68	57.42/9.52	57.32/9.14	57.55/10.41
HMG-TF†	**57.36/10.52**	**57.79/9.98**	**57.90/10.33**	**58.70/14.87**

Table 13: Model Comparison against suggested Transformers

We use two measures to assess prediction performance: accuracy and MCC

$$\text{MCC} = \frac{\text{tp} \times \text{tn} - \text{fp} \times \text{fn}}{\sqrt{(\text{tp} + \text{fp})(\text{tp} + \text{fn})(\text{tn} + \text{fp})(\text{tn} + \text{fn})}}$$

B-TF : B-TF is an encoder-only Transformer variation with L blocks of multi-head self-attention layers along with position-wise feed forward layers.

MG-TF : Multi-Gaussian Prior and Orthogonal Regularization improvements to the B-TF model

HMG-TF : Trading Gap Splitter has been added to the MG-TF model.

The Adam optimizer is used to implement the solution, with a learning rate of 1e - 4 as a starting point. The mini-batch size is set as 256 and the trade-off hyper-parameter is set at 0.05. Each of the three multi-head self-attention blocks in the TF-based models has four heads. We train the model from start to finish using only raw quotation data and no data augmentation. We exclusively use MG-TF on our NASDAQ dataset since Trading Gap Splitter is less suitable for low-frequency data like daily quotation data.

2.12.3 Results

In all situations, the suggested methods B-TF, MG-TF, and HMG-TF demonstrate considerable increases on both the Accuracy and MCC measures when compared to baselines. It demonstrates that Transformer-based techniques outperform RNN- and CNN-based approaches in terms of performance.

The ability to understand long-term dependencies is better with transformer-based methods. It is particularly difficult for RNN-based methods to discover the interdependencies across a large number of time steps on China A-shares data, where the window of 40 days comprises 640 (40*16) time-periods. While the benefits of the self-attention mechanism allow Transformer-based methods to continuously outperform other approaches as the window grows larger.

The modified Transformer models MG-TF and HMG-TF outperform the standard Transformer model.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

The main objective of this study is to first build our portfolio with high quality assets chosen based on various features extracted from raw price (OHLCV) data. Based on the momentum of the assets, we can choose the winning stocks and bet on the momentum effect and/or go with the losing stocks if we see a reversal effect trend based on historical data. Once the stocks have been finalized for our portfolio, we will evaluate and compare the DL models for price forecasting based on returns/profits achieved. The same models will be built using standard market indexes (NIFTY50, SENSEX) in order to answer our research question i.e. “Does the quality of financial assets matter if we keep the base forecasting model performance as constant?”

Fig 14. depicts the research methodology on a step-by-step manner

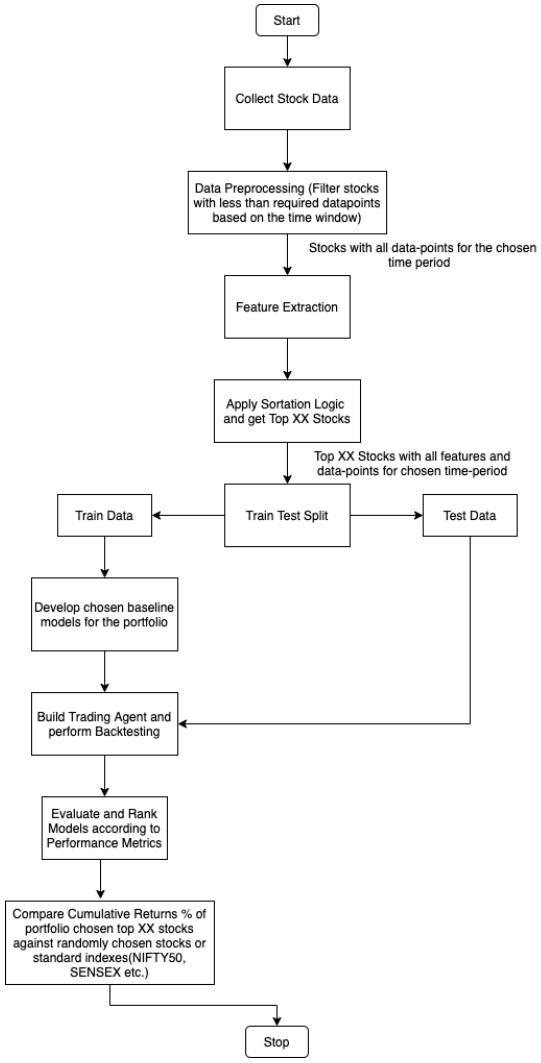


Fig. 14

3.4 Dataset Description

The data is downloaded using nsepy package, an open source python package that offers various methods/functions to download financial data from the official NSE website. It also provides some daily financial information like daily top gainers/losers.

The top 10 equities from each sector from the Indian Stock Market will be utilized as the dataset for equites. We will be using the raw price data(OHLCV) to derive/extract more features which can play a significant role while deciding on the stocks for our investment portfolio.

Column Summary:

Base Attributes from dataset :

Date	Date of the record
Open	Opening Price
High	High for the day
Low	Low for the day
Close	Closing Price
Volume	Total Volume

Table 14: Base Attributes

3.5 Data Preparation and Pre-Processing

The selected ranges of training and testing data encompassed at least one entire market cycle in order to assess the models' performance in diverse market conditions (i.e., bullish, bearish, and oscillating markets). The following are the key pre-processing processes that will be followed at this stage of the study:

- Check for missing/NaN values and treat them if necessary.
- Filter out stocks with less than required data-points based on the time window.
- Perform some basic Exploratory Data Analysis to get an idea of overall trend

3.4 Feature Extraction and Engineering

Since selection of quality assets for our portfolio is an important requirement for this study, we will be applying Feature Engineering to generate new derived features which can be potentially helpful in selection of financial assets qualitatively.

Financial data is unique and needs a methodical approach. First, in compared to datasets used in applications of ML, it is fairly restricted in availability and breadth — in fact, we don't have high-quality data for the great majority of markets prior to the 1990s. Many crucial factors that drive results can be left out of a model or don't get to see the entire range of their values during training. The significantly divergent behaviour of asset prices during high and low market volatility periods, or regimes, is a specific illustration of this difficulty; training the model on a subset of the data that spans only one of the regimes will degrade the model's capacity to generalize well on the test set. The signal-to-noise ratio of financial data is quite low. Models will overfit the data noise because the most powerful models, such as neural networks, which are low bias and high-variance learners. Surely, additional factors such as macroeconomic statistics, sentiment related to monetary policy announcement, and so on might help to describe the overall situation of the market.

Following are some of the derived features which have been identified. This list will be updated and new features may be added during feature extraction and engineering phase of the Research Plan.

Derived Attributes :	
Close Delta	Daily Change in Closing Price
Close Change %	% Daily Change in Closing Price
Market Cap	Daily Market Capitalization
Volume Delta	Daily Change in Volume
Volume Change %	% Daily Change in Volume
Weekly Volatility	Close – Weekly Close Avg.
Monthly Volatility	Close – Monthly Close Avg.
Yearly Volatility	Close – Yearly Close Avg.
Nearness to 52W High	Close – Yearly Close Max
(% positive / % negative) trading days	
% Returns over look back period	
Volatility over look back period	

Table 15: Derived Features

3.6 Stock Sortation Logic

After feature engineering we can now use these new features to sort the stocks and choose the top XX stocks. The sortation logic is as follows :

Look Back Period Returns > Risk Free Rate	Sorting Condition
Stock with Highest momentum(returns %) during look back period	Descending Sort
(% positive / % negative) trading days	Descending Sort
Nearness to 52 week high	Ascending Sort
Volatility or Downside Volatility	Ascending Sort
Top XX Stocks	Filter

Table 16: Sortation Logic

3.5 Model Development

Once we have finalized the stock pool which we will be building our portfolio with based on the previously derived features, we will be employing various DL based models to forecast the prices for the upcoming period using the extracted/derived features.

The Models which we have decided upon after the Literature Revies are as follows :

1. **LSTM (Baseline)**

The LSTM RNN is a kind of RNN that can recall both short and long-term values. When it comes to difficult issues like automated speech recognition and handwritten character identification, many DL model developers use LSTM networks. Time-series data is the most common type of data for LSTM models. Natural Language Processing (NLP), language modelling, language translation, speech recognition, sentiment analysis, predictive analysis, financial time series analysis, and other applications use it. LSTM networks can be more successful on time series data processing, such as language translation, using attention modules and AE structures.

The LSTM is a kind of RNN that is more specialized. As a result, the recommended optimization methods and weight updates are same. Furthermore, the number of hidden layers, the number of units in each layer, network weight initialization, activation functions, learning rate, momentum values, the number of epochs, batch size (minibatch size), decay rate, optimization algorithms, sequence length for LSTM, gradient clipping, gradient normalization, and dropout are all hyperparameters that are similar to RNN. The hyperparameter optimization approaches used for RNN may also be used to LSTM in order to discover the optimum hyperparameters.

2. **CNN-LSTM**

As we have observed from our literature review of (Liu et al., 2017a, p.), their experiments show that the CNN-LSTM neural network model may be successfully used to create quantitative strategies and get greater returns than the Benchmark index.

3. **ALSTM(Feng et al., 2019) (SOTA)**

We observed that Adversarial Attentive LSTM is a model training method that uses adversarial training to mimic stochasticity. We witnessed Feng's comprehensive tests on two benchmark datasets and confirmed the efficacy of the suggested method, demonstrating the relevance of accounting for stock price stochasticity in stock movement prediction. Furthermore, the findings revealed that adversarial training improves the prediction model's resilience and generalization.

4. **Transformer(Ding et al., 2020) (SOTA)**

We witnessed that when compared to baselines, the recommended techniques B-TF, MG-TF, and HMG-TF show significant gains on both the Accuracy and MCC metrics. In terms of performance, it indicates that Transformer-based techniques beat RNN- and CNN-based approaches.

Transformer-based techniques improve the capacity to grasp long-term dependencies. Learning dependencies over so many time steps is particularly difficult for RNN-based techniques.

3.6 Model Evaluation Metrics

The following measures will be used to evaluate all models under consideration:

- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Root Mean Square Error (RMSE)
- Profit/Returns %

The Cumulative Returns % for the portfolio with the top XX stocks will be compared against a portfolio with random stocks or an index like NIFTY, SENSEX etc.

In addition to the indicators listed above, the temporal complexity of the models may be taken into account while assessing them, i.e. the overall amount of features provided, as well as the model chosen, might result in varying levels of temporal complexity, which is important when the models are deployed into production. Since some business decisions require the model response time to be as low as possible, the time complexity of the model becomes an important factor.

3.7 Resource Requirements

3.7.1 Hardware Requirement:

This study will be carried out on a desktop computer that meets the following requirements:

Processor	Intel® CoreTM i9-9900K CPU @ 5.0 GHz
Memory	32 GB DDR4
GPU	RTX 2080 (if required)
Operating System	Windows 10 64-bit

Table 3

3.7.2 Software Requirements:

- Python 3.x
- Jupyter Notebook
- Libraries required for general data analysis and EDA like numpy, pandas, matplotlib
- Libraries required for pre-processing, model selection, feature selection like sklearn
- Libraries required for evaluation metrics
- Any other libraries as and when required

REFERENCES

Arpaci, U., & Karaoglu, S. (2017). A Deep Learning Approach for Optimization of Systematic Signal Detection in Financial Trading Systems with Big Data. *International Journal of Intelligent Systems and Applications in Engineering*, *Special Issue*(Special Issue), 31–36. https://doi.org/10.18201/ijisae.2017SpecialIssue31421

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, *12*(7), e0180944. https://doi.org/10.1371/journal.pone.0180944

Batres-Estrada, B. (2015). *Deep learning for multivariate financial time series*.

Chandra, R., & Chand, S. (2016). Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance. *Applied Soft Computing*, *49*, 462–473. https://doi.org/10.1016/j.asoc.2016.08.029

Chang, P.-C., Liao, T. W., Lin, J.-J., & Fan, C.-Y. (2011). A dynamic threshold decision system for stock trading signal detection. *Applied Soft Computing*, *11*(5), 3998–4010. https://doi.org/10.1016/j.asoc.2011.02.029

Cho, K., Merrienboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*.

Das, P., & Banerjee, A. (2011). Meta optimization and its application to portfolio selection. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '11*, 1163. https://doi.org/10.1145/2020408.2020588

De BONDT, W. F. M., & Thaler, R. (1985). Does the Stock Market Overreact? *The Journal of Finance*, *40*(3), 793–805. https://doi.org/10.1111/j.1540-6261.1985.tb05004.x

Ding, Q., Wu, S., Sun, H., Guo, J., & Guo, J. (2020). Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. *IJCAI*.

Dixon, M., Klabjan, D., & Bang, J. H. (2015). Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. *Proceedings of the 8th Workshop on High Performance Computational Finance*, 1–6. https://doi.org/10.1145/2830556.2830562

Elmsili, B., & Outtaj, B. (2018). Artificial neural networks applications in economics and management research: An exploratory literature review. *2018 4th International Conference on Optimization and Applications (ICOA)*, 1–6. https://doi.org/10.1109/ICOA.2018.8370600

Feng, F., Chen, H., He, X., Ding, J., Sun, M., & Chua, T.-S. (2019). Enhancing Stock Movement Prediction with Adversarial Training. *IJCAI*.

Fischer, T., & Krauss, C. (2017). *Deep learning with long short-term memory networks for financial market predictions* (Issue 11/2017). Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics. https://ideas.repec.org/p/zbw/iwqwdp/112017.html

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2003). Learning Precise Timing with Lstm Recurrent Networks. *J. Mach. Learn. Res.*, *3*(null), 115–143. https://doi.org/10.1162/153244303768966139

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). *Maxout Networks*.

Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. *ArXiv:1308.0850 [Cs]*. http://arxiv.org/abs/1308.0850

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Gyorfi, L., Lugosi, G., & Udina, F. (2006). NONPARAMETRIC KERNEL-BASED SEQUENTIAL INVESTMENT STRATEGIES. *Mathematical Finance*, *16*(2), 337–357. https://doi.org/10.1111/j.1467-9965.2006.00274.x

Hameed, A., & Ting, S. (2000). Trading volume and short-horizon contrarian profits: Evidence from the Malaysian market. *Pacific-Basin Finance Journal*, *8*(1), 67–84.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2018). Deep Learning in Finance. *ArXiv:1602.06561 [Cs]*. http://arxiv.org/abs/1602.06561

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In S. C. Kremer & J. F. Kolen (Eds.), *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huck, N. (2009). Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, *196*(2), 819–825.

Kearney, C., & Liu, S. (2013). Textual Sentiment Analysis in Finance: A Survey of Methods and Models. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2213801

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, *3*(3), 263–286. https://doi.org/10.1007/PL00011669

Khadjeh Nassirtoussi, A., Aghabozorgi, S., Ying Wah, T., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653–7670. https://doi.org/10.1016/j.eswa.2014.06.009

Kraus, M., & Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, *104*, 38–48. https://doi.org/10.1016/j.dss.2017.10.001

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

Lee, S. I., & Yoo, S. J. (2018). Threshold-Based Portfolio: The Role of the Threshold and Its Applications. *ArXiv:1709.09822 [q-Fin]*. http://arxiv.org/abs/1709.09822

Li, B., & Hoi, S. C. H. (2013). Online Portfolio Selection: A Survey. *ArXiv:1212.2129 [Cs, q-Fin]*. http://arxiv.org/abs/1212.2129

Li, Y., & Ma, W. (2010). Applications of Artificial Neural Networks in Financial Economics: A Survey. *2010 International Symposium on Computational Intelligence and Design*, 211–214. https://doi.org/10.1109/ISCID.2010.70

Liu, S., Zhang, C., & Ma, J. (2017a). CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets. In D. Liu, S. Xie, Y. Li, D. Zhao, & E.-S. M. El-Alfy (Eds.), *Neural Information Processing* (Vol. 10635, pp. 198–206). Springer International Publishing. https://doi.org/10.1007/978-3-319-70096-0_21

Liu, S., Zhang, C., & Ma, J. (2017b). *CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets* (p. 206). https://doi.org/10.1007/978-3-319-70096-0_21

M, H., E.A., G., Menon, V. K., & K.P., S. (2018). NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Computer Science*, *132*, 1351–1362. https://doi.org/10.1016/j.procs.2018.05.050

Minami, S. (2018). Predicting Equity Price with Corporate Action Events Using LSTM-RNN. *Journal of Mathematical Finance*, *08*(01), 58–63. https://doi.org/10.4236/jmf.2018.81005

Mittermayer, M.-A., & Knolmayer, G. (2006). *Text mining systems for market response to news: A survey*.

Moritz, B., & Zimmermann, T. (2016). Tree-Based Conditional Portfolio Sorts: The Relation between Past and Future Stock Returns. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2740751

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *ArXiv:1704.02971 [Cs, Stat]*. http://arxiv.org/abs/1704.02971

Samarawickrama, A. J. P., & Fernando, T. G. I. (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–6. https://doi.org/10.1109/ICIINFS.2017.8300345

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks*.

Takeuchi, L. (2013). *Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks*.

Tkáč, M., & Verner, R. (2016). Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, *38*, 788–804. https://doi.org/10.1016/j.asoc.2015.09.040

Xing, F. Z., Cambria, E., & Welsch, R. E. (2018). Natural language based financial forecasting: A survey. *Artificial Intelligence Review*, *50*(1), 49–73. https://doi.org/10.1007/s10462-017-9588-9

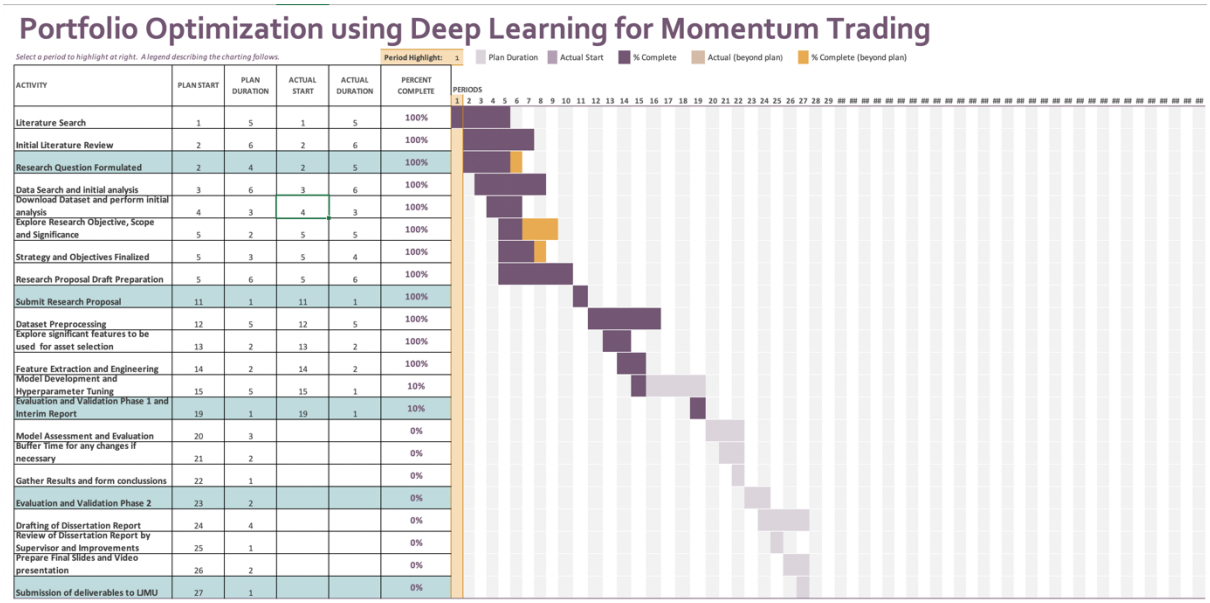
Xu, Y., & Cohen, S. B. (2018). Stock Movement Prediction from Tweets and Historical Prices. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1970–1979. https://doi.org/10.18653/v1/P18-1183

Yuan, Z., Zhang, R., & Shao, X. (2018a). Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation. *Proceedings of the 2018 International Conference on Computing and Data Engineering*, 39–43. https://doi.org/10.1145/3219788.3219793

Yuan, Z., Zhang, R., & Shao, X. (2018b). Deep and Wide Neural Networks on Multiple sets of Temporal data with Correlation. *Proceedings of the 2018 International Conference on Computing and Data Engineering*, 39–43. https://doi.org/10.1145/3219788.3219793

Zhang, L., Aggarwal, C. C., & Qi, G.-J. (2017). Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

APPENDIX A: RESEARCH PLAN



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature Search	1	5	1	5	100%
Initial Literature Review	2	6	2	6	100%
Research Question Formulated	2	4	2	5	100%
Data Search and initial analysis	3	6	3	6	100%
Download Dataset and perform initial analysis	4	3	4	3	100%
Explore Research Objective, Scope and Significance	5	2	5	5	100%
Strategy and Objectives Finalized	5	3	5	4	100%
Research Proposal Draft Preparation	5	6	5	6	100%
Submit Research Proposal	11	1	11	1	100%
Dataset Preprocessing	12	5	12	5	100%
Explore significant features to be used for asset selection	13	2	13	2	100%
Feature Extraction and Engineering	14	2	14	2	100%
Model Development and Hyperparameter Tuning	15	5	15	1	10%
Evaluation and Validation Phase 1 and Interim Report	19	1	19	1	10%
Model Assessment and Evaluation	20	3			0%
Buffer Time for any changes if necessary	21	2			0%
Gather Results and form conclussions	22	1			0%
Evaluation and Validation Phase 2	23	2			0%
Drafting of Dissertation Report	24	4			0%
Review of Dissertation Report by Supervisor and Improvements	25	1			0%
Prepare Final Slides and Video presentation	26	2			0%
Submission of deliverables to IJMU	27	1			0%