INTERCONNECTED FINANCIAL PREDICTION USING TIME-SERIES AND NETWORK DATA

An Undergraduate Research Scholars Thesis

by

MAXWELL HUFFMAN

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Yang Shen

May 2021

Major: Electrical & Computer Engineering

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Maxwell Huffman, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Research Faculty Dr. Yang Shen prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office

TABLE OF CONTENTS

ABSTRACT

Interconnected Financial Prediction Using Time-Series and Network Data

Maxwell Huffman
Department of Electrical & Computer Engineering
Texas A&M University

Research Faculty Advisor: Yang Shen
Department of Electrical & Computer Engineering
Texas A&M University

The digital technology age has created an unprecedented production and consumption of data. With this surge, data-analytics have become increasingly important in industries that can generate value based on predictive models: online retail, meteorology, transportation, social media, etc. This paper will focus on the finance industry because it contains years of available time-series data, a high correlation between prediction accuracy and generated value, and a high ceiling for creating a perfect prediction. Historically, time-series forecasting has been used to predict how different companies' stock prices might behave in the future given how they have behaved in the past. However, focusing solely on sequential data for prediction is missing the potential of interdependency within the information. Companies do not exist in a vacuum; they exist in markets that can heavily influence multiple entities at once: government regulation, supply and demand, etc. By grouping these entities and examining their relationships in the form of graph based networks, interdependent data can be used in conjunction with sequential data to improve prediction accuracy. In this thesis, Long Short-Term Memory (LSTM) deep neural networks will act as the foundation for which Graph Convolutional Neural networks (GCN) will be built atop each other to combine both sequential and relational embeddings in formulating stock price prediction. To verify this hypothesis, closing stock price data from over a five year period was used.

ACKNOWLEDGMENTS

Contributors

I would like to thank my faculty advisor, Dr. Yang Shen, his research student, Yuning You, and my undergraduate senior design teaching assistant, Sambandh Dhal, for their guidance and support throughout the course of this research.

Thanks also go to my friends, colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

The data analyzed for this project was provided by the Electrical & Computer Engineering Department at Texas A&M University. All other work conducted for the thesis was completed by the student independently.

Funding Sources

Undergraduate research was supported by the Electrical & Computer Engineering Department at Texas A&M University.

NOMENCLATURE

NASDAQ	National Association of Securities Dealers Automated Quotations
NYSE	New York Stock Exchange
TYO	Tokyo Stock Exchange
B3	São Paulo Stock Exchange
JPX	Japenese Exchange Group
MA	Moving Average
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
GCN	Graphical Convolutional Network

1. INTRODUCTION

The finance market is an ever growing industry. As of August 2020, global market capitalization reached 87.8 trillion US dollars, which is a 37% increase from 64.3 trillion US dollars in December 2015. The total value of money invested in the stock market is increasing every year, on average [1]. Outside of fundamental company values, entities invest their money in financial markets because it is more profitable than letting inflation reduce their individual dollar value. Choosing incorrect companies to invest in though, can be devastating to an investor's portfolio, which is why many investors select index or mutual funds that diversify their capital among many companies instead of few. This reduces volatility and risk, and has historically been the safest method of increasing capital value. These funds are limited to growing at the pace of the market, which decreases both upside and downside. However, if only the best companies were selected from day to day, capital growth could be multiplied significantly beyond average market increase. Therefore, there is always a critical need of predictive models to optimize the investment options and maximize the investment return.

In this chapter, we will first introduce traditional analytic methods and emerging machine-learning methods (in particular, deep neural networks) for financial market prediction. We will then describe the mathematical notations and accordingly define the problem in this study. We will end this chapter with the organization for the rest of the thesis.

1.1 Investment Fundamentals

There are two types of analyses for financial market prediction: fundamental and technical analyses. Fundamental analysis determines a company's value based on economic environment, industry status, internal financial performances, and more. Technical analysis evaluates historic movements of price and volume to predict future outcomes under the assumption that a stock's numeric value already incorporates fundamental data intrinsically. Both types of analyses are used in varying amounts across individuals and companies to predict which stocks will rise and which stocks will fall. However, stock markets have existed for hundreds of years, and many people have attempted to create perfect models for capital growth. While it has been shown that no investment portfolio manager has ever accurately and consistently forecast the stock market [2], the field is divided on whether or not the stock market can be regarded as truly random [3]. In the field of machine learning, many state of the art models have been created for technical analysis; however, company stocks can move in aggregate based on external factors, and fundamental analysis of markets demonstrates potential for prediction improvement [4].

1.2 Deep Neural Networks

Historically, machine learning models in finance have focused on technical analysis, which generates predictions based on analyzing time-series data sets. Examples include convolutional and recurrent neural networks. Baseline time-series data consists of price and volume, and using regression to predict their future values is a common technique for forecasting future movements of stock price. They've been utilized for time series prediction on the NYSE, B3, JPX, and TYO stock exchanges with varying degrees of success [5] [6] [7], but we are currently on the edge of new discoveries with state-of-the-art deep learning models that utilize graph structures to yield increased performance [8].

1.3 Mathematical Notations

Throughout the paper, capital script letters (e.g. $\mathcal{X}$) denote tensors, bold capital letters (e.g. $\mathbf{X}$) denote matrices, and bold lowercase letters (e.g. $\mathbf{x}$) denote vectors. Constants are represented by normal uppercase letters (e.g.N) Scalars are represented by normal lowercase letters (e.g. x) and hyperparameters are represented by Greek symbols (e.g. $\alpha, \beta, \ldots$). Subscripts i and j denote the i-th row and j-th column of a given matrix, where i and j span the total number of entities; the superscript t represents the time-step where a given scalar, vector, or matrix stores data.

For matrix operations, $\cdot$ represents matrix multiplication, and $\odot$ represents element-wise multiplication. For 3-dimensional tensors with a time dimension, the previous operations will be applied to their slices at various time points (thus matrices) for each time-step within the time dimension. As an example, to calculate $\mathcal{C} = \mathcal{A} \times \mathcal{B}$, where $\mathcal{A} \in \mathbb{R}^{X \times T \times Y}$, $\mathcal{B} \in \mathbb{R}^{Y \times T \times Z}$ and $\mathcal{C} \in$

$\mathbb{R}^{X \times T \times Z}$ each time step in the time (T) dimension of $\mathcal{C}$ is the result of $\mathbf{A}^t \times \mathbf{B}^t$ where $\mathbf{A}^t \in \mathbb{R}^{X \times Y}$ and $\mathbf{B}^t \in \mathbb{R}^{Y \times Z}$.

1.4 Problem Definition

On each day t, given historical price data in a fixed-sized time window, our goal is to select the stock(s) with the highest potential to increase in value of the next day price so that we can sell the stock on day $t+1$ and iterate the process for the testing set. To determine which stock increases the most between two consecutive time steps we use the return ratio:

$$\hat{r}^{t+1} = \frac{\hat{p}^{t+1} - p^t}{p^t} \qquad \text{(Eq. 1)}$$

Given the predictions for day $t+1$, represented as $\hat{p}^{t+1}$, we can compare the predicted price with the ground-truth price on the previous day, p^t, to predict the next-day return-ratio $\hat{r}^{t+1}$. The company with the highest predicted return ratio,will be the highest-ranked choice for day $t + 1$ according to the model having seen history within the specified window size. To achieve this goal we minimize the loss function L_R described in Section 3.1.3.

1.5 Trading Strategy

To determine the performance of each model after training, the following trading strategy is used: Each model is given \$10,000 for investing and will purchase the highest ranked stock(s) at their closing price on day t, which we can assume is equivalent to the opening price on day $t + 1$ if the model trades during the after-market. At the end of day $t + 1$, the model will sell the shares it purchased on the previous day and purchase the next highest rated stock(s) for day $t + 2$, having seen price history up to $t + 1$ within the time window. The model will spend no more than \$10,000 per day, and will invest the entire portfolio if it has fallen below the \$10,000 thresh hold.

1.6 Organization

The remainder of the paper is organized as follows: Chapter 2 will lay the foundation for preliminary information such as mathematical notation, data types, and assessment metrics. Chapter 3 will explain the theory behind the temporal model used in this thesis and the results it achieved on our data set. Chapter 4 will expand upon Chapter 3 by introducing a graph based structure to enhance predictions to be interdependent rather than independent. Chapter 5 details our strategies for unraveling the mysteries inside our models, and we will conclude with Chapter 6, where we comment on overarching results of our models and further research that can be explored.

2. GROUNDWORK

The chapter covers our coding framework, how data was collected and evaluated, their dimensionality, their sources, and our baseline prediction strategies. We will also define our metrics for assessing model performance.

2.1 Coding Environment

All machine learning code for this project was built using Tensorflow 2.3 and implemented using Python 3.8. JupyterNotebook, Ipywidgets, bqplot, and Matplotib were used in conjunction to create interactive models and plots for viewing and interpreting the data. Resources for reproducing the results in this paper can be found in our GitHub repository.[1]

2.2 Data Types

2.2.1 Time-Series Data

The temporal training data used as the input for each model is stored as a tensor $\mathcal{X} \in \mathbb{R}^{N \times T \times D}$ where N is the number of entities, T is the number of time-steps, and D is the number of features. The output labels are stored as $\mathcal{Y} \in \mathbb{R}^{N \times T}$; given T time-steps as input, the output is T time-steps shifted 1 day into the future.

2.2.2 Network Data

For relational data, the initial information is stored in a one-hot encoded adjacency matrix, $\mathcal{A} \in \mathbb{R}^{N \times N}$, where a 1 represents a relationship and a 0 represents nothing. This adjacency structure models a graph network where each company is a node and each relationship is an edge.

2.3 Data Collection

The project uses data from the NASDAQ stock exchange as curated in [5], which can be found on their GitHub repository[2].

[1]https://github.com/maxoflife-student/TAMU-ECEN-404-IFPTSND
[2]https://github.com/fulifeng/Temporal_Relational_Stock_Ranking

2.3.1 Time-Series Data

The dataset spans from 01/02/2013 to 12/08/2017, which is 1239 time-steps of trading days after pruning, and contains the closing prices of 1,026 trading companies. However, in place of masking, we reduced the data set to 880 companies to remove entities that contained missing values for days that they were not publicly available for trading. For every company, 5 features were used as input data: closing price and the 5-day, 10-day, 20-day, and 30-day moving averages (MA) of the closing price. Additionally, each feature was normalized by dividing by the maximum value within each time series; this guarantees that all features are bounded between 0 and 1. There are various normalization techniques that can affect the performance of time-series, so we note that this could be expanded in further research [9]. The features for Apple Inc. (AAPL) can be seen as an example in Table 2.1.

Table 2.1: Example Input Features after Normalization for AAPL

Closing Price	**5D-MA**	**10D-MA**	**20D-MA**	**30D-MA**
0.429273	0.439414	0.428402	0.425433	0.441990
0.431209	0.427177	0.427924	0.425488	0.441073
0.433480	0.424421	0.426965	0.425639	0.440044
...	...	...	...	...

Each entities' time-series data was saved using the comma-separated value (CSV) format and such data for all companies were stored in the same directory with their company name as the file name; this allows the relational code to categorize the companies by name.

2.3.2 Network Data

For relational data between companies, they are divided into 92 industries derived from company classification for stocks within the NASDAQ. All companies that share an industry are designated as having a relationship. All relations are stored within a .JSON file containing key values corresponding to industry type. Each key contains a list of all companies within that industry.

After the previous data set was reduced from 1,026 to 880 companies, the network data file was also cleaned of companies that were removed from the data set.

2.4 Assessment Metrics

For the models compared in this thesis, 4 assessment metrics will be used to evaluate their performance. While the Intraday Return Ratio is the most important, the Mean-Squared-Error, Mean Rank, and Diversity provide a more holistic picture for model assessment, reducing the black-box nature of their results.

1. Intraday Return Ratio (IRR)

   The IRR for a given testing set is the total return ratio of a model's portfolio value between the first and last days of the testing set. This metric is extremely important because the model with the highest IRR is the ultimate goal of stock prediction. However, a model can be successful due to many reasons, one of which is random chance, so other metrics need to be evaluated to ensure the model has learned a pattern rather than blindly generated value.

2. Mean-Squared-Error (MSE)

   The MSE is a standard metric for evaluating model accuracy in regression problems. The more accurately a model can predict the next-day price of a stock, the more accurate it should be in predicting which stock will have the highest next-day return ratio. However, if the model has a distribution of accuracy between high-value stocks and low-value stocks, then a lower MSE might not necessarily mean a higher value return.

3. Mean Rank (MR)

   The MR is the average ground truth rank position of the stock selected by the model. If the model selected the stock with the highest ground truth return ratio for each day of the testing set, i.e., a perfect prediction, this score would be 1. If the model selected the stock with the lowest ground truth return ratio for each day in the testing set, the score would be N, the total number of available choices to make each day. This metric is valuable because it demonstrates performance based on rank-ability at each time step rather than at the end of the test set.

4. Diversity

   The diversity of a model is a simple validation that the model is making varied decisions day to day rather than selecting one entity and purchasing it for the entire testing period. Let N_{u} be the number of unique choices made on a given data set, and let T_{test} be the number of days in a testing set:

$$Diversity = \frac{N_{\text{u}}}{T_{\text{test}}} \tag{Eq. 2}$$

If the diversity is 1, then a unique stock choice was made on each day of the testing set. If the diversity is approximately 0, then the model only selected the same few stocks during the testing set. A low diversity score is not always a negative performance indicator for a model.

2.5 Baseline Average for Comparison

To realistically evaluate each model's performance, its IRR is compared to the IRR of a baseline model that equally divides its capital among all stocks on every time step as a lower limit. This mimics the least volatile and lowest risk investment strategy: dividing capital equally among all stocks in the market. This metric is useful for comparing models on the same data set since long periods of negative return ratios are possible depending on external factors: recession, market crash, or a global pandemic. A model might be able to achieve significantly less loss than the aggregate average, so this baseline comparison ensures that the market itself is considered when evaluating the model's performance. Regardless of the state of the market, a model that is unable to achieve a higher IRR than the aggregate average is a failure, since its stock choices are no better than the expected value.

3. MODELING TEMPORAL DATA

In this chapter we study machine learning models that make use of temporal data. An LSTM [10] network is used as the foundation to capture long-term temporal dependencies on the time-series dataset. Earlier Recurrent Neural Networks (RNN), which are the predecessors for LSTM networks, focus on pattern recognition and prediction for data formatted in a sequence: speech recognition, natural language processing, or video classification. They have been widely used in many fields of data science for learning systems that can be improved with historical information. However, earlier RNN models suffer from the "vanishing gradient problem", where no useful gradient information can back-propagate deeply to train the models [11]. LSTM networks avoid this by using cell-states that can capture dependencies on hundreds of time steps without losing relevant information. In this chapter, an LSTM network will be trained over the time-series features of company stock closing prices predict the next-day closing price of each stock; using this information, we implement the trading strategy described in Section 1.4 using the resulting predictions.

3.1 Methods

3.1.1 LSTM Model

State information is passed forward through LSTM cells through a vector $\mathbf{C}_t$, which allows prior information to be passed to future outputs. Within each cell is a forget gate ($\mathbf{f}_t$), input gate ($\mathbf{i}_t$), and output gate ($\mathbf{o}_t$) each with dimensionality $\mathbb{R}^{\mathbb{U}}$, where U is size of the hidden layer. The forget gate contains a learnable parameter, $\mathbf{W}_f$, to determine the magnitude of how much old information will be utilized from the previous timestep's LSTM cell; the input gate contains a learnable parameter, $\mathbf{W}_i$, to determine the magnitude of how much the current input and previous output will be utilized, and the output gate contains a learnable parameter, $\mathbf{W}_o$, which is multiplied by a hyperbolic tangent activation of the sum of input and forget gates, $\mathbf{C}_t$, to calculate the final output [12].

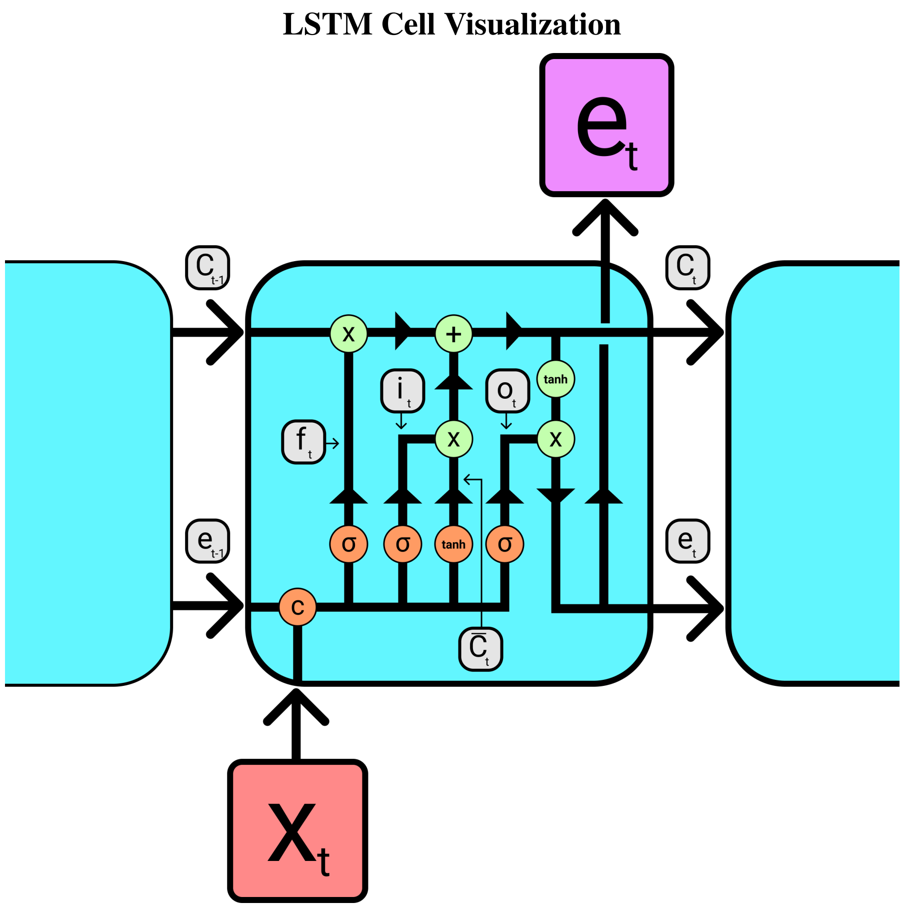

Figure 3.1: Internal information gates of an individual LSTM cell remade based on figures in [12]

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{e}_{t-1},\, \mathbf{x}_t] + \mathbf{b}_f) \qquad \text{(Eq. 3)}\\
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{e}_{t-1},\, \mathbf{x}_t] + \mathbf{b}_i)\\
\hat{\mathbf{C}}_t &= \tanh(\mathbf{W}_C[\mathbf{e}_{t-1},\, \mathbf{x}_t] + \mathbf{b}_C)\\
\mathbf{C}_t &= \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \hat{\mathbf{C}}_t\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{e}_{t-1},\, \mathbf{x}_t] + \mathbf{b}_o)\\
\mathbf{e}_t &= \mathbf{o}_t * \tanh(\mathbf{C}_t)
\end{aligned}
$$

Note: the subscript t used within the LSTM cells represents an index for past and future cells, while the subscript t on output and input vectors represent which time step they originate from.

While specific internal implementation may vary, Figure 3.1 is the general structure and dimensionality of an LSTM cell. To formulate the final prediction for each company's next day price, the output embeddings of each cell, $\mathbf{e}_i^t \in \mathbb{R}^U$, are summarized as a layer to encompass all time steps and companies as $\mathbf{E}_t \in \mathbb{R}^{N \times T \times U}$, which is fed into a dense layer with output dimensionality $\mathbb{R}^{N \times T}$ so that all next day time steps can be calculated and trained for each company.

3.1.2 Hyperparameters

3.1.2.1 Batch Size

When training an LSTM model in Tensorflow, the input data is of shape [batch size, number of time steps, number of features]. This means our data, which is of dimensionality $\mathbb{R}^{N \times T \times D}$, uses companies as the batch-dimension. The batch-size hyperparameter was tuned after realizing that batches of different sizes were affecting how much the loss function could be minimized. By changing the batch size, we divide the initial input data into varying batches of companies that are passed through the model. For example, with a batch size of 10, the input is of dimensionality $\mathbb{R}^{10 \times T \times D}$ and the output is $\mathbb{R}^{10 \times T \times 1}$. After each batch, the internal model parameters are updated using the loss function. The results of varying the batch size for a subset of our data are shown in Figure 3.2.

3.1.2.2 Window Size

On day t, the model is given a window of historical price data as input which spans $[t - WindowSize,\ t]$. Using this data, the model will form a prediction for days $[t - WindowSize + 1,\ t + 1]$. As t is incremented, the sliding window of input data shifts so that the most recent data is appended to the front and the oldest data is truncated from the back.

Using the implementation of the LSTM model provided by TensorFlow, we have the option of implementing our model with return sequences. Using return sequences expands the output dimension of our model from $\mathbb{R}^N$ to $\mathbb{R}^{N \times T}$ so that a prediction is generated between every time step rather than just the final one. Additionally, this requires that the input time dimension of the model remain static, so the training and validation splits must be identical for training.

3.1.3 Loss Functions

3.1.3.1 Mean-Squared-Error

The MSE is a common loss function used in regression models. It converges to 0 as the predicted value converges to the ground truth. In our model, the output predictions are price, but the ground truth and predicted return ratios are used for calculating the MSE. See Eq. 1 for calculating r from p.

$$L_{\text{MSE}}(p^t, \hat{p}^t) = \left\| r^t - \hat{r}^t \right\|^2 \tag{Eq. 4}$$

For predicting the next day closing price of each stock using regression, the MSE is calculated between the predicted return ratio and the ground truth return ratio at time t. However, since the primary objective of the model is to choose the most profitable stock(s) regardless of their ground truth closing price, a pair-wise ranking-aware loss term [5] is added to the loss function.

3.1.3.2 Rank Oriented Pair-Wise Margin Loss

$$L_R(p^t, \hat{p}^t) = \left\| r^t - \hat{r}^t \right\|^2 + \alpha \cdot \frac{\sum_{i=0}^{N} \sum_{j=0}^{N} ReLU(-(\hat{r}_i^t - \hat{r}_j^t)(r_i^t - r_j^t))}{N^2} \tag{Eq. 5}$$

The first term encourages the model to correctly predict closing price by punishing predictions based on their difference with the ground truth return ratio. The second term encourages the relative order of each stock's return ratio to match the ground truth's order as a pair-wise max-margin loss [13]. For each entity pair, the difference between their predicted return ratios is multiplied by the difference between their ground truth return ratios with a sign inversion; A ReLU function is applied, creating a loss term that is nonzero and increases for entity pairs that are incorrectly ranked. The final values of the second term are averaged to avoid scaling the loss function with the number of entities or time steps present in the training data. Additionally, the hyperparameter α is added and tuned to balance the impact of the rank-loss term which might have a vastly different magnitude than the MSE. The results of varying the magnitude of α is shown in

Figure 3.4.

3.1.4 Adam Optimizer

We used the Adam optimizer proposed in [14] to optimize the parameters of our models. The internal mechanisms for controlling the learning rate for gradient descent are adaptive and require little modification to achieve optimal results.

3.1.5 Implementing the Trading Strategy

In order to implement the trading strategy described in Section 1.5, we calculate the next day return ratios as a vector, $\hat{\mathbf{r}}^{t+1}$, described in Eq. 1 using the output predictions, $\hat{\mathbf{p}}^{t+1}$, of the LSTM model. Since each value in the vector corresponds to an individual company's predicted next day return ratio, we select the index with the maximum value to determine the company selection. For the top k selections, this process is iterated on a single time-step until the k highest stocks have been selected. Once the top stock(s) have been selected, \$10,000 from the portfolio is multiplied by the average ground truth return ratios for the selected stock(s) from $\mathbf{r}^{t+1}$ + 1.

3.2 Results

3.2.1 Hyperparameter Tuning

3.2.1.1 Batch Size Tuning

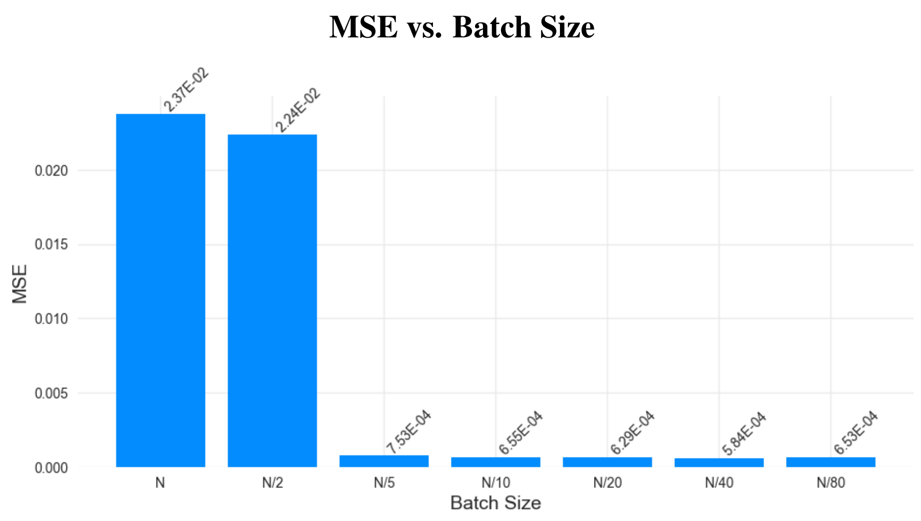

Figure 3.2: Comparison of the average prediction MSE for the validation set between LSTM models of varying batch sizes

As seen in Figure 3.2, when the batch size is reduced to $N/5$ or lower, the average MSE on the validation set significantly drops. This pattern continues with incrementally smaller batch sizes but does not yield the same magnitude of increased accuracy.

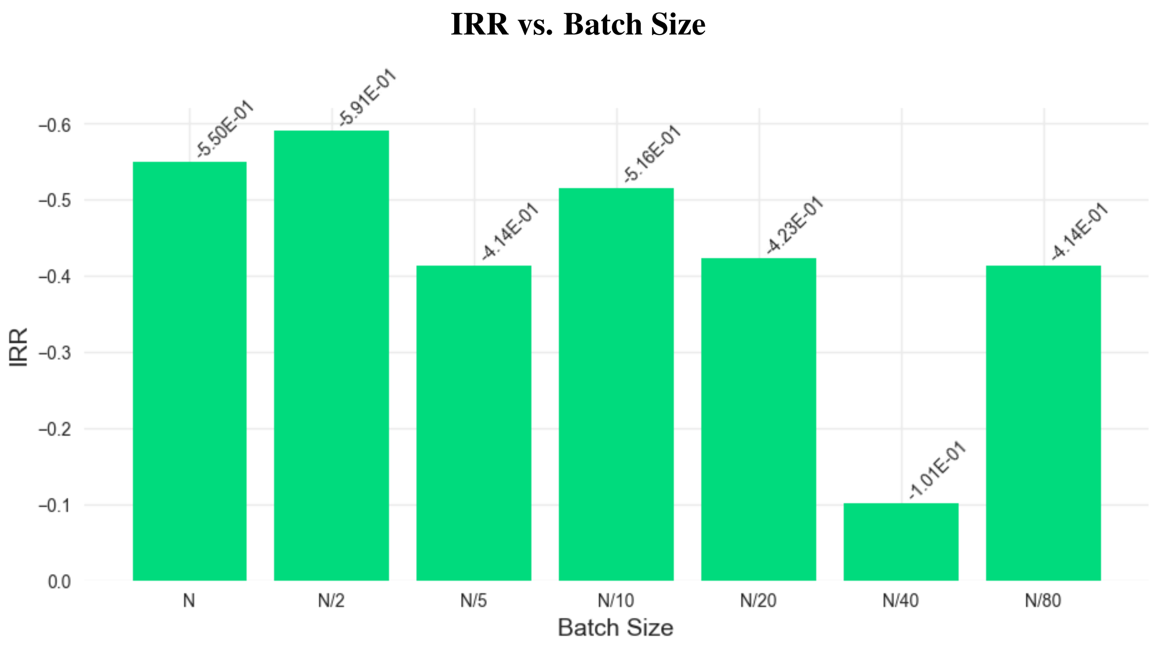

Figure 3.3: Comparison of the IRR at the end of the validation set between LSTM models trained with various batch sizes

As seen in Figure 3.3, a batch size of N/40 yielded the highest IRR on the validation set. There is a trend that decreasing the batch size increases the IRR, but the N/40 variation yields the highest return. Thus, we adopt $N/40$ for the remainder of the thesis. **Note: On this dataset Y-axis is flipped.**

3.2.1.2 Rank Loss Tuning

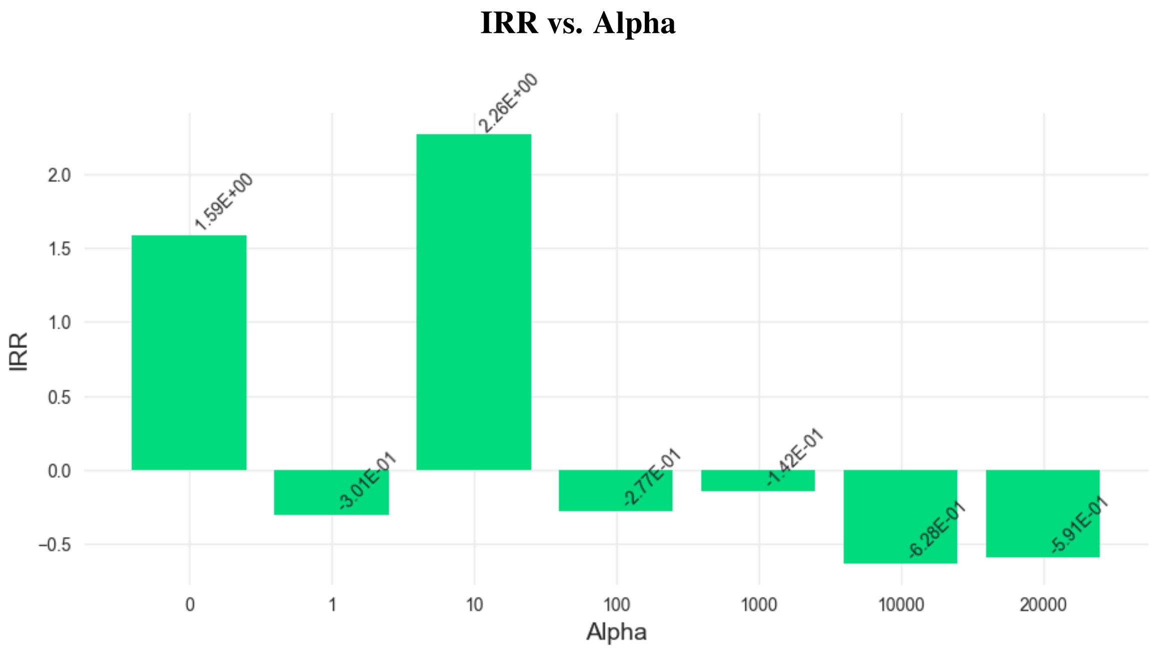

Figure 3.4: Comparison of the IRR at the end of the validation set between LSTM models of varying alpha magnitudes

Table 3.1: Validation Metrics for Various α Values.

α	**MSE**	**IRR**	**MR**	**Di**
0	8.03e-04	1.5890	438.19	39.19%
1	7.68e-03	-0.3014	384.35	9.09%
10	5.91e-04	2.2649	421.79	28.69%
100	5.73e-04	-0.2767	358.40	9.70%
1000	8.27e-04	-0.1417	457.81	13.94%
10000	1.11e-02	-0.6280	540.59	29.29%
20000	3.15e-02	-0.5910	407.05	11.11%

As can be seen in Figure 3.4, the LSTM model with $\alpha = 10$ yielded the highest IRR after making predictions on the validation set. We note that the model with $\alpha = 0$ is equivalent to the baseline MSE model with no regard to stock ranking, which performed worse. These results attest to the importance of both MSE and rank-based loss functions.

3.2.2 Tuned Temporal Model Performance

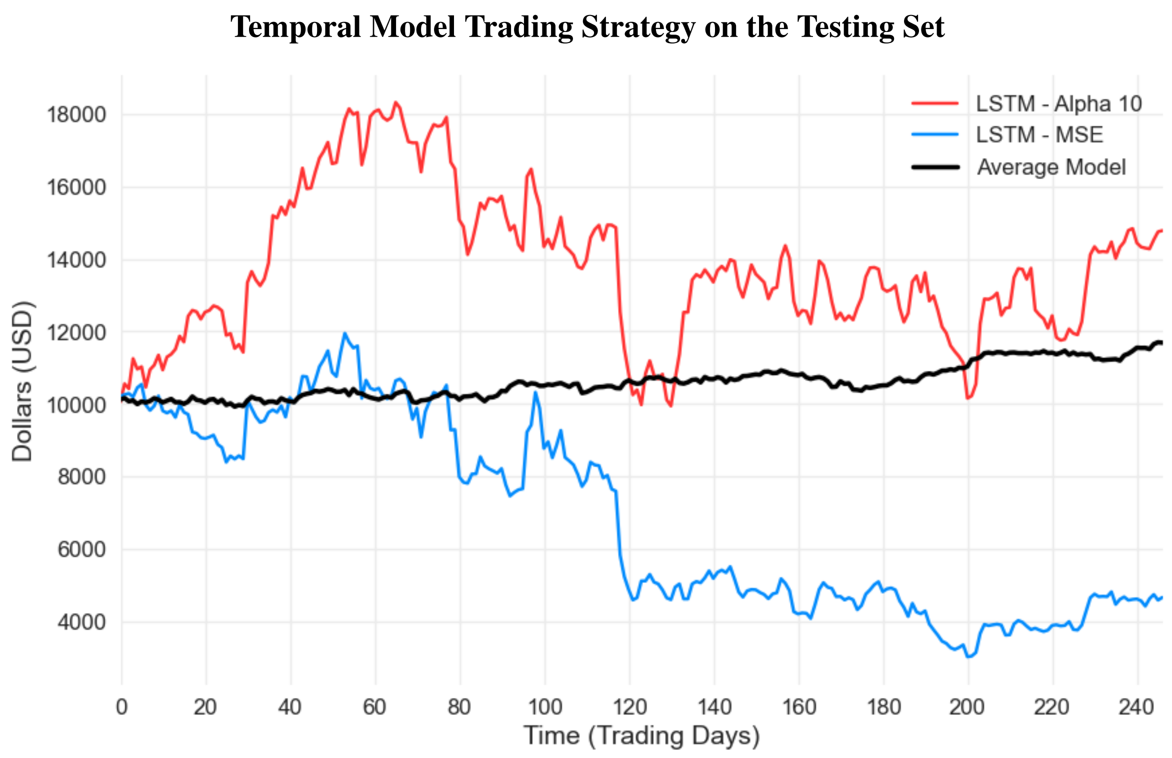

Figure 3.5: The baseline LSTM model and the model trained with a rank aware loss function are compared against the testing set average

Table 3.2: Comparison between baseline LSTM, rank oriented LSTM, and the testing set average

Model	**MSE**	**IRR**	**MR**	**Di**
Average	-	0.1689	-	-
LSTM (MSE)	5.03e-04	-0.5336	374.53	31.33%
LSTM ($\alpha = 10$)	6.30e-04	0.4782	449.90	42.17%

3.3 Conclusions

3.3.1 Decreased Batch Size

One hypothesis for why decreasing the batch size increases model accuracy is that a lower batch size encourages more generalized prediction results [15]. If the model updates the internal parameters after viewing each subset of companies, the outlier entities are less likely to affect the standard entities since their impact on the overall model is reduced by a factor of the batch size. Since a batch size of $N/40$ produced the highest IRR on its validation set, it will be used for the remainder of the thesis within the LSTM model.

3.3.2 Rank Oriented Loss Function

By introducing a rank-oriented loss function [5] into our temporal model, we were able to significantly increase model performance compared to both the standard LSTM with an MSE loss function and the average return rate for the testing period. With an increase in model performance, this model is better equipped for the next chapter in which we will build upon the temporal embeddings by introducing relational information.

4. JOINTLY MODELING TEMPORAL AND RELATIONAL DATA

In this chapter, we aim to further improve model performance by using additional data beyond temporal. Whereas LSTM models treat temporal data of individual companies as independent, new structures will be introduced to consider the dependency among companies. We will do this using a GCN, which is currently a state-of-the-art deep neural network structure that's found success in protein-protein interactions, social networks, transportation, and many more fields [16]. Our GCN will learn parameters over a graph structure where each company is a node with edges connected to other companies. It will be combined with our pre-trained LSTM by smoothing its embeddings using learned parameters from relational data. Predictions will not only be based on historical information of a company, but also the historical information of all companies in its shared industry. Using sector-industry relationships described in Section 2.3.2 as a baseline, we hope to learn relationships utilizing the output embeddings of the LSTM in both explicit and implicit forms. We will expand upon previous work by implementing a different variation of the explicit model.

4.1 Methods

4.1.1 LSTM + Aggregate Smoothing

The output embeddings of the LSTM model have already captured the time-dependent patterns of stock price. The goal of the relational layer is to further enhance those sequential embeddings by introducing a learned, interdependent connection between companies within the same industry. Initially, the predicted closing price of each stock is dependent only upon its own prior history; after inserting an aggregate smoothing layer into the model, the predicted closing price will also be dependent on the values of its neighbors.

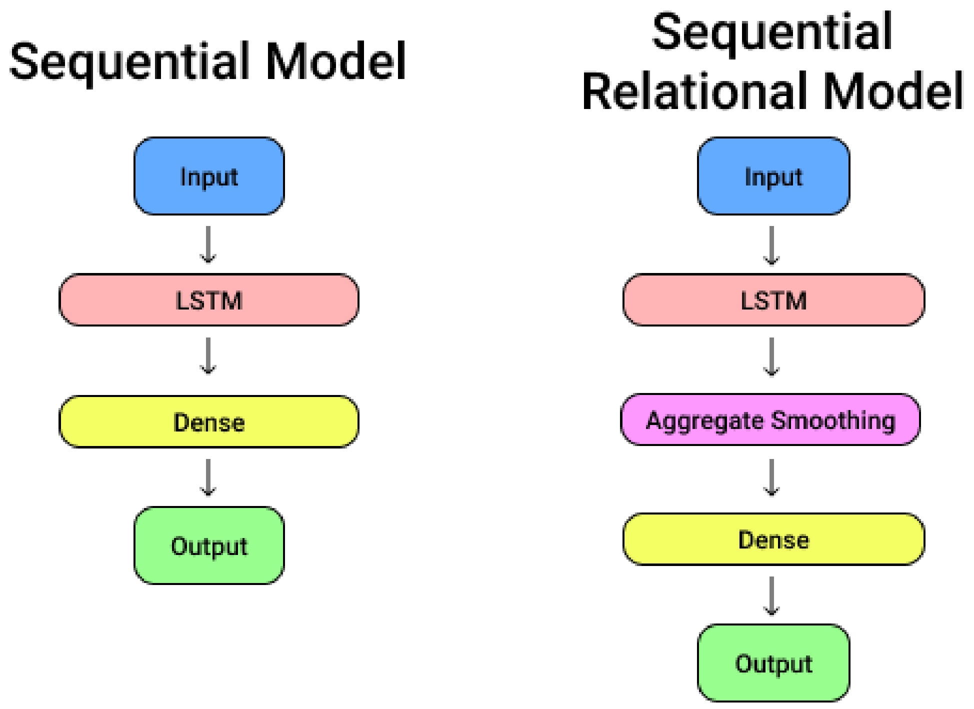

Figure 4.1: Comparison of network layers between the temporal and temporal-relational models

4.1.2 Modeling Relationship Strength

Expanding upon the research done in [5], we model a GCN using aggregate feature smoothing where the strength function, g, is learned from the output sequential embeddings of the temporal model. This introduces interdependencies into the model before finalizing its prediction values. Let n_r be the number of relationships between any company i and j.

$$\bar{\mathbf{e}}_i^t = \sum_{j=0}^{n_r} \frac{g(\mathbf{a}_{ij}, \mathbf{e}_i^t, \mathbf{e}_j^t)}{d_j} \mathbf{e}_j^t \qquad \text{(Eq. 6)}$$

In the following sections, we will outline different variations of the strength function g, that utilize the output embeddings and relational data in different ways.

4.1.2.1 Explicit Model

For the explicit model, we let the weight matrix $\mathbf{W}_g$ learn a strength value using only the given adjacency matrix. The output embeddings of company i and j are multiplied using the dot product, which is an explicit and intuitive way to determine the similarity between the two entities at a given time step t. The more highly correlated that two companies' histories are, the higher the the value of the strength function:

$$g(\mathbf{a}_{ij}, \mathbf{e}_i^t, \mathbf{e}_j^t) = ({\mathbf{e}^t}_i^\top \mathbf{e}_j^t)\, \phi(\mathbf{w}_g^\top \mathbf{a}_{ij} + b_g) \qquad \text{(Eq. 7)}$$

4.1.2.2 Implicit Model

In the implicit design, we let the weight matrix $\mathbf{W}_g$ learn the strength value based on the given adjacency matrix, output embeddings for company i, and the output embeddings for company j. Instead of forcing a strength function based on similarity, the weight matrix should be capable of learning the relationships *implicitly* without the need for explicitly defining it.

$$g(\mathbf{a}_{ij}, \mathbf{e}_i^t, \mathbf{e}_j^t) = \phi(\mathbf{w}_g^\top [\mathbf{a}_{ij},\ \mathbf{e}_i^t,\ \mathbf{e}_j^t] + b_g) \qquad \text{(Eq. 8)}$$

4.1.2.3 Expanded Explicit Model (Explicit+)

To expand upon the explicit design, we hypothesize that it might be possible to have the benefits of explicitly defining the similarity function between two entities, but also reap the benefits of learning the strength value using a weight matrix and non-linear activation function. Let $\mathbf{S}^t \in \mathbb{R}^{N \times N}$ contain all $({\mathbf{e}^t}_i^\top \mathbf{e}_j^t)$ pairs.

$$g(\mathbf{a}_{ij}, \mathbf{e}_i^t, \mathbf{e}_j^t) = \sigma(\mathbf{w}_e^\top \mathbf{s}^t)\, \phi(\mathbf{w}_g^\top a_{ij} + b_g) \qquad \text{(Eq. 9)}$$

4.2 Results

4.2.1 Highest Rank Choice for GCN Models

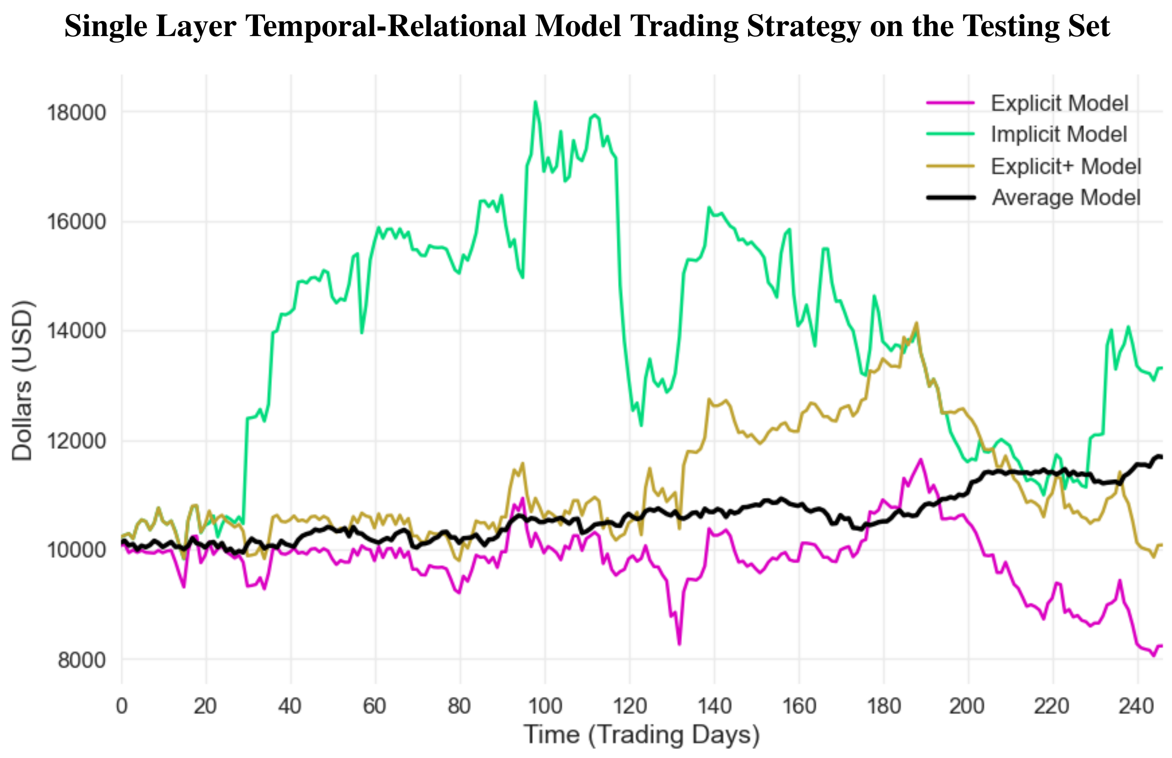

Figure 4.2: Temporal-relational models are compared on the testing set compared to average: explicit, implicit, explicit+

Table 4.1: Comparison between the explict, implict, explicit+, and the testing set average

Model	**MSE**	**IRR**	**MR**	**Di**
Average	-	0.1689	-	-
Explicit	2.59e-02	-0.1761	218.02	0.80%
Implicit	1.94e-03	0.3314	426.53	15.26%
Explicit+	1.88e-02	0.0082	335.19	2.01%

4.2.2 Top k Choices for GCN Models

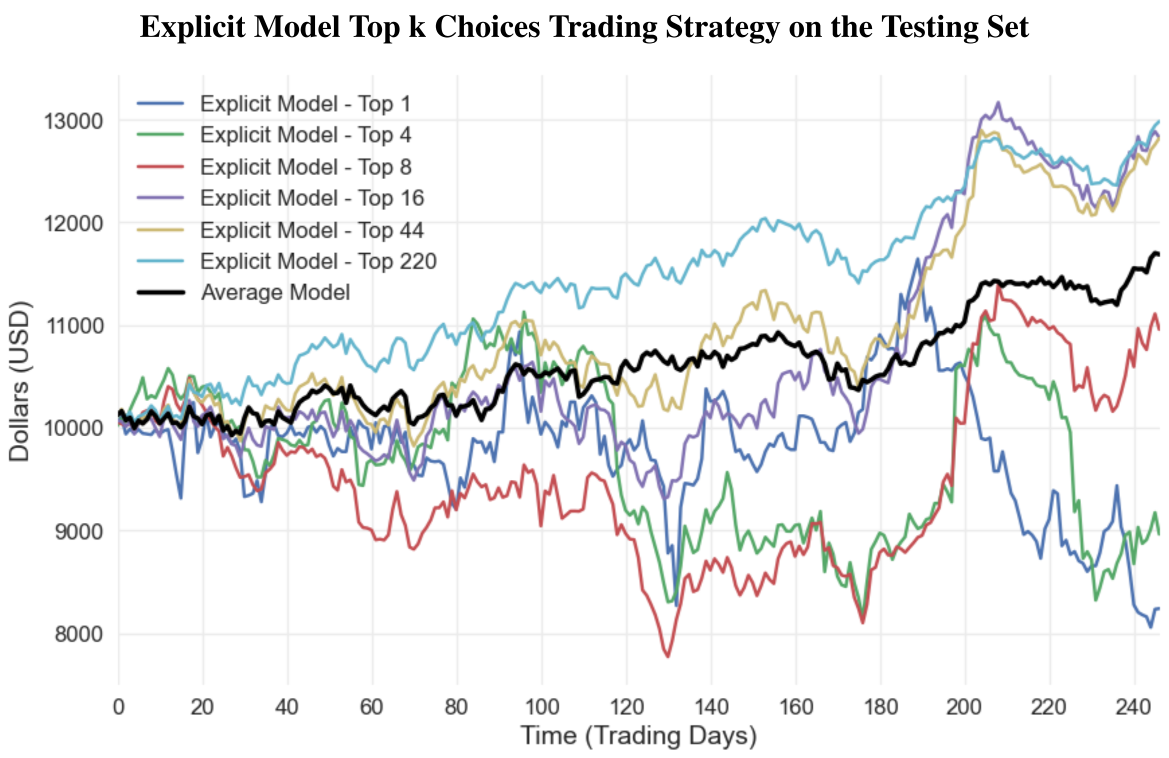

Figure 4.3: The daily portfolio value of explicit model strategies where the top k companies are selected

Table 4.2: Performance comparison between explicit model strategies where the top k companies are selected

Model	**MSE**	**IRR**	**MR**	**Diversity**
Average	-	0.1689	-	-
Explicit - Top 1	2.59e-02	-0.1761	218.02	0.80%
Explicit - Top 4	2.59e-02	-0.1037	249.27	1.91%
Explicit - Top 8	2.59e-02	0.0957	266.24	2.41%
Explicit - Top 16	2.59e-02	0.2826	281.61	2.66%
Explicit - Top 44	2.59e-02	0.2821	322.79	2.58%
Explicit - Top 220	2.59e-02	0.2984	443.22	1.24%

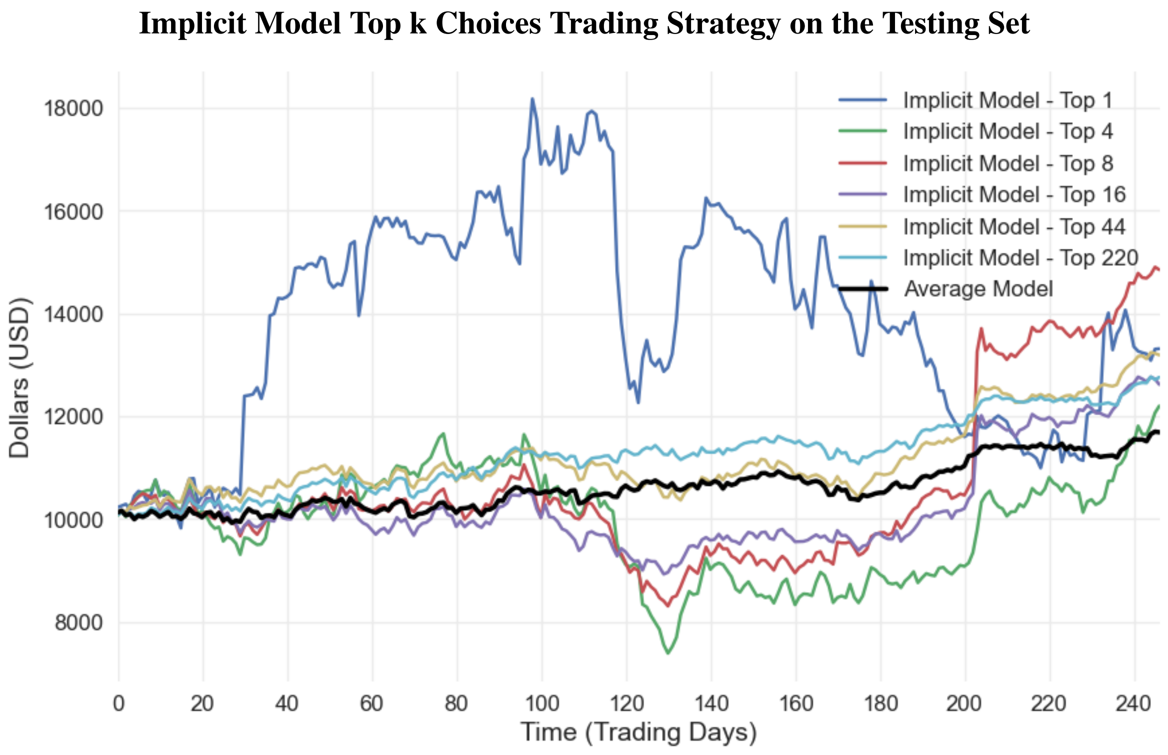

Figure 4.4: The daily portfolio value of implicit model strategies where the top k companies are selected

Table 4.3: Performance comparison between implicit model strategies where the top k companies are selected

Model	**MSE**	**IRR**	**MR**	**Diversity**
Average	-	0.1689	-	-
Implicit - Top 1	1.94e-03	0.3314	426.53	15.26%
Implicit - Top 4	1.94e-03	0.2199	419.98	14.16%
Implicit - Top 8	1.94e-03	0.4851	419.95	14.91%
Implicit - Top 16	1.94e-03	0.2619	419.83	11.72%
Implicit - Top 44	1.94e-03	0.3191	424.73	6.50%
Implicit - Top 220	1.94e-03	0.2758	480.16	1.60%

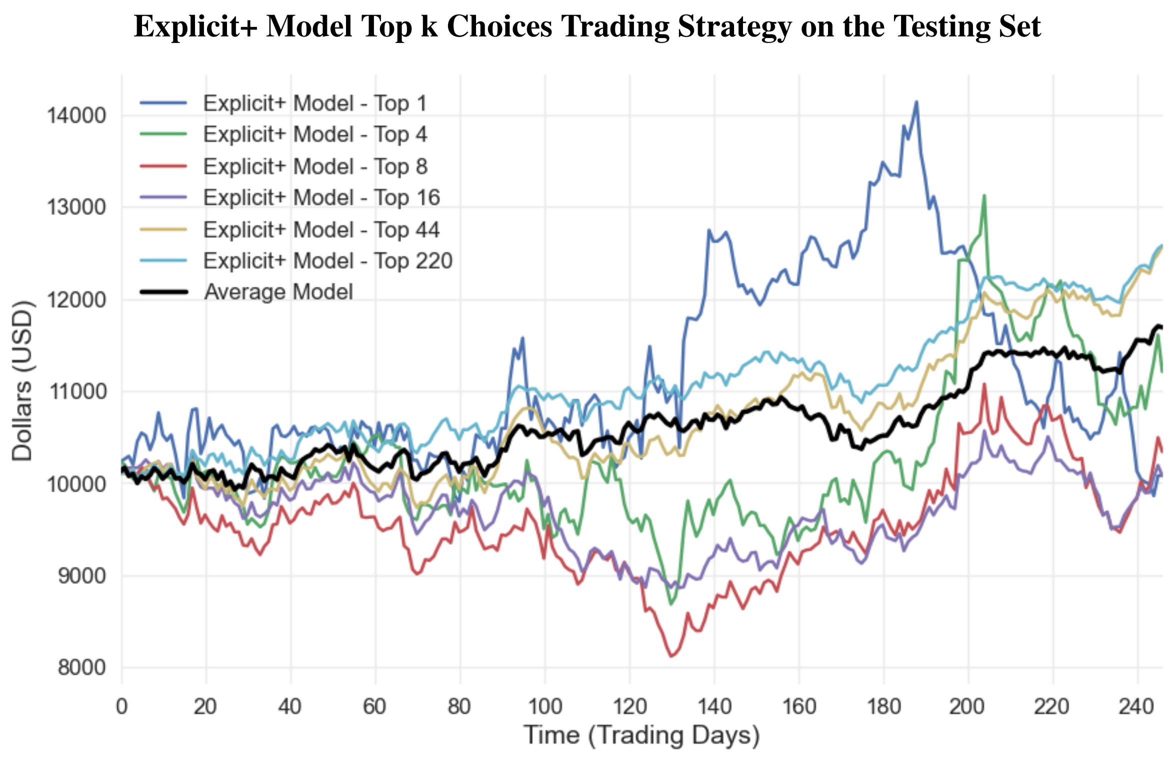

Figure 4.5: The daily portfolio value of explicit+ model strategies where the top k companies are selected

Table 4.4: Performance comparison between explicit+ model strategies where the top k companies are selected

Model	**MSE**	**IRR**	**MR**	**Diversity**
Average	-	0.1689	-	-
Explicit+ - Top 1	1.88e-02	0.0082	335.19	2.01%
Explicit+ - Top 4	1.88e-02	0.1207	351.49	3.51%
Explicit+ - Top 8	1.88e-02	0.0337	353.58	3.66%
Explicit+ - Top 16	1.88e-02	0.0075	357.59	3.01%
Explicit+ - Top 44	1.88e-02	0.2569	378.57	2.47%
Explicit+ - Top 220	1.88e-02	0.2578	460.89	1.35%

4.2.3 Similarity among Various Models' Investment Choices

Let T_{test} represent the number of time steps present in the testing set. At each time step, each model selects the highest-ranked choice based on its predictions of return ratios. Each selection is stored as an integer c (company index) where $c \in \{0, ..., (N-1)\}$. By counting the total occurrences when two models made the same decision on the same day, and then dividing by T_{test}, we calculate a percentage total of pairwise similarity between any pair of models. In Table 4.5 we show the pair-wise similarity between the between all $k = 1$ models trained up to this point.

Table 4.5: Similarity comparison between all the $k = 1$ models

	LSTM ($\alpha = 0$)	LSTM ($\alpha = 10$)	LSTM+GCN Explicit	LSTM+GCN Implicit	LSTM+GCN Explicit+
LSTM ($\alpha = 0$)	100%	53.04%	0%	21.86%	6.07%
LSTM ($\alpha = 10$)	53.04%	100%	2.83%	31.17%	6.47%
LSTM + GCN Explicit	0%	2.83%	100%	45.74%	89.47%
LSTM + GCN Implicit	21.86%	31.17%	45.74%	100%	55.06%
LSTM + GCN Explicit+	6.07%	6.48%	89.47%	55.06%	100%

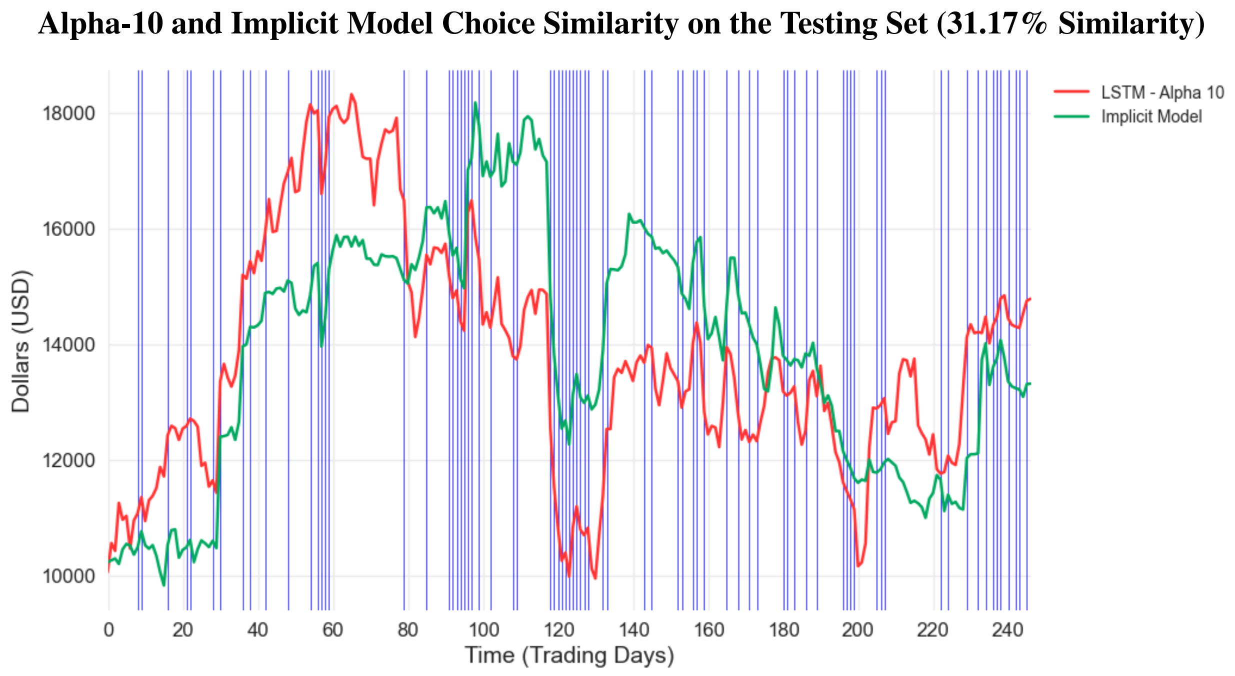

Figure 4.6: Alpha-10 and implicit model portfolios are shown, and the day is highlighted if models chose the same company on the same day

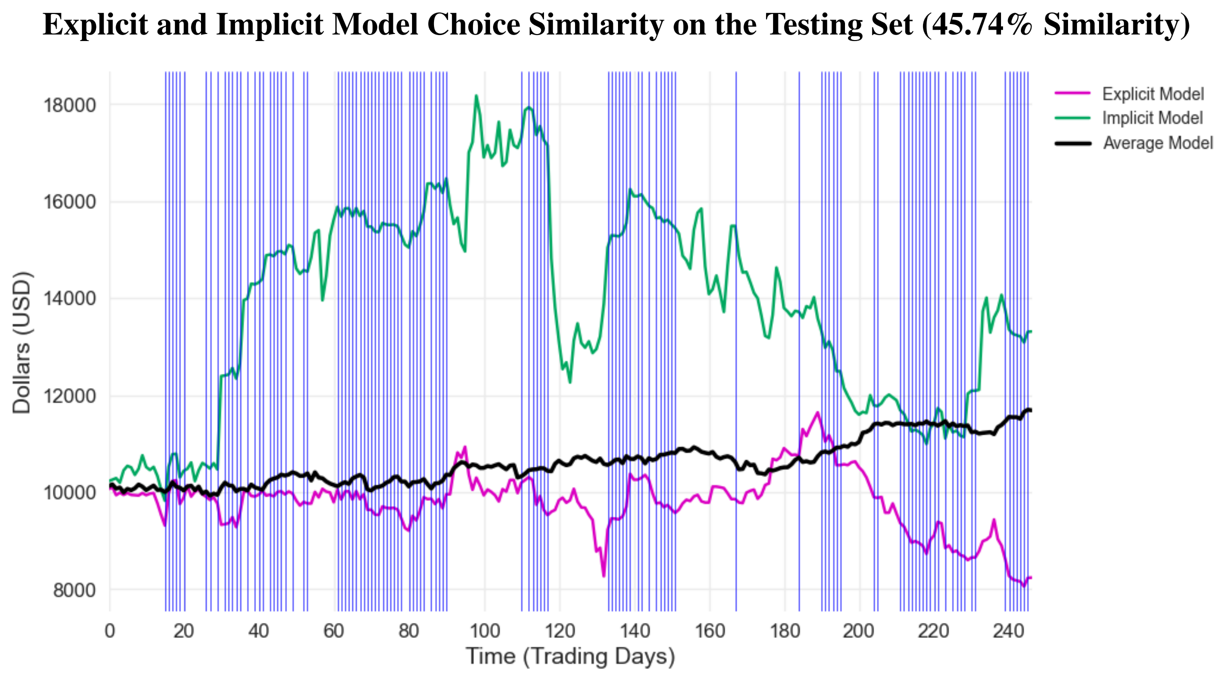

Figure 4.7: Implicit, and explicit model portfolios are shown, and the day is highlighted if models chose the same company on the same day

4.3 Conclusions

4.3.1 Implicit vs. Explicit Modeling

Looking at the results of the single-choice implicit and explicit models in Section 4.2.1, the implicit model is vastly superior on this testing set. Not only did it increase its portfolio value by 33.14%, but it was also able to do so with diverse company decisions, choosing 37 different companies over the trading year. The explicit and explicit+ models were worse than average, meaning they are not viable in a real financial market. Their diversity scores indicated that they only chose 2 companies and 5 companies respectively. In [5], selecting the highest ranked company on each time-step returned the highest IRR. However, given the low diversity score of the explicit models, but better mean-rank score, perhaps selecting the top k choices can improve performance.

4.3.2 Top k Stock Selection

4.3.2.1 Explicit Model

When looking at the top k selection strategies for the explicit model, the results drastically improve. The $k = 8$ model resulted in an IRR below average. But past $k = 8$, the IRR continued to improve with each successive increase in the number of stocks selected. For this data set, this implies that the similarity between two stocks is not optimal selecting the best possible stock on any given day. However, using the similarity to provide a general ranking was effective because the top 25% of selections performed better than the expected value.

4.3.2.2 Implicit Model

Given that the highest-ranked choice was better than average, it is not surprising to see that all implicit models performed better than average given that the success of the selected stocks in the $k = 1$ case was not an outlier. As can be seen in Figure 4.4, the $k = 220$ and $k = 44$ models demonstrate sustained growth rather than the volatile nature of $k = 1$. Either model choice might be preferred depending on the investor, state of the market, and individual risk tolerances.

The $k = 8$ choice for the implicit also model demonstrated the best performance out of all models tested on this data set with a portfolio increase of 48.51% after approximately one trading year. However, this result was only achieved after a significant price increase near day 200, which might be an outlier.

4.3.2.3 Explicit+ Model

While expanding our horizons to look at different k top choices for the explicit and implicit models revealed interesting insights into their performances, the explicit+ model appears to be worse than either predecessor. The addition of a learned parameter only hindered the original explicit model rather than improve it. The IRR was worse than average until $k = 44$, at which point the model improved similarly, but less effectively, compared to the implicit and explicit models.

4.3.3 Selected Company Similarity

For the models demonstrated so far, we've only compared their performance as functions of individual decisions. When looking at the portfolio values across time, most model choices visibly diverge. However, sometimes two models will make identical decisions later in the dataset because internal indicators for prediction are strong enough to pull both models to a certain choice. This is visibly apparent on day 130 of Figure 4.2, and then verified by Figure 4.7 where we've highlighted all time steps where both models invested in the same company.

We show the similarity percentage between models in Table 4.5 to further validate and assess their overall performances. It provides numerical proof that the consecutive upgrades of each model are not only providing an increase in return ratio but that they do so by learning new strategies that can avoid pitfalls of previous models while still relying on old successes. As can be seen in 4.6, even though the two portfolios are visually similar, their choices are 68.8% dissimilar, highlighting that visual interpretations of time-series data can be misleading without understanding the underlying systems.

Not only can we assess our models using their similarity, but we can also use their similarity to assess the qualities of our data. By analyzing which time steps are most similar or most dissimilar across drastically different models, we can gain insights about the outlier time steps. If a certain stock is always purchased on a certain time step across all temporal models but never purchased for temporal-relational models, we learn more about the nature of our data. If a poor investment choice is made across all models, as can be seen on day 118 of both Figure 4.6 and Figure 4.7, perhaps a deeper issue exists with the trading strategy or data set which cannot be solved by training or tuning models with the given data alone.

5. MODEL INTERPRETABILITY

While the ultimate goal of model performance is to select the highest-ranked stocks, treating a model as a black box is wasting the potential in interpreting and understanding underlying patterns of financial markets. In this chapter, we will cover several methods for better understanding and evaluating the internal mechanisms that govern our model decisions.

5.1 Temporal Feature Importance with Perturbation Analysis

To analyze the feature importance of our temporal models , we run our testing data $\mathcal{X}$ through the LSTM ($\alpha = 10$) model undisturbed to output a resultant $\mathcal{Y}$. Then we perturb one feature, f_i ($i = 1, \ldots, 5$), of the input data by adding a random normal variable $V \sim \mathcal{N}(0, 0.2)$ to every value in the feature column. The perturbed $\hat{\mathcal{X}}$ is run through the model to produce $\hat{\mathcal{Y}}$, which can be compared to the original $\mathcal{Y}$ using the MSE to determine what effect the perturbation had on the output. This process is repeated for each individual feature. Since each feature was perturbed with the same Gaussian distribution with a sufficiently large sample size, we can normalize the results to determine the relative strength that each feature has on the output prediction.

Table 5.1: Output perturbation values for individual features of the temporal data set

Feature	**Perturbation Effect**	**Perturbation Effect (Normalized)**
Closing Price	0.0823	0.468
5D-MA	0.0324	0.184
10D-MA	0.0238	0.135
20D-MA	0.0230	0.131
30D-MA	0.0142	0.081

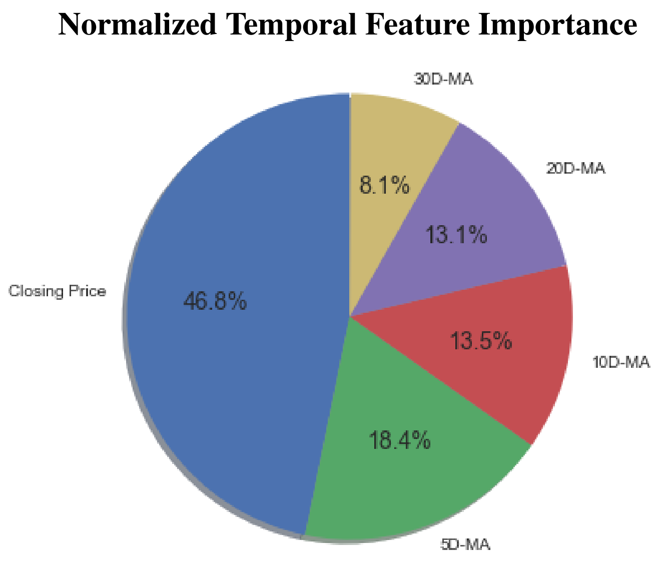

Figure 5.1: The strength of each feature on the temporal data set is shown

Unsurprisingly, the most important feature for the LSTM model was the closing price of each stock. This makes sense given that, on average, the previous day's closing price should be closer to the current day's actual price compared to the 5, 10, 20, and 30-day moving averages. This pattern is sequentially shown as well; as the number of days calculated within the MA increases, the feature importance decreases. While the value of the 30D-MA is pessimistically low for attempting to learn extremely long temporal patterns, it is motivating to know that over 50% of feature importance is not reliant on just the previous day's closing price.

Given that the feature importance decreases approximately by a factor of 2 with each addition of an MA, expanding the features on this data set to 50D-MA and 100D-MA might further improve model performance. Many other temporal stock indicators like volume, volatility, and momentum could also be added for improved performance or for interpretability about which predictors are best suited for different markets or strategies. Additionally, the perturbation method for extracting feature importance can be expanded into the graph domain for better understanding relational models as well.

5.2 Company Relation Importance with Heatmap Analysis

To interpret the mechanisms behind decisions made by a GCN, we start with a heatmap where the X and Y axes are symmetric and each tick is a company, and the strength between them is represented by a gradient color-scale. Shown below in Figure 5.2 are four random subsets of companies and their relationship pairs.

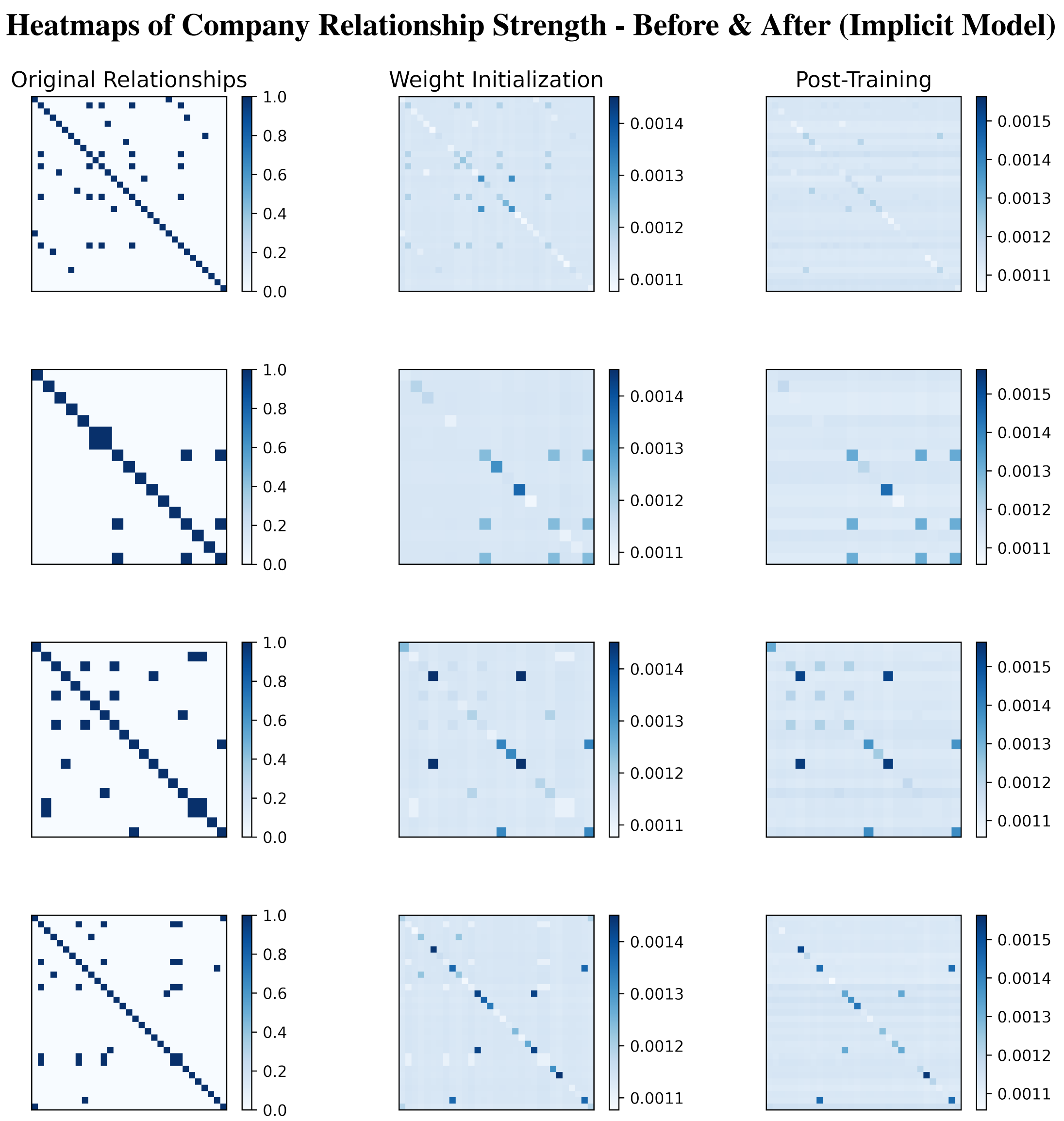

Figure 5.2: Each row shows input company relationship strengths, their weight initialization, and post-training relationship strengths for a random subset of companies

The first column visualizes the original relationships specified in our adjacency matrix, $\mathcal{A}$. The second column shows the internal adjacency matrix of the implicit model after one epoch of training. The third column shows the matrix after the last epoch of training. Each column's relationship strength scale is normalized by the minimum and maximum of the entire $N \times N$ adjacency matrix for that epoch.

As can be seen in the original relationships column, the diagonal of each matrix is 1; this represents the self-loop of the graph, specifying that every company shares a relationship with itself [17]. Intuitively, this makes sense because even though our objective is to utilize relational information, attempting to predict the closing price of a company while only relying on neighboring values seems unnecessarily difficult. However, by looking at the third column, we see that some companies learned not to purely rely on their own price history after training. Table 3.2 and Table 4.1 support this result since they show that temporal models have lower MSE than temporal-relational models. Instead of utilizing the self-loop, it appears as though the model has found a local minimum where some company predictions are the result of hundreds of small aggregations; this might be correlated with the higher error.

After training, the distribution total of strength is much more dispersed compared to the original adjacency matrix. Even if the contributions are small, almost every company relationship pair is aggregated with each other since the minimum value is non-zero. Interestingly, in Figure 5.2, we cannot find an occurrence where an initially zero strength relationship became darker than the surrounding aggregate noise. This signifies that our implementation of the learned relationships might be too restrictive and limiting the exploratory capabilities of machine learning models.

For future research, we will analyze the outlier relationships where the model decided to ignore the self-loop or was able to maintain a strong strength function. Companies without a self-loop might be too volatile to predict, whereas relationships that withstood hundreds of epochs of training could indicate two extremely correlated companies. Figure 5.2 only scratches the surface of possible interpretabilities for GCNs.

6. CONCLUSIONS AND FUTURE DIRECTIONS

6.1 Conclusions

In this thesis, we conducted numerical experiments demonstrating gradual improvements in time-series forecasting accuracy for the NASDAQ financial market. Starting with a state-of-the-art sequential model (LSTM) as a baseline, we showed that LSTM models can be further enhanced by considering entity ranking in the loss function beyond standard regression errors. The addition of a graph structure allowed the resulting LSTM+GCN models to aggregate time-series data of multiple entities together before making a prediction. This demonstrated similar and better than average accuracy compared to independent models but did so by choosing different stock choices: signifying that multiple successful strategies for investment are possible on a given data set.

We also provided multiple methods for comparing and understanding the output predictions of our models, such as selection similarity for all models and heatmap visualizations for relational models. We hope that any data scientist or finance-market stakeholder interested in the field of temporal, relational, or temporal-relational machine learning models can also utilize these methods for their own interpretability or assessment.

6.2 Future Directions

We note that this is only the tip of the iceberg for the potential of graphical neural networks in financial market predictions. The models were only applied to the NASDAQ stock exchange as a proof of concept, but can be applied to other exchanges. Additionally, less traditional markets like cryptocurrency [18] and trading card exchanges [19] have seen a surge in market capacity during the COVID-19 pandemic and unexplored strategies in these markets might also be uncovered.

The ultimate objective of our models was to maximize profits by purchasing a stock each day and selling it the next day with a fixed spending budget. This is a simple trading strategy that mimics rational actors in a system, but financial markets offer a variety of different trading strategies that can be researched further:

- *Variable Investments* - Instead of always investing the same amount of money each day, the model could be incentivized to learn a confidence interval for each investment. Time steps with large losses could be minimized while time steps with large gains could be exploited.
- *Buy and Hold* - We can incentivize longer investment positions by predicting how long a stock should be held instead of selling it after one time step. Expanding the time window in which a stock's value moves might generate models with less risk or assist in avoiding situations where multiple good options are neglected for the best option.
- *Short Selling* - Short selling allows investors to profit on a stock's value decreasing rather than increasing. Instead of selecting the best potential stock at any given time step, it might be more effective to select the worst potential stock and utilize short positions. Indicators for failure might be more trainable than indicators for success.

Another application of jointly modeling sequential and relational predictions in the finance market is reinforcement learning. Since the ultimate objective of financial prediction is earning the most return, it might be more effective to skip the regression step of predicting each stock's rank and directly incorporate the IRR into the loss function. GCN structures have already shown success in the reinforcement field [20], and applying those models to the financial market should produce completely new results since the problem is being approached from a different perspective.

The results of this thesis would not be possible without the combined efforts of many researchers and data scientists building its foundations. Anyone is welcome to reproduce the results in this thesis or utilize the models on another data set to expand the research. The availability information of our codes and data is provided in Section 2.1.

REFERENCES

[1] Bloomberg, “Global market capitalisation exceeds world gdp, evoking ’bubble’ concern,” *Business Standard*, Aug 2020. Available: https://www.business-standard.com/article/markets/global-market-capitalisation-exceeds-world-gdp-evoking-bubble-concern-120081100073_1.

[2] G. L. Musgrave, “Review: A random walk down wall street,” *Business Economics*, vol. 32, no. 2, pp. 74–75, 1997.

[3] A. Lo and A. MacKinlay, *A Non-Random Walk Down Wall Street*, pp. 3–11. Princeton University Press, 2011.

[4] E. Beyaz, F. Tekiner, X.-j. Zeng, and J. Keane, “Comparing technical and fundamental indicators in stock price forecasting,” in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1607–1613, IEEE, 2018.

[5] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, “Temporal relational ranking for stock prediction,” *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 2, pp. 1–30, 2019.

[6] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, “Stock market’s price movement prediction with lstm neural networks,” in *2017 International joint conference on neural networks (IJCNN)*, pp. 1419–1426, IEEE, 2017.

[7] D. Matsunaga, T. Suzumura, and T. Takahashi, “Exploring graph neural networks for stock market predictions with rolling window analysis,” *arXiv preprint arXiv:1909.10660*, 2019.

[8] R. Kim, C. H. So, M. Jeong, S. Lee, J. Kim, and J. Kang, “Hats: A hierarchical graph attention network for stock movement prediction,” *arXiv preprint arXiv:1908.07999*, 2019.

[9] S. Bhanja and A. Das, “Impact of data normalization on deep neural network for time series forecasting,” *arXiv preprint arXiv:1812.05519*, 2018.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[12] C. Olah, “Understanding lstm networks,” 2015. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs.

[13] Z. Zheng, K. Chen, G. Sun, and H. Zha, “A regression framework for learning ranking functions using relative relevance judgments,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 287–294, 2007.

[14] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[15] I. Kandel and M. Castelli, “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,” *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.

[16] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020.

[17] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[18] B. Bambrough, “Bitcoin’s next boom has already begun,” *Forbes*, Apr 2020. Available: https://www.forbes.com/sites/billybambrough/2020/04/10/bitcoins-next-boom-has-already-begun/?sh=15db19885c64.

[19] T. Beer, “Ebay reports increase of 4 million trading cards sold in 2020,” *Forbes*, Feb 2021. Available: https://www.forbes.com/sites/tommybeer/2021/02/11/ebay-reports-increase-of-4-million-trading-cards-sold-in-2020/?sh=4d2d34f91963.

[20] J. Jiang, C. Dun, T. Huang, and Z. Lu, “Graph convolutional reinforcement learning,” *arXiv preprint arXiv:1810.09202*, 2018.