5-31-2022

Representation learning in finance

Ajim Uddin
New Jersey Institute of Technology, au76@njit.edu

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

REPRESENTATION LEARNING IN FINANCE

by
Ajim Uddin

Finance studies often employ heterogeneous datasets from different sources with different structures and frequencies. Some data are noisy, sparse, and unbalanced with missing values; some are unstructured, containing text or networks. Traditional techniques often struggle to combine and effectively extract information from these datasets. This work explores representation learning as a proven machine learning technique in learning informative embedding from complex, noisy, and dynamic financial data. This dissertation proposes novel factorization algorithms and network modeling techniques to learn the local and global representation of data in two specific financial applications: analysts' earnings forecasts and asset pricing.

Financial analysts' earnings forecast is one of the most critical inputs for security valuation and investment decisions. However, it is challenging to fully utilize this type of data due to the missing values. This work proposes one matrix-based algorithm, "Coupled Matrix Factorization," and one tensor-based algorithm, "Nonlinear Tensor Coupling and Completion Framework," to impute missing values in analysts' earnings forecasts and then use the imputed data to predict firms' future earnings. Experimental analysis shows that missing value imputation and representation learning by coupled matrix/tensor factorization from the observed entries improve the accuracy of firm earnings prediction. The results confirm that representing financial time-series in their natural third-order tensor form improves the latent representation of the data. It learns high-quality embedding by overcoming information loss of flattening data in spatial or temporal dimensions.

Traditional asset pricing models focus on linear relationships among asset pricing factors and often ignore nonlinear interaction among firms and factors. This

dissertation formulates novel methods to identify nonlinear asset pricing factors and develops asset pricing models that capture global and local properties of data. First, this work proposes an artificial neural network "autoencoder" based model to capture the latent asset pricing factors from the global representation of an equity index. It also shows that autoencoder effectively identifies communal and non-communal assets in an index to facilitate portfolio optimization. Second, the global representation is augmented by propagating information from local communities, where the network determines the strength of this information propagation. Based on the Laplacian spectrum of the equity market network, a network factor "Z-score" is proposed to facilitate pertinent information propagation and capture dynamic changes in network structures. Finally, a "Dynamic Graph Learning Framework for Asset Pricing" is proposed to combine both global and local representations of data into one end-to-end asset pricing model. Using graph attention mechanism and information diffusion function, the proposed model learns new connections for implicit networks and refines connections of explicit networks. Experimental analysis shows that the proposed model incorporates information from negative and positive connections, captures the network evolution of the equity market over time, and outperforms other state-of-the-art asset pricing and predictive machine learning models in stock return prediction.

In a broader context, this is a pioneering work in FinTech, particularly in understanding complex financial market structures and developing explainable artificial intelligence models for finance applications. This work effectively demonstrates the application of machine learning to model financial networks, capture nonlinear interactions on data, and provide investors with powerful data-driven techniques for informed decision-making.

REPRESENTATION LEARNING IN FINANCE

by
Ajim Uddin

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Business Data Science

Martin Tuchman School of Management, NJIT

May 2022

APPROVAL PAGE

REPRESENTATION LEARNING IN FINANCE

Ajim Uddin

Dr. Dantong Yu, Dissertation Advisor — Date
Associate Professor of Business Data Science, NJIT

Dr. David A. Bader, Committee Member — Date
Distinguished Professor of Data Science, NJIT

Dr. Xinyuan Tao, Committee Member — Date
Assistant Professor of Finance, NJIT

Dr. Zhi Wei, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Hui Shao, Committee Member — Date
Director, Quantitative Finance Manager, Bank of America Merrill Lynch, New York, NY

BIOGRAPHICAL SKETCH

Author: Ajim Uddin

Degree: Doctor of Philosophy

Date: May 2022

Undergraduate and Graduate Education:

- Doctor of Philosophy in Business Data Science,
  New Jersey Institute of Technology, Newark, NJ, 2022
- Master of Science in Economics and Finance,
  Southern Illinois University Edwardsville, Edwardsville, IL, 2017
- Master of Business Administration,
  Shahjalal University of Science & Technology, Sylhet, Bangladesh, 2014
- Bachelor of Business Administration,
  Shahjalal University of Science & Technology, Sylhet, Bangladesh, 2013

Major: Business Data Science

Presentations and Publications:

A. Uddin, X. Tao, C.-C. Chou, and D. Yu, "Are missing values important for earnings forecasts? A machine learning perspective," *Quantitative Finance*, pp. 1–20, 2022.

M. Fang, S. Taylor, and A. Uddin, "The network structure of overnight index swap rates," *Finance Research Letters*, vol. 46, p. 102425, 2021.

A. Uddin, X. Tao, and D. Yu, "Attention based dynamic graph learning framework for asset pricing," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1844–1853.

M. A. F. Chowdhury, M. S. Meo, A. Uddin, and M. M. Haque, "Asymmetric effect of energy price on commodity price: New evidence from NARDL and time frequency wavelet approaches," *Energy*, vol. 231, p. 120934, 2021.

A. Uddin, M. A. F. Chowdhury, S. D. Sajib, and M. Masih, "Revisiting the impact of institutional quality on post-gfc bank risk-taking: Evidence from emerging countries," *Emerging Markets Review*, vol. 42, p. 100659, 2020.

A. Uddin and D. Yu, "Latent factor model for asset pricing," *Journal of Behavioral and Experimental Finance*, vol. 27, p. 100353, 2020.

A. Uddin, X. Tao, C.-C. Chou, and D. Yu, "Nonlinear tensor completion using domain knowledge: An application in analysts' earnings forecast," in *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2020, pp. 377–384.

M. E. S. M. Rasid, A. Uddin, and M. A. F. Chowdhury, "Islamic corporate finance: Capital structure," in *Islamic Corporate Finance*. New York, NY: Routledge, 2019, pp. 54–73.

A. Uddin, M. A. F. Chowdhury, and M. N. Islam,"Determinants of financial inclusion in Bangladesh: Dynamic GMM & quantile regression approach," *The Journal of Developing Areas*, vol. 51, no. 2, pp. 221–237, 2017.

A. Uddin, M. A. F. Chowdhury, and M. N. Islam, "Resiliency between islamic and conventional banks in Bangladesh: Dynamic GMM and quantile regression approaches," *International Journal of Islamic and Middle Eastern Finance and Management*, vol. 10, pp. 400-418, 2017.

A. Uddin, X. Tao, and D. Yu, "Network Centrality, Leadership, and Institutional Investors Portfolio Performance," *Northeast Decision Sciences Institute Annual Conference*, Newark, NJ, 2022.

A. Uddin, "The network factor of equity pricing: A signed graph Laplacian approach," *Northeast Decision Sciences Institute Annual Conference*, Newark, NJ, 2022.

A. Uddin, "Centrality, leadership, and information asymmetry: A bipartite network model for institutional investors investment performance," *Financial Management Association Annual Meeting (FMA)*, Denver, Colorado, 2021.

A. Uddin, "The network factor of equity pricing: A signed graph Laplacian approach," *Financial Management Association Annual Meeting (FMA)*, Denver, Colorado, 2021.

A. Uddin, X. Tao, and D. Yu, "Attention based dynamic graph learning framework for asset pricing," *37th International Conference of the French Finance Association (AFFI)*, Nantes, France, 2021.

A. Uddin, "A graph Laplacian approach for change point detection in the U.S. equity market network," *37th International Conference of the French Finance Association (AFFI)*, Nantes, France, 2021.

To my loving parents, Tara Miah and Husewara Begum,
and my caring brother, Mohammed Jafor Uddin

ACKNOWLEDGMENT

I would like to give my heartfelt thanks to my advisor, Dr. Dantong Yu, for his academic supervision, remarkable mentoring, and unconditional support through each stage of my doctoral study. His immense knowledge, motivation, and experience have guided and shaped my ability as a researcher and will influence me for the rest of my life.

I also want to express my gratitude to all my committee members: Dr. Xinyuan Tao, Dr. David A. Bader, Dr. Zhi Wei, and Dr. Hui Shao. I am very grateful for their time, suggestions, and kind guidance throughout these years. Their expert comments improve the quality of my research significantly.

I appreciate the financial support I received from my beloved school, Martin Tuchman School of Management. My sincere gratitude to the Dean Dr. Oya Tukel, for sponsoring my conference expenses and journal submission fees. I also appreciate the research funding from the National Institutes of Health (NIH) Award (Grant #UL1TR003017) and Leir Research Institute for Business, Technology, and Society (LRI) for "Leir Fellowship - Summer 2021".

I would like to thank my PhD labmates Shibo Yao, Dan Zhou, and Uras Varolgunes for their insightful comments and suggestions during our group meetings, which helped me to refine my ideas and improve my research productivity.

Finally, my heartfelt thanks to my family and friends for their unconditional love, support, and inspiration. They helped me to focus when things were difficult. They believed in me when I even doubted myself. Without their motivation and emotional support, this achievement would not be possible.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF TABLES
(Continued)

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

"Big data will become a key basis of competition, underpinning new waves of productivity growth, innovation, and consumer surplus—as long as the right policies and enablers are in place." – Mckinsey. The amount of data has exploded in the last decade around the globe. The fundamental question is how to analyze the data efficiently. The data structure can be complicated. Some are too sparse and incomplete; some are noisy; and some are not even numerical, but in the forms of text or graphs. Machine learning (ML) technologies pave the way for dealing with big and unstructured data, solving data inconsistency problems, and learning proper representation to make accurate predictions. Although most of these ML techniques are initially proposed to solve a specific problem, their applications are never constrained to the original domain, e.g., convolutional neural network (CNN) was initially proposed for image prediction [1], over the years it has been proved also effective for natural language processing [2,3], health care [4], game playing [5,6], and time series forecasting [7–9].

The advancement of machine learning-based technologies also brings a seismic shift in the business processes of financial industries. Rapid technology integration, the evolution of sharing economy, fierce competition, and ever-increasing customer expectation bring this industry to a tipping point to embrace revolutionary changes in business practice. Technological innovations and the application of machine learning create opportunities for financial institutions to develop and grow and, in the meanwhile, increase risks and make them susceptible to failure. Digital banking and mobile payment system eliminate the need for cash and neighborhood bank branches. Automated trading and app-based brokerage service, e.g., Robinhood, created a new

domain for financial service while making some traditional services obsolete. Today more than 75% of the stocks traded in the United States (U.S.) exchanges originate from automated trading systems orders that take various names, including mechanical trading, algorithmic trading, etc. The emergence of automatic trading results from the exceptional capability of ML techniques, i.e., high-frequency trading by detecting profitable trading opportunities at the tick level.

The scope and application of ML in finance are broad and diverse. In this dissertation, I focus on applying a specific aspect of ML – *Representation Learning* – in finance. Representation learning attempts to extract useful representation from the data and use the extracted representation for downstream classification or prediction [10]. The success of machine learning techniques generally depends on extracting the proper representation from the data. Different representations capture different aspects of the data, resulting in different downstream classification/prediction performance. Traditionally, feature engineering has been used to ensure proper data representation for machine learning algorithms. Nevertheless, feature engineering relies on hand-craft techniques. Consequently, it is inflexible for new application domains and often not scalable for big data. Recent breakthroughs in machine leanings make it possible to design algorithms that automatically learn proper representation from data without depending heavily on feature engineering. Machine learning techniques learn effective representations from data by capturing the posterior distribution of hidden explanatory factors of the input data. Principal component analysis (PCA), independent component analysis (ICA), matrix factorization (MF), linear discriminant analysis (LDA), manifold learning, autoencoders, deep belief networks, and CNN are the examples of representation learning and have been successfully applied in computer vision [11–13], health care [14–16], voice recognition [17], natural language processing [18–20], and sentiment analysis [21, 22].

All these techniques essentially try to extract key information from the input data and transform them into lower-dimensional simplified embedding.

Similar to many types of real-world data, financial data are often complex, noisy, high-frequency, big in volume, unstructured, and unbalanced. These representation learning techniques are proven to reduce dimensionality, mitigate complexity, and used successfully to extract valuable information from big-noisy financial data. In this work, I try to solve two specific problems in finance through representation learning: (i) representation learning for firm earnings prediction, and (ii) representation learning for asset pricing.

1.1 Representation Learning for Firm Earnings Prediction

Earnings per share (EPS) is the ratio of a firm's earnings to its number of common shares outstanding. It conveys vital information about a firm's future performance and is one of the fundamental inputs for security pricing [23]. Investors, regardless of investment banks, fund managers, and individuals, all rely on the market expectation of a firm's EPS in the future, i.e., the next quarter EPS, to make investment decisions: buying (selling) the stock of a firm if its future EPS is projected to be high (low).

Analysts' consensus (mean or median) forecast of EPS is considered as the common and plausibly dominant measure of market expectation.[1] Unlike institutional investors (e.g., mutual funds, pension funds, and insurance companies), most individual investors do not have the necessary skills, information, and time to conduct their own analysis. They simply make their investment decisions based on financial analysts' forecast of EPS that is available to the public. In addition to individual investors, institutional investors also use analysts' forecast as the benchmark to gauge their pricing models or as the complementary predictor variables to their models. Moreover, firm managers might also pay attention to analysts'

[1]Following industry practice, the "consensus forecast" is defined as the average of all the analysts forecasts for a firm at a given quarter and referred to as "mean forecast".

forecast to learn the market expectation to their firms and adjust their business operations accordingly. Financial analysts apply their domain knowledge to generate their earnings forecast using large sets of information, such as proxy statements, published financial reports, conference calls, management communications, behavioral assumptions, and macro-economic conditions [24–27]. Therefore, analysts' earnings forecast has a superior value in finance as it has human-in-the-loop for combing through a variety of data sets. Early studies in both finance and accounting show that analysts' earnings forecast is a better estimator of firm earnings than time series models due to their information and timing advantage [23, 28–30].

The analysts' forecasts of EPS data nevertheless holds several challenges to perform traditional regression-based analysis. First, analysts' forecast of EPS is not balanced. At a given time, multiple analysts follow one firm and generate individual reports. Analysts only track a limited set of firms and generate reports for only these firms while skipping all other firms. Even for the limited set of firms, analysts occasionally skip reporting at some time breaks and change the firms that they follow. As a result, the EPS forecast dataset is highly sparse and has a high number of missing values in the dataset, i.e., over 99%. Second, the EPS announcement only comes quarterly along with individual forecast of hundreds of analysts, resulting in more predictors (each analyst generates one predictor) than observations. Third, the EPS dataset has three dimensions, time, firm, and analysts. The common time-series practice is to aggregate information from individual analysts based on average, personal judgment or analysts' past performance and flatten the third-order tensor into a *firm-time* matrix to ease the data complexity [31–33]. Such a practice only provides information from time-domain while neglecting important information from the spatial-domain that encapsulates analyst correlation and implicit interactions between analyst and firms [34, 35]. In this case, it ignores the useful information of an individual analyst who follows multiple firms concurrently and firms' commonality

within the same group. In addition, literature also suggests that analysts vary in their forecasting accuracy [36, 37] and may show a systematic bias [37, 38]. Aggregating all the efficient and inefficient analysts forecast indiscriminately will contaminate the information content of efficient analysts' forecast.

To overcome these three challenges, this dissertation investigates the scope of representation learning on analyst earnings forecast data and proposes novel matrix and tensor techniques for data integration and completion. The first technique is Coupled Matrix Factorization (CMF), and the second technique is neural network enabled Nonlinear Tensor Coupling and Completion Framework (NLTCC). Matrix (tensor) factorization identifies the low-rank representation of matrix (tensor) from the observed entries and then reconstructs the complete matrix (tensor) from the low-rank representation. The goal is to learn a better representation from available sparse data to impute missing values while preserving the data properties, patterns, structure, and distribution, and then using this imputed data to build a better predictive model for firms' future earnings.

In the analyst forecast data, the percentage of missing value is very high, and a straightforward application of matrix or tensor factorization may not always be enough to learn high-quality embedding. To overcome this, I supplement the analysts' EPS forecast with two additional data sources: firm characteristics and daily return data. The firm characteristics data provide fundamental information of a firm and have much fewer missing values. Return data provide up-to-date market performance and expectations. Studies also show that firm characteristics and market expectations affect analysts' forecast accuracy [39–42].

The proposed CMF supplements the imputation of the sparse analysts forecast with firm characteristics data. By fusing two data sets with CMF, I incorporate their comprehensive information into the imputation process. CMF uses customized loss functions and regularization concerning specific finance applications and selectively

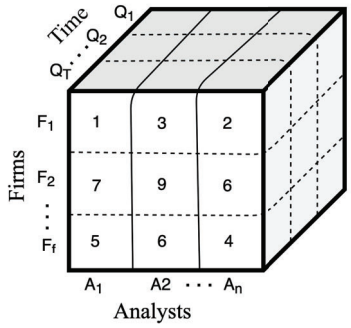
incorporates information from two datasets to learn a better representation. Many existing approaches use the common alternating least-squares minimization (ALS) approach that incurs significant computation and memory expenses and fails to scale to large datasets. Instead, I design a novel stochastic gradient descent (SGD) based CMF and attain significant improvements over the standard (coupled) matrix factorizations in terms of accuracy, simplicity for implementation, and computing speed for extremely large datasets [43].

To examine the quality and usefulness of the learned representation, I apply advanced ML methods to the complete matrix to predict firms' next quarter earnings once the data is imputed. Those ML methods include Least Absolute Shrinkage and Selection Operator (LASSO) [44], eXtreme Gradient Boosting (XGBoost) [45], and Support Vector Regression (SVR) [46]. I show that all these ML methods predict better with imputed data. The performance of XGBoost is the best, slightly better than that of SVR, whereas both are much better than the consensus (mean) forecast. More interestingly, I find that the most advanced ML method, XGBoost, cannot even beat the consensus forecast without the help of our data imputation method. The finding suggests the necessity of imputing missing values before applying advanced ML methods.

Although CMF provides a useful starting point for understanding the importance of imputing missing value in predicting firm earning forecast accurately, it has some limitations. It cannot overcome the third challenge - efficiently capturing Spatio-temporal dependency from three-dimensional data. To apply matrix factorization, we either need to apply it to firms individually or flatten the tensor along any two dimensions to convert it to a matrix. Analyzing individual firms cannot capture intra-company information and lose important information about the market, industry, and economic condition. On the other hand, flattening the

Quarter	Firms	A_1	A_2	A_n
Q_1	F_1	2	2	3
	F_2	7	6	9
	F_f	5	4	5
Q_2	F_1	3	2	3
	F_2	9	8	9
	F_f	4	4	3
Q_T	F_1	1	3	2
	F_2	7	9	6
	F_f	5	6	4

(a) Matrix



(b) Tensor

Figure 1.1 Representing financial analysts earning forecast data in matrix and in tensor format.

third-order tensor into a *firm-time* matrix loses the important information from the spatial-domain [34, 35].

To overcome this problem, I propose a nonlinear tensor decomposition-based model NLTCC. By projecting analyst EPS forecast in *time*, *firms* and *analyst* dimension, we can efficiently represent panel data as a third-order tensor (as shown in Figure 1.1). The tensor representation captures information from both the spatial and temporal domains and thereby, overcome the panel data problem of missing critical information. The propose NLTCC is a convolutional neural network-based nonlinear tensor coupling and completion framework for heterogeneous data. It performs an intelligent data fusion process by leveraging the synergistic knowledge embedded in three financial data sets, firm characteristics, daily returns, and analysts' EPS forecasts and couples them into one multi-dimensional tensor. The approach uses latent information from the time and firm dimension of the complete return and characteristics data sets to impute missing values in sparse forecast data. Once the data is imputed, I use machine learning techniques, including simple linear regression, SVR, and XGBoost, to build a predictive model for firms' future earnings.

The experimental analysis shows that the tensor-based approach is superior to the traditional approaches including consensus forecast and CMF. In addition to

imputing analysts' missing forecasts, NLTCC can also distinguish influential analysts from ineffective ones and excel even more at the sectors with high heterogeneity and volatility. The portfolio analysis shows that NLTCC based prediction is better at identifying good (bad) performing stocks and offers more profitable investment opportunities.

1.2 Representation Learning for Asset Pricing

Asset pricing is a fundamental problem of predicting assets risk premia.[2] Because risk premia are the conditional expectation of assets future realized excess return, I define risk premia as the task of predicting an appropriate value or a 'price' for a financial asset [47]. Traditional asset pricing models, e.g., capital asset pricing model (CAPM), arbitrage pricing theory (APT), and Fama-French factor models have important implication in asset pricing literature. These models help us develop the initial understanding of risk factors and risk premia. Factor models have been used as the primary yardstick for explaining the assets risk premia over the years [48–51]. They explain a large portion of return variability on assets. Nevertheless, they have some serious limitations. CAPM is too simple and assumes a mean-variance efficient market portfolio. On the other hand, APT fails to specify the factors. In theory, an infinite number of factors can affect the risk premium of an asset, but it is nearly impossible to identify the most significant factors upfront [52, 53].

After the inception of APT [53] in 1976, researchers have proposed over a hundred factors based on empirical evidence, market influence, personal belief, and specific research needs. For building a perfect model, one brute-force solution may be to include the highest number of factors possible to maximize the model fit. This

[2]The risk premium is the conditional expected return (excessive return) over the risk-free rate. Although there are some minor technical differences, I use the term "risk premium" and "expected return" interchangeably in this dissertation.

practice of including a high number of factors without proper justification has several drawbacks that we will detail in the subsequent paragraphs.

Traditional predictors are highly correlated. For example, in a standalone analysis, both GDP and GNP of a country will significantly impact individual asset returns. However, including both factors will introduce multicollinearity in the model, as one can be expressed with a linear combination of another. The same is true for including the total assets and total liabilities plus owner equity as predictors of the model. The inclusion of these highly correlated factors introduce variance and significantly impacts the model performance because the beta of the risk factor will behave erratically for small changes in the data or the model. Besides, for each additional predictor, the model will lose one degree of freedom [54]. Including a high number of predictors with a small number of observations can significantly diminish a model's degree of freedom, resulting in an overfitted model. In such a case, a perfect in-sample model will perform poorly for out-of-sample data. That is also true for correlated predictors in nonlinear form. As shown in [47], a large number of factors reduce the out-of-sample R^2 to be negative.

Traditional asset pricing models assume a linear relationship between the factors and returns. Nevertheless, the assumption of linearity rarely holds in real-world financial data and ignores time-variance volatility. When compounded with the frequent factor selection problem, this assumption unavoidably limits linear models' applicability in real-world data. Studies proved that incorporating nonlinear elements can significantly improve the explanatory power of asset pricing models [47, 55].

The third challenge in asset pricing is that firms do not operate independently in the marketplace. They influence each other through multiple channels, including but not limited to their supply chain networks, board of directors, fundamentals similarity, industry sector, and market condition. As a result, the performance of a firm depends on not only its own operation but also other relevant firms, i.e., the interconnection

among firms affects each other and their market prices. This interconnectedness among firms through contemporaneous links forms the firms' network. Previous studies suggest that the structure, properties, and dynamism of a network provide important insights into how information flows and shocks are transmitted across firms and thereby affect their stock prices [56–60]. There is a general consensus about the importance of network in asset pricing. Meanwhile, most traditional asset pricing models mainly focus on firm-specific and market/macro factors, overlooking the indispensable interconnection among firms.

In this dissertation, I used advanced machine learning techniques to tame the large factor universe and incorporate network information into asset pricing models. Machine learning facilitates data-driven models for learning proper representation from hundreds of input features and turns into simple informative embedding that can easily be used for the downstream regression model. The graph algorithm and network modeling techniques allow learning efficient embedding (representation) for the equity market network and improve the prediction accuracy in asset returns. In addition, nonlinear activation functions helps capture any nonlinear interaction among the features and cross-section of firms in the high dimensional latent space.

First, I propose an "autoencoder" based representation learning method for learning latent factors for asset pricing. Autoencoder takes daily returns for both input and output and generates a compressed representation for the daily returns of all stocks in an index. The compressed representation stores both linear and nonlinear correlations, reduce dimensionality and provides a feature map best resembling the original data [61]. I use these latent factors generated from the autoencoder to identify communal assets (representative stocks) and non-communal assets (non-representative) in an index. The overarching idea is that the return of more representative stocks can be easily explained by the other stocks in the same index and demonstrates less return volatility. On the other hand, the return of

non-representative stocks of an index will be difficult to explain by the other stocks in the index and therefore are more likely to have high return volatility.

Extensive analysis of all the stocks in three major indices, i.e., S&P-500, Russel-3000, and NASDAQ-100, reveal that the latent factors identified using autoencoder efficiently explain the return difference between the high communal assets and low communal assets. On average, non-communal stocks earn 0.05% higher than the communal ones. However, the risk associated with these non-communal stocks is also 0.8% higher than communal stocks. Combined with the Fama-French factor model, the daily average return difference between small-high-non-communal stocks and big-low-communal stocks is at 0.10%.

Second, I develop a novel network factor based on the Laplacian spectrum of the U.S. equity market. The network factor incorporates both positive and negative connections of firms in the equity market and provide a vector representation of the market. The encoded representation of the network structure also incorporates the changing market condition, economic environment, and uncertainties in a macro context. Moreover, to capture the dynamic evolution and detect change points in networks, I construct the network factor "Z-score" by measuring the difference between the current network state and network states of previous months. I evaluate the importance and implication of the network factor in equity pricing by examining the factor in relation to the market, macro, and other asset pricing factors. The empirical results confirm that the network factor Z-score aligns well with significant market events and reveals information that is not captured by traditional asset pricing models. The Z-score has a significant negative risk premium and can be incorporated into conventional asset pricing models to reduce asset pricing anomalies.

Finally, I propose a novel two-step graph learning model to capture the dynamic nonlinear interconnections among firms and investigate their contributions to the stock price movement. In the first step, I use an attention function to learn

network connections among firms at each time point. In the second step, I model spatial-temporal relationships among firms by combining graph convolutional neural network and recurrent neural network to predict future returns. The architecture of the model is inspired by two advanced machine learning techniques for graph learning and panel data modeling. The first one is the attention mechanism. The attention function learns the interaction (inter-dependency) coefficient between each firm and its neighbors in a pre-selected network at each time point based on the observed features, historical return, and return correlation. As a result, the learned network is dynamic and captures multiple aspects of firm connectedness, including their explicit and implicit interactions. Attention function also denoises the network by removing less important connections. The second is the diffusion mechanism to model Spatio-temporal information. It has roots in physics and is extensively used in financial derivatives to model the stochastic process in asset pricing, interests rates, and bond pricing. Diffusion processing in the continuous space can be extended to the discrete space (graph) in the format of random walk in networks [62] and inspires a series of graph neural networks: Graph diffusion convolution (GDC) that incorporates diffusion mechanism to improve graph learning [63], node classification [64], and traffic prediction [65, 66]. In this work, I follow the recurrent convolutional diffusion process proposed in [66] to propagate firms' information to their neighbors. In this method, we apply the convolutional operation on firm features to model spatial dependency among each firm and the Gated Recurrent Units (GRU) [67] framework to capture temporal patterns in each time series. In addition, the convolutional diffusion function facilitates the model to incorporate firms' characteristics as node features into the learning process.

By combining the attention and diffusion mechanism, my model discovers multiple interesting findings. The proposed graph learning model is superior in prediction accuracy compared to the traditional asset pricing methods and other

off-the-shelf machine learning models. I design an intelligent mechanism to learn positive and negative networks separately. This learning mechanism shows that the networks learned from the model are meaningful and capture the dynamic equity market movement over time.

The network representation learning in asset pricing is related to three streams of literature. The first one is the network study in computer science and information system. A number of seminal works on graph and network representation learning initially appear in computer science and are sequentially adopted by other domains. These efforts include network spectral analysis, node embedding, node classification, edge embedding, edge dynamics modeling, and their applications in various domains, including image processing, protein-protein interactions, and social network analysis [68–70]. The advancement in machine learning and deep learning paves the way for developing neural networks on large-scale complex graphs [71]. Specially, a surge of spectral-based graph neural network follows the seminal work in [72]. These Graph Convolutional Network models (GNN) focus primarily on static networks with predefined topologies and are mainly used for node classification and link prediction [73]. A detailed review of the literature is referred to [74]. The success in network classification (predicting categorical values) motivates researchers to apply graph neural networks in regression problems (predicting continuous values). In recent years, several spatiotemporal models enhanced by graph neural networks attain impressive results in traffic prediction [65, 75, 76], ride-hailing demand forecasting [77, 78], and COVID-19 trend prediction [79]. Motivated by these successes, I apply graph neural networks to forecast asset returns in my work.

The second stream is related to the network representation of firms. A handful of recent finance research attempts to understand the network dynamics of financial assets. These efforts include developing graph representation for financial market [59, 60, 80] and modeling information flow and shock transmission among financial

assets [81–83] and their corresponding institutions [84]. Among various techniques used for representing firms networks, the correlation of historical return is the most dominant. This technique constructs the initial networks based on the Pearson correlation of historical returns and applies a threshold function or minimum spanning tree to sparsify the correlation network. Because most popular graph techniques are developed for unsigned graph with no negative connetions, to extend those techniques in any network, researchers often use only positive correlation or absolute value of the correlation in these cases [60, 80]. Cohen and Frazzini [56] and Herskovic [59] study the customer-supplier network at the industry level and construct new pricing factors from the network. Muslu et al. [57] suggest that the stock price of one firm be affected by other stocks within the same analysts' coverage. Study also shows firms are affiliated with investment banks, then the information of investment banks produces an underwriter network of initial public offering (IPO) and seasoned equity offering (SEO) [58]. Studies also show that firms can be linked through managers who have connections via prior employment, education, or memberships in social, cultural, and charity organizations [85, 86].

Those studies also provide essential insights into the significance of networks on the information flow between financial entities. A large body of literature documents the importance of the network in contagion effects and systematic risks [81, 82, 84, 87–91]. Carvalho [92] shows that local shocks occurring in the production network might propagate across the economy and stimulate aggregated fluctuations. Ozdagli and Weber [93] finds that the effects of monetary policy shocks are largely driven by the production networks, suggesting that the network be an important propagation mechanism of monetary policy to the real economy. Hou [94] suggest that the structures of product markets affect managers' equilibrium operation decisions. These decisions affect the risk of a firm's cash flow and sbsequent stock prices.

The third stream is related to the studies on using machine learning models to predict asset prices. Predicting the asset price is of great importance to academic research and real-world investment. Traditional asset pricing models mainly focus on uncovering risk factors [48,49]. Over the years, economists identify hundreds of factors to explain the variability in returns among assets [51]. The advancement in machine learning and deep learning allows researchers to include the full spectrum of these factors (for example, market, macroeconomic, and firm's accounting fundamentals) to predict returns or asset prices [47, 66, 95–99]. One advantage of deep learning models is to select and learn crutial factors automatically. Among them, auto-encoder is applied in [96] and [97] to forecast stock returns from historical data. Gu, Kelly, and Xiu [47] use Multi-layer-perceptron (MLP) on 94 characteristics variables to predict stock returns. Chen, Pelger, Zhu [100] use generative adversarial network to propose a non-linear asset pricing model based on no-arbitrage condition.

Given the importance of networks in financial entities, incorporating network information in asset pricing is surprisingly scant. There are only a few studies that attempt to fill the gap and examine how the network enhances the predictability of the future stock returns [59, 101–103]. My works differ from these network-based approaches in multiple aspects. A temporal graph convolutional neural network is proposed to rank assets for portfolio optimization in [102]. Different from my work, [102] use feature similarities to learn the initial graph. Herskovic [59] introduce the multi-sector information to build a static input-output network. In contrast, my models capture the time-varying information and overcomes the drawback associated with the assumption of sector similarity. Li et al. [103] propose LSTM Relational Graph Convolutional Network to predict overnight stock movement. They use a fixed correlation network among firms that fails to capture both the latent components and the dynamic changes in the network. The stock prediction model in [101] applies hierarchical attention in learning node representations with only spatial convolution.

Also, they do not consider any temporal dynamics. By contrast, my model in the recurrent diffusion step captures both spatial and temporal dependencies.

1.3 Organization of the Dissertation

The dissertation is organized as follows. Chapter 2 first discusses the problem with missing values in financial analyst earning forecast data. This chapter proposes a coupled matrix factorization algorithm to solve the missing value problem in analyst earning forecast. Finally, it provides the experimental results and important implications of the proposed methods in predict future firm earnings.

Chapter 3 extends the matrix factorization algorithm in tensor form. It discusses the importance of representing financial time-series as a tensor. This chapter introduces a novel nonlinear couple tensor completion model for heterogeneous data integration and missing value imputation. It also provides detailed experimental results and discusses the significance of the findings.

Chapter 4 starts with a brief discussion on the traditional asset pricing theories and their limitations, provides a brief survey of applying popular machine learning techniques in developing asset pricing models, and then proposes an autoencoder-based novel latent factor model for asset pricing.

Chapter 5 introduces the importance of networks on asset pricing. This chapter presents a signed graph Laplacian approach to construct a network factor "Z-score" that reflects the change points in market network. We test the implication of this network factor on equity prices.

In Chapter 6, I combine two state-of-the-art machine learning techniques method to develop an end-to-end graph neural network model and shows its applicability in asset pricing. This chapter also presents extensive experiments concerning the proposed approach and its application on asset return predictability.

Finally, Chapter 7 provides concluding remarks and discusses future research direction.

CHAPTER 2

MATRIX FACTORIZATION FOR REPRESENTATION LEARNING

2.1 Introduction

Earnings forecast conveys the expectation of firms' future cash flow and is one of the fundamental inputs for security pricing. Both academic researchers and market participants devote significant time and efforts to produce their 'best' forecast. Among them, analysts' forecast is plausibly the most common and important one. It serves as a benchmark for other prediction models and provides information to those who do not conduct their own analysis.

Over a given time period (e.g., over a quarter), multiple analysts follow one firm and generate individual reports. To capture the market overview, the mean or median forecast across analysts, also known as analysts' consensus forecast, is widely used.A large body of research shows that analysts' forecast is superior to time-series models due to analysts' knowledge and business insights and the timing advantage of the forecast [104, 105]. Despite the advantages, analysts' consensus forecast may not be the 'best' proxy for the market overview. Studies have shown that analysts' forecast accuracy is affected by their abilities, incentives, and previous experiences [32, 106–109]. As individual biases are systematic [105, 110], they cannot be offset by aggregating. Some general practices attempt to assign different weights to analysts based on their past performance. More recent papers propose to incorporate crowd-sourcing data and data with different frequencies [31, 111]. These papers attempt to produce better earnings forecast by utilizing available information. Nevertheless, none of them address the problem of missing values in the individual analyst's forecast.

The issue of missing data is common in the analysts' forecast: an individual analyst does not always generate financial reports for the same firm, resulting in a

highly incomplete and unbalanced dataset. There are different reasons behind: some are systematic whereas others are caused by randomness. Analysts may intentionally skip reporting for a firm for personal or professional interests resulting in systemic missing values [105, 110]. On the other hand, analysts enter, exit, re-enter this job market, follow one firm, stop following, and re-follow, resulting in randomly missed values. In the sample from 2000 to 2016, the average percentage of missing values for 2082 U.S. firms is 78.83% (see details in Section 2.3), which is too high to be negligible.

In this work, I aim to complement current research by investigating whether missing values in the individual analyst's forecast can reveal additional information after imputation and consequently improve the prediction of firms' future earnings. In doing so, I also evaluate how to efficiently impute missing values while preserving data properties, patterns, structure, and distribution.

In practice, there are two approaches to deal with missing data problems: marginalization and imputation. Marginalization is a list-wise deletion approach to exclude the data instances with missing values. It is not viable to the analysts' earning forecast dataset where the majority of data instances has missing values and will be eliminated by marginalization. Imputation involves replacing missing values with estimated values, for example, zero imputation, average imputation, and class mean imputation [112, 113]. These imputation techniques have several drawbacks: zero imputation on a highly sparse dataset significantly skews data towards zero and changes distribution; mean imputation is very sensitive to outliers [112]. To overcome the limitations of single imputation, [112] proposes multiple imputation (MI). More recently, machine learning techniques have been applied to handling missing data, including the k-nearest neighbor imputation (k-NNI) [114], singular value decomposition (SVD) [115], nuclear norm minimization [116], and matrix factorization (MF) [117].

These techniques work well for simulated data. However, their performance may vary for real data [118, 119]. In this chapter, I compare the imputation performance among zero imputation, mean imputation, MI, iSVD, k-NNI, and MF and show that MF consistently performs the best. The imputation accuracy captured by R^2 reaches 96%, even when the missing rate is about 90%. MF factorizes the sparse analysts forecast matrix into two low-rank factor matrix, i.e., quarter matrix U and analysts matrix V. By extracting low-rank representations as latent factors, MF simultaneously learns both individual analyst's bias and the firm's earnings trend over time that are implicitly encoded or modeled by these latent factors. Subsequently, the captured information in the latent factors is used to impute the missing values.

I also propose a coupled matrix factorization (CMF) to further improve the standard MF approach and impute the sparse analysts forecasts with another dataset of firm characteristics. The firm characteristics from quarterly financial statements provide fundamental information of a firm and have much fewer missing values. The decision to use firm characteristics as the additional dataset is motivated by previous research findings that showed firm characteristics affect analysts' forecast accuracy [39–42] and firms' earnings capacity [41, 111, 120]. By fusing two data sets with CMF, I incorporate their comprehensive information into predicting firms' future earnings. Many existing approaches use the common alternating least-squares minimization (ALS) approach that incurs significant computation and memory expenses and fails to scale to large datasets. Instead, I design a novel stochastic gradient descent (SGD) based coupled matrix factorization and attain significant improvements over the standard (coupled) matrix factorizations in terms of accuracy, simplicity for implementation, and computing speed for extremely large datasets [43].

To examine the quality and usefulness of imputation, I apply advanced machine learning methods in the downstream predictions of firms' actual earnings. Those methods include Least Absolute Shrinkage and Selection Operator (LASSO) [44],

eXtreme Gradient Boosting (XGBoost) [45], and Support Vector Regression (SVR) [46]. To evaluate the prediction performance, I apply three measures: R^2, mean squared errors (MSE), and mean absolute percentage errors (MAPE). I show that all these machine learning methods predict better with imputed data. Interestingly, I find that the most advanced machine learning method, XGBoost, without imputed data, cannot beat consensus forecast. Conversely, with the help of CMF imputed data, XGBoost beats analysts' consensus forecast by 34%. This finding confirms the necessity of data imputation and the effectiveness of the proposed CMF method.

To ensure the reliability of the results, I group all firms based on the number of analysts following and the level of forecast dispersion individually and jointly. The first one measures the quantity of input data and the latter captures the associated noise of the data. Compared with the consensus forecast, the improvement of the prediction using imputed data and machine learning methods increases as the number of analyst increases and/or analyst dispersion increases. On the other hand, the prediction performance using machine learning cannot beat the consensus forecast when the number of analysts following is small and analyst dispersion is the lowest. Those findings have several implications: first, the quality of imputing missing values depends on the amount of data; Secondly, imputing missing values is more important when available data is volatile; Third, the effectiveness of advanced machine learning methods is determined by the quantity and quality of input data, which confirms the importance of imputing missing values right before applying any advanced machine learning method.

This work contributes to the finance and accounting literature in several ways. In contrast to previous research focusing on utilizing available analysts' forecasts or combining analysts' forecasts with other factors, we, for the very first time, investigate the importance of missing values in the individual analyst's forecast. I show that with properly imputed values, the prediction accuracy of firms' future

earnings is significantly improved. This finding helps us understand better about financial analysts from two perspectives. First, missing values in the individual analyst's forecast can reveal additional and useful information about firms' future earnings. Second, available/observed analysts' forecasts do not fully reflect the firm-level information. If all fundamental information has been involved in the observed analysts' forecasts, imputed values should not contain additional information and earnings prediction will not be improved with imputed values. The results are opposite to this case and confirm the necessity of imputing missing values. Moreover, this work enriches the application of machine learning techniques in finance and accounting. Data-driven studies depend on the quality of input data. Machine learning can help to improve data quality and facilitate the following research. Using this work as an example, CMF is able to recover missing values with high quality and consequently benefits earnings prediction model. It also shows that machine learning models are imperative when data have high heterogeneity and sparsity.

The rest of the chapter is organized as follows: Section 2.2 presents the details of the models that I use for missing value imputation and earnings prediction. It also presents the criteria to evaluate model performances. Section 2.3 discusses the data used for this study and hyperparameters for experimental settings. Section 2.4 elaborates on the result of the analysis. Finally, Section 2.5 summarizes the findings of this chapter.

2.2 Methods

I align the methodology with the two distinct objectives of this work: missing value imputation and earnings prediction. I first discuss two state-of-the-art missing value imputation techniques –MF and CMF– alongside their implementations based on the SGD algorithm. Then, I detail three widely used machine learning models, including

XGBoost, LASSO, and SVR, for predicting firms' actual earnings from the imputation results.

2.2.1 Missing Value Imputation Methods

Imputing missing value for real data is always challenging because real data often involves complex distributions and interactions that must be preserved to retain the original data structure and knowledge. Accurate imputation is also essential for building successful downstream predictive models. Over the years, several machine learning-based data imputation techniques were proposed for missing data. In this chapter, I compare two advanced imputation methods, MF and CMF, with other six benchmarks. These include four traditional techniques, zero imputation, mean imputation, random-walk imputation, and multiple imputation (MI), and two machine learning-based techniques, k-NNI and i-SVD. Zero imputation involves replacing the missing values with zero. The mean imputation replaces the missing value with the average of existing values from the same class, here, the average forecast of all the non-missing analysts in the same quarter. Random-walk imputation replaces the missing value with the analyst's last available value. If the analyst's last forecast is unavailable, it will be replaced with the last reported actual Earning Per Share (EPS). MI creates multiple predictions for each missing value considering the uncertainties involved with missing values [112]. For MI imputation, I use the widely used framework for robust MI – Multiple Imputation with Chained Equation (MICE) algorithm. k-NNI imputes the missing value with the average value of its k-nearest neighbors [115]. Finally, i-SVD uses singular value decomposition to approximate missing values [121]. More details about all these benchmark methods are presented in Appendix A.1.

Matrix Factorization MF came to fame after its success in the Netflix competition in 2006. MF directly factorizes the original incomplete matrix into two low-rank

factor matrices. These low-rank factor matrices learned from observed entries in the original matrix constitute the compact representation of the underlying data structure. Therefore, they can be used to reconstruct the entire dense matrix from only a fraction of matrix entries [117]. To estimate the missing values in the original quarter-analysts' forecast matrix $X \in \mathbb{R}^{T \times N}$, I factorize it into $U \in \mathbb{R}^{T \times K}$ and $V \in \mathbb{R}^{K \times N}$ as $X = UV$ (shown in Figure 2.1). The $U \in \mathbb{R}^{T \times K}$ contains the temporal latent factors with one row for each quarter, and $V \in \mathbb{R}^{K \times N}$ represents the analysts' latent factors with each column corresponding to each analyst. Throughout the chapter, I use $t = 1, ..., T$ to represent quarters and $n = 1, ..., N$ to represent analysts.[1] K is the number (rank) of latent factors and $K < T$ and $K < N$. The earnings forecast by analyst n in quarter t is imputed as follows:

$$MF(X_{tn}) = \begin{cases} \sum_{k=0}^{K} U_{tk} V_{kn} & \text{if } X_{tn} \text{ is missing value} \\ X_{tn} & \text{otherwise} \end{cases} \tag{2.1}$$

The latent factors $U_{t:}$ and $V_{:n}$ are the embeddings of the respective quarter t or analyst n. If two analysts' forecasts are similar to each other, their embeddings will be close to each other. It is also the case for the time periods. Figure 2.1 shows that to estimate $X_{3,2}$, MF uses the latent factors from the 3rd row of U and 2nd column of V. U and V are modified by minimizing the difference between $U \cdot V$ and X on the observed entries. The objective of the matrix factorization is represented by a

[1]Follwoing standard practice, throughout the chapter, I use upper case letter for matrix, e.g., X, and range, e.g., T, and lower case letter for index, e.g., t.

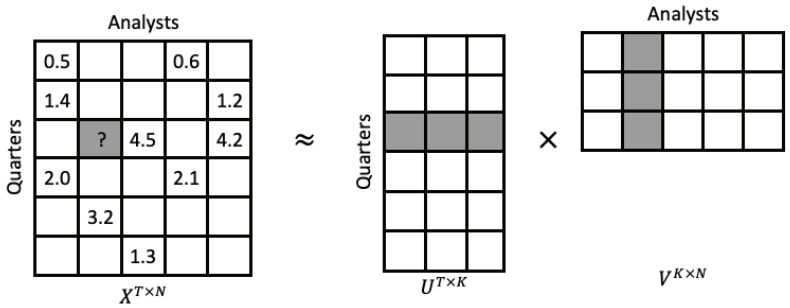


Figure 2.1 Matrix Factorization (MF) to predict missing entries. An incomplete matrix X represent all existing forecasts for N analysts and T quarters. MF adopts the machine learning paradigm and applies SGD to factorize and reconstruct the incomplete matrix element-by-element. It iterates the following steps: calculate the error between the predicted values and known values (the ground truth), apply a gradient descent optimization algorithm to estimate individual analyst's embedding (the column in gray) and a firm's temporal behaviors (the row in gray), adjust the learned embeddings in factor matrices U and V. To impute the entry marked by '?', MF performs a dot product on the corresponding row and column in gray.

function of the parameters (U, V) and defined as follows:

$$\begin{aligned}\min_{U,V} \mathcal{L}(X; U, V) &= \min_{U,V} \|(X - U \cdot V) \odot \mathbb{1}_X\|_F^2 + \alpha(\|U\|_{1,1} + \|V\|_F^2) \\ &= \min_{U,V} \sum_{t=1}^{T} \sum_{n=1}^{N} (((X_{tn} - U_{t:} \cdot V_{:n}) \odot \mathbb{1}_X)^2 + \alpha(\frac{\|U_{t:}\|_1}{T} + \frac{\|V_{:n}\|_2^2}{N}))\end{aligned} \tag{2.2}$$

Herein, $\mathbb{1}_X$ is an indicator matrix, $(\mathbb{1}_X)_{tn} = 1$, if X_{tn} has value, else $(\mathbb{1}_X)_{tn} = 0$. I use the Hadamard product $\odot$ between $(X - U \cdot V)$ and $\mathbb{1}_X$ to select the elements with the valid values because the loss function only considers the non-missing values and ignores the imputed values that have no ground truth for validation. The hyper-paramter α controls the relative impact of regularization on factor matrices U and V. Following [117], it is determined by cross validation. I use different regularization on the factor matrices, $\ell_{1,1}$ norm for matrix U and Frobenius norm $\ell_{2,2}$ for matrix V, to derive desired properties from analysts and quarters. On one hand, U encodes the timing information with K latent time factors and adopts sparsity to focus on the most relevant time factors and ensure the best generalization capability in prediction.

On the other hand, V represents analysts who tend to generate consistent forecast reports for the same firm due to the herding affect and Efficient Market Hypothesis (EMH). Consequently, I apply the ridge regularization on V to penalize any large deviation in analysts' underlying embedding factors and ultimately smooth across the future earning forecasts for the same firm.

To take the advantage of the powerful Stochastic Gradient Descent (SGD), I decompose the matrix norm into the regularization on the rows or columns in a matrix in Equation (2.2), each row in U is regularized by an ℓ_1-*penalty*[2] and each column in V by an ℓ_2-*penalty*.[3] Equation (2.2) helps understand the machine learning-based Matrix Factorization, with all values in non-empty cells of matrix X becoming the training targets and their indices being the input variables. When the training algorithm iterates over each training sample, it performs the regularization on the relevant row and column simultaneously.

I set aside 10% of the non-missing data as the validation sample set, and train the model through the SGD on the remaining 90% non-missing data (training sample). The SGD based MF is presented in Algorithm 1. SGD minimizes the given loss $\mathcal{L}(X; U, V)$ by descending on the gradient with respect to the parameters $\theta = (U, V)$. With the initial starting point θ_0, SGD iterates over the stochastic difference equation $\theta_{m+1} = \theta_m - \epsilon_m \frac{\partial \mathcal{L}}{\partial \theta}|_{\theta=\theta_m}$ to update the parameter values, where m is the index of the current step and $\epsilon_m, 1 \leq m \leq M$ is a sequence of the decreasing step sizes [122]. Instead of summarizing the gradients of all training data, I apply the SGD to decompose the loss function $\mathcal{L}$ into the individual loss term $\mathcal{L}_{tn}$ in Equation (2.2) for each data element in the training set Z. Algorithm 1 reaches the convergence once the minimum improvement in the validation set is ≤ 0.0001.

[2]The ℓ_1, the sparsity penalty, uses the absolute value of magnitude to penalize the loss function.
[3]The ℓ_2 norm used a square of the magnitude to penalize a large deviation from the ground truth.

Algorithm 1 SGD for Matrix Factorization

Require: A training set Z, initial values U_0 and V_0
  while not converged **do** {step}
    Select a training point (t, n) $\in$ Z uniformly at random.
    $U'_{t:} \leftarrow U_{t:} - \epsilon_m \frac{\partial}{\partial U_{t:}} \mathcal{L}_{tn}(X; U, V)$
    $V'_{:n} \leftarrow V_{:n} - \epsilon_m \frac{\partial}{\partial V_{:n}} \mathcal{L}_{tn}(X; U, V)$
    $U_{t:} \leftarrow U'_{t:}$
    $V_{:n} \leftarrow V'_{:n}$
  end while

Algorithm 1 uses only one data sample each time to evaluate the gradient of the loss function and leads to an unstable statistical estimation and long convergence time, particularly when the optimization process fluctuates around the minimum of the objective function. To mitigate the problem, I use the mini-batch size b bigger than one during each training iteration. The concept of stochastic gradient descend, adaptive step-size (for example, Adam Optimizer), and the mini-batchsize lays the foundation of deep learning. I leverage one popular deep learning framework, Keras [123], for a cost-effective implementation of the Distributed Stochastic Gradient Descent Matrix Factorization that is elaborated in [122]. Keras has the built-in capability of supporting distributed programming within each mini-batch and ensure the scalability of the MF model. With the help of the distributed SGD (DSGD) and the Keras deep learning framework, MF works with datasets that contain millions of rows and columns and billions of missing entries [122] and run on multiple GPUs simultaneously, resulting in faster convergence and higher scalability than the MF in Algorithm 1.

MF in Algorithm 1 only factorizes one matrix and does not use any other information except the known values in matrix X. Besides, for new quarters or for an analyst who make forecast for the very first time, it is difficult to associate the novel quarters or analyst to a reference point, resulting famous *cold start* problem[4]. In the

[4]In recommender systems, the cold start occurs when the model cannot make any inferences for users (analysts) or items (quarters) because of insufficient information. In other words,

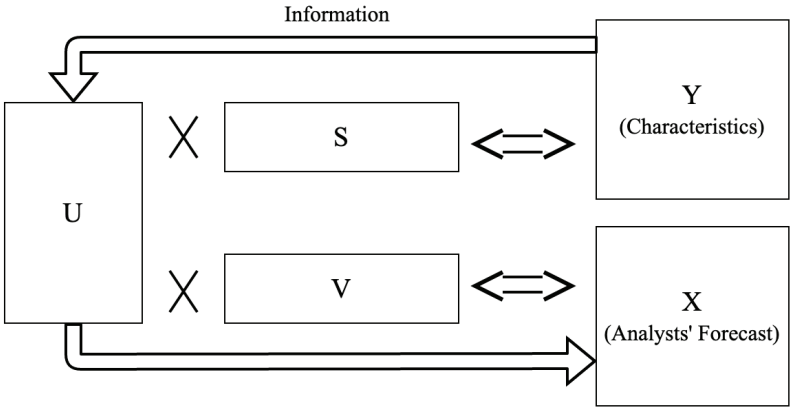


Figure 2.2 Couple Matrix Factorization (CMF). It allows the information to propagate along the temporal factor U from matrix Y to matrix X as U is shared as the temporal factor matrix for both X and Y.

following section, I discuss a novel approach to use external information to overcome the problem associated with the single dataset MF.

Coupled Matrix Factorization A variety of financial datasets are related to firms and their quarterly earnings: daily stock data show short-term returns in the market, the accounting fundamental data reflect a firm's health condition, finance news reports any major event of a company, and analysts consensus forecast reveals the general market expectation of firms' future earnings. These datasets offer different views concerning a firm and any one source alone, e.g., analyst forecast, might not provide the true representation of a firm.

In this work, I hypothesize that the internal accounting information provides objective insights of a firm's earnings. Once incorporated in the MF-based data imputation, it helps filling in absent details and reduces analysts' bias in earnings forecasts. In addition, the missing value imputation algorithms often suffer the cold start and data sparsity problems [126]. Under these circumstances, the *time* and *analysts*' latent factors learned from the analysts' matrix are less informative

lack of reference point for an analyst or quarter to be associated with other users or quarters [124, 125].

Algorithm 2 Coupled Matrix Factorization

Require: Training set Z, Analyst matrix X, Characteristics Y, initial values U_0, V_0, and S_0.
while not converged or reach the maximum steps **do** {Alternating Gradient Descent}
while not converged **do** {step}
Select a training point (t, p) $\in$ Z uniformly at random.
$U'_{t:} \leftarrow U_{t:} - \epsilon_m \frac{\partial}{\partial U_{t:}} \mathcal{L}_{tp}(Y; U, S)$
$S'_{:p} \leftarrow S_{:p} - \epsilon_m \frac{\partial}{\partial S_{:p}} \mathcal{L}_{tp}(Y; U, S)$
$U_{t:} \leftarrow U'_{t:}$
$S_{:p} \leftarrow S'_{:p}$
end while
while not converged **do** {step}
Select a training point (t, n) $\in$ Z uniformly at random.
$U'_{t:} \leftarrow U_{t:} - \epsilon_m \frac{\partial}{\partial U_{t:}} \mathcal{L}_{tn}(X; U, V)$
$V'_{:n} \leftarrow V_{:n} - \epsilon_m \frac{\partial}{\partial V_{:n}} \mathcal{L}_{tn}(X; U, V)$
$U_{t:} \leftarrow U'_{t:}$
$V_{:n} \leftarrow V'_{:n}$
end while
end while

because of insufficient signals and even missing critical data in the incomplete analyst forecast dataset. Machine learning algorithms alone cannot make up the weak signal. Therefore, I introduce the additional dataset of firm characteristics that share the same time dimensionality with the forecast dataset and provide more complete information about a firm, its operational environment, and growth pattern to regulate imputation process.

The traditional matrix completion algorithm treats two datasets separately and generates two matrices for each dataset, i.e., analyst and temporal factor matrices for forecast data, and fundamental and temporal factor matrices for characteristics dataset. On the contrary, a coupled matrix factorization algorithm (CMF) attempts to enforce the same time factor matrix during factorization and decomposition [126, 127]. By sharing the time factor, the information propagates from the dense matrix (firms accounting data) to the sparse matrix of the forecast dataset during data imputation (shown in Figure 2.2).

I use firm characteristics matrix $Y \in \mathbb{R}^{T \times P}$, with T representing quarters and P representing characteristics, to regularize the reconstruction of analysts' earnings forecast matrix X. Both X and Y share the time dimension in the unit of quarter U. The combined objective (loss) function of coupled matrix factorization is defined on the variables U, V, and S and written as follows:

$$\begin{aligned}\mathcal{L}(U,V,S) &= \lambda\|(X - U \cdot V) \odot \mathbb{1}_X\|_F^2 + (1-\lambda)\|(Y - U \cdot S)\|_F^2 + \alpha(\|U\|_{1,1} + \|V\|_F^2 + \|S\|_F^2) \\ &= \lambda\mathcal{L}(X;U,V) + (1-\lambda)\mathcal{L}(Y;U,S) \end{aligned} \tag{2.3}$$

where λ controls the weights between the two matrices and α control the relative impact of the regularizations on factor matrices U, V and S. I split the loss function into the reconstruction losses (and associated regularization penalties) $\mathcal{L}(X;U,V)$ for X and $\mathcal{L}(Y;U,S)$ for Y, and alternately process them in Algorithm 2. Similar to MF, I apply $\ell_{1,1}$ norm for matrix U and Frobenius norm for matrices V and S.

Algorithm 2 minimizes the objective function in Equation (2.3). I alternately apply the same SGD introduced in section 2.2.1 to factorize the analysts' forecast and characteristics matrices. I first factorize the dense matrix of firm characteristics Y into the time latent factor matrix $U \in \mathbb{R}^{T \times K}$ and characteristics latent factors $S \in \mathbb{R}^{K \times P}$, and iterate until convergence. Afterward, I factorize the sparse matrix X into the time latent factor matrix $U \in \mathbb{R}^{T \times K}$ and analyst latent factors $V \in \mathbb{R}^{K \times N}$, and iterate until convergence. For factorizing X, I use the trained time latent factors U of Y to set the time factor matrix U of X and apply random initialization on the analysts' factor matrix V. The shared matrix U overcomes the challenge of the cold start and data sparsity and imputes data with the high-quality key performance factors of a firm.

2.2.2 Earnings Prediction Methods

The capability and effectiveness of missing value imputation using the proposed method can be evaluated by the ability of using imputed values to predict firms' future earnings. An improvement in prediction accuracy with imputation over those without will validate the importance of MF and CMF. In this chapter, I compare MF and CMF models' performance in predicting future earnings against MI, Random-walk, and analysts' consensus forecast. I take advantage of MF and CMF and apply three widely used machine learning algorithms –XGBoost, LASSO, and SVR– on the imputed matrix to estimate firms' future earnings. The next quarter earning of a firm $\hat{y}_t$ is the function of analyst earning forecasts.

$$\hat{y}_t = f(x_{t1} \ \cdots \ x_{tN}; \theta) \tag{2.4}$$

where x_{tn} is the actual forecast if available, otherwise imputed value using MF or CMF imputation function for time period t and analyst n, $f(\cdot)$ denotes the machine learning algorithms, i.e., XGBoost, LASSO, and SVR and θ is the model parameters of respective algorithms.

XGBoost XGBoost is a gradient boosting algorithm based on the tree-based machine learning model and the first and second order derivatives in Equation (2.7) [45]. XGBoost becomes a dominant algorithm for prediction problems in various fields, especially for small to medium-size structured data where deep learning might easily overfit data. Given T data samples with N features $X \in \mathbb{R}^{T \times N}$, a decision or regression tree $f(\cdot)$ defines a mapping $q : \mathbb{R}^N \to M, w \in \mathbb{R}^M$. q maps a sample $x_t \in \mathbb{R}^N$ to one of M leaf nodes in the regression tree by following a decision path from root to the leaf. Here, q is the tree structure and M is the index of tree leaves. XGBoost ensembles K trees with the final prediction being the sum of the regression

results by each regression tree.

$$\hat{y}_t = \phi(x_t) = \sum_{k=1}^{K} f_k(x_t), \quad f_k \in \mathcal{F}. \tag{2.5}$$

$\mathcal{F} = \{f(X) = w_{q(X)}\}$ is the space of regression trees. Each f_k is associated with a different tree structure q and leaf weights w. Instead of using a fixed tree structure, XGBoost adds a new function tree f_k and generates a prediction $\hat{y}_t^{(k)} = \sum_{t=1}^{T} f_k(x_t)$ for x_t at iteration k. To mitigate the overfitting problem, XGBoost penalizes the model in the magnitude of the weight w and depth of all constituency trees by optimizing the following objective function of variable $\hat{y}_t$:

$$\mathcal{L}(\phi) = \sum_{t} l(y_t, \hat{y}_t) + \sum_{k} \Omega(f_k) \tag{2.6}$$

The first term in the loss function is the fitness of the model, and the second part $\Omega(f) = \gamma M + \frac{1}{2}\lambda\|w\|^2$ is a regularization term to control the model complexity and penalize large and complex model parameters. During each iterative, a Taylor series of Equ. 2.6 expands the loss function into the constant term that is fixed in the previous $k-1$ stages and the first and second-order derivatives of the loss function. The new objective function for the current iterative k is defined as follows:

$$\widetilde{\mathcal{L}}^{(k)} = \sum_{t=1}^{T} [g_t f_k(x_t) + \frac{1}{2} h_t f_k^2(x_t)] + \Omega(f_k), \tag{2.7}$$

where $g_t = \frac{\partial l(y_t, \hat{y}_t)}{\partial \hat{y}_t}\big|_{\hat{y}_t = \hat{y}_t^{(k-1)}}$ and $h_t = \frac{\partial^2 l(y_t, \hat{y}_t)}{\partial \hat{y}_t^2}\big|_{\hat{y}_t = \hat{y}_t^{(k-1)}}$ are the first and second order gradient statistics of the loss function respectively. XGBoost essentially builds a regression tree to minimize the new objective function in Equ. 2.7 with the optimal

Quarter	Analyst-1	Analyst-2
201201	?	2.9
201202	3.5	?
201203	4.1	3.9
201204	4.2	4.0

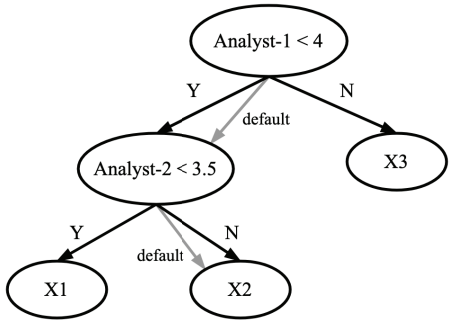


Figure 2.3 XGBoost tree structure with default (gray) directions.

weight w_m^* for leaf m that is calculated by $w_m^* = -\frac{\sum_{t\in I_m} g_t}{\sum_{t\in I_m} h_t+\lambda}$. However, the computational complexity of enumerating all possible trees is steep. Starting from a single terminal node tree, [45] uses a greedy algorithm to iteratively add branches to the tree (For details see [45]).

XGBoost has a built-in sparsity awareness and, therefore, can handle missing values in data analysis, which prompts us to evaluate it as a downstream predictive model as well. Figure 2.3 demonstrates that XGBoost adds a default path in each tree node: when the model encounters a missing value, the path takes the default route to descent. As a result, the algorithm automatically learns the best path to take in processing missing values [45]. In the experiment, I apply XGBoost to predict from both the original data with missing values (XGBoost) and the imputed data by MF (MF+XGBoost) and CMF (CMF+XGBoost). This enables us to compare the performance improvement from the imputation on financial analysts' earnings forecast by MF and CMF.

Least Absolute Shrinkage and Selection Operator (LASSO) I use LASSO as the second model for evaluating the downstream prediction task. LASSO is a regression analysis method that has the capacities of automatically selecting variables and regularizing the model complexity. By using ℓ_1-penalty in the loss function, LASSO shrinks the weights of multiple input feature variables toward zero and

essentially performs an automatic feature selection. As a result, the model removes correlated features and avoids overfitting (For details see [44]).

$$\mathcal{L}(w) = \sum_{i=1}^{T}(y_t - \sum_{n} x_{tn}w_n)^2 + \lambda\sum_{n=1}^{N}|w_n| \tag{2.8}$$

where λ is the shrinkage parameter that controls the strength of the ℓ_1 penalty.

Support Vector Regression (SVR) The final predictive machine learning techniques use for predicting firms earning is SVR [46]. Instead of minimizing the error rate, SVR tries to fit the error within a certain threshold α. For this study, I use a nonlinear SVR with the radial basis function (RBF) kernel, also known as the Gaussian kernel. The nonlinear SVR minimizes the loss function as the follows:

$$\begin{aligned}\mathcal{L}(\alpha) &= \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{T}(\alpha_t - \alpha_t^*)(\alpha_i - \alpha_i^*)G(x_t, x_i) + \epsilon\sum_{t=1}^{T}(\alpha_t - \alpha_t^*) - \sum_{t=1}^{N} y_t(\alpha_t - \alpha_t^*) \\ \text{subject to } &\sum_{t=1}^{T}(\alpha_t - \alpha_t^*) = 0; \;\; \forall t : 0 \le \alpha_t \le C \;\text{ and }\; \forall t : 0 \le \alpha_t^* \le C\end{aligned} \tag{2.9}$$

where, α_t and α_t^* are non-negative Lagrange multipliers for each observation x_t, and $G(x_t, x_i) = exp(-\frac{\|x_t - x_i\|^2}{2\sigma^2})$ is the RBF kernel function. SVR predicts the actual EPS as follows:

$$\hat{y}_t = f(x) = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)G(x_n, x) + b. \tag{2.10}$$

2.2.3 Evaluation Metrics

I adopt three evaluation criteria to compare the performance between the proposed method and other reference methods: R-Squared (R^2), mean squared error (MSE), and mean absolute percentage error (MAPE). To evaluate the imputation algorithms,

I compare the observed analyst-quarter forecast (ground truth) with the imputed analyst-quarter forecasts. The R^2 and MSE are defined as follows:

$$R^2 = 1 - \frac{\sum_{t=0}^{T}\sum_{j=0}^{N}(x_{tn} - \hat{x}_{tn})^2}{\sum_{t=0}^{T}\sum_{n=0}^{N}(x_{tn} - \bar{x})^2} \tag{2.11}$$

$$MSE = \frac{1}{TN}\sum_{t=0}^{T}\sum_{n=0}^{N}(x_{tn} - \hat{x}_{tn})^2 \tag{2.12}$$

where, $\hat{x}_{tn}$ is the imputed value for time period t for analyst n, x_{tn} is the actual forecast, and $\bar{x}$ is the mean forecast among all analysts' of the firms. A higher value of R^2 indicates a better prediction accuracy, i.e., the estimated value is closer to the original value. $R^2 = 0$ signifies that the model is unable to explain any variability of the data around mean. Whereas, a negative $R^2 < 0$ signifies that the model performance is worse than the mean. On the other hand, a smaller value for MSE indicates better prediction accuracy. $MSE = 0$ denotes that the model accurately estimates the original values.

Although R^2 and MSE are the two most popular measures in evaluating machine learning model performance, large values in the data set might influence these two metrics. These two metrics are sensitive to the analysts' EPS forecast data because the EPS of some firms can be very large. Therefore, I apply a third measure, Mean Absolute Percentage Error (MAPE), defined as follows:

$$MAPE = \frac{1}{TN}\sum_{t=0}^{T}\sum_{n=0}^{N}\frac{\|(x_{tn} - \hat{x}_{tn})\|}{\|x_{tn}\|} \times 100 \tag{2.13}$$

To avoid the unexpected divided-by-zero error, I use a small constant (0.005) to the denominator in the equation if the scale of actual earnings is smaller than 0.005. A smaller value for MAPE indicates better prediction accuracy. The matrix imputation and the downstream earning predictions are all supervised machine learning algorithms with the regression target. Their evaluation metrics are identical. To evaluate the performance of the downstream earning prediction, I make a slight modification in the above three Equation (2.11, 2.12, and 2.13) to compare the prediction result $\hat{y}_t$ for each quarter with the actual EPS of that quarter y_t.

$$R^2 = 1 - \frac{\sum_{t=0}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=0}^{T}(y_t - \bar{y})^2} \tag{2.14}$$

$$MSE = \frac{1}{T}\sum_{t=0}^{T}(y_t - \hat{y}_t)^2 \tag{2.15}$$

$$MAPE = \frac{1}{T}\sum_{t=0}^{T}\frac{\|(y_t - \hat{y}_t)\|}{\|y_t\|} \times 100 \tag{2.16}$$

Similar to the imputation MAPE, I set the minimum of the denominator to 0.005.

2.3 Data and Hyperparameters

The quarterly financial analysts' forecast of earnings per share (EPS) data is collected from I/B/E/S, Thomson Reuters. I consider the time from quarter 1, 2000 to quarter 4, 2016. This data contains almost 10,000 firms and 14000 analysts. Because machine learning models require a large amount of data, I perform some filtering procedures on all the available firms and analysts while retaining the maximum number of firms

Table 2.1 Data Description

		Missing Value				No. of Analysts			
Data Group	N	Average	STD	Max	Min	Average	STD	Max	Min
Original Data	2082	78.83%	4.66%	90.88%	60.64%	45.68	27.95	173	11
High Analysts	687	76.27%	3.92%	86.40%	60.84%	75.98	26.19	173	30
Moderate Analysts	702	78.81%	4.11%	88.18%	61.06%	39.41	10.41	76	14
Low Analysts	693	81.39%	4.46%	90.88%	61.85%	21.99	6.82	45	11
High Volatility	687	80.36%	4.73%	90.88%	60.84%	37.78	23.71	171	11
Moderate Volatility	702	79.02%	4.39%	90.19%	62.88%	44.70	27.88	170	11
Low Volatility	693	77.13%	4.28%	89.25%	61.85%	54.51	29.39	173	11

Note: N denotes number of firms. Firms are divided into tercile based on the average number of analysts following (middle panel) and the average standard deviation among analysts forecasts over time (bottom panel).

with the maximum observed data instances. First, I only select firms that appear at least 48 quarters out of the total 68 quarters, which allows us to have sufficient historical records. Second, I remove firms with less than ten different analysts during the sample period. Third, I remove analysts who make less than three forecasts in the considered period. This helps us eliminate some potential noises. The final sample involves 2082 firms and 9785 analysts. The relevant statistics of the dataset is presented in Table 2.1.

However, the data distribution of the analysts' earnings forecast is not uniform across firms. Not all firms receive the same level of attention from financial analysts: big firms with high-growth potential attract more analysts than small firms. In addition, a firm's earnings forecasting varies in terms of complexity. The EPS forecasts for big and mature firms generally have lower variance compared to those for small firms. Table 2.1 also reports the statistics for different groups. I consider two grouping criteria: the number of analysts following and the analysts' dispersion/volatility. The number of analysts following is the average number over time for a given firm, a proxy for data quantity. The analysts' volatility is measured as the average value of standard deviation among analysts' forecasts over time for a

Table 2.2 Firm Characteristics

Variable	Description and calculation procedure
FV_1	Inventory = ΔInventory (INVTQ) − ΔRevenue (REVTQ)
FV_2	Account receivable = ΔAccount receivable (RECTQ) − ΔRevenue
FV_3	Capital expenditure = ΔIndustry CAPX − ΔFirm CAPX (CAPXQ)
FV_4	Gross margin = ΔRevenue − ΔGross margin ($REVTQ - COGSQ$)
FV_5	SG&A expenses = ΔSG&A($XSGAQ$) − Δ$Revenue$
FV_6	log(Total assets (ATQ))
FV_7	Dividend payment (DVY)
FV_8	Divd = 1 if the firm at a given year has dividend information and dividend payment is greater than zero, otherwise 0.
FV_9	Net income (IBQ)
FV_{10}	negNI = 1 if the firm at a given year has negative earnings, otherwise 0.
FV_{11}	$IBQ - OANCFQ$
FV_{12}	Investment = ΔATQ
FV_{13}	$BM = \log(\frac{CEQQ}{PRCCQ \times CSHOQ})$
FV_{14}	$ME = \log(PRCCQ \times CSHOQ)$
FV_{15}	Firm-specific stock return from CRSP during month m less the same-industry portfolio return in month m before actual earnings announcement t.
$FV_{16/17}$	Most recent daily/monthly close price before the date of earnings announcement t
FV_{18}	Standard deviation of (stock daily return minus market return) in month m before actual earnings announcement t.
FV_{19}	Leverage = long-term debt (DLTTQ)/total equity (CEQQ)

given firm, a proxy for data complexity. I then sort firms into tercile based on those two criteria, respectively, top 33%, bottom 33%, and remaining 34%.

As briefly mentioned in the introduction, the missingness in analysts' forecast data can be both random and systematic. Analysts may intentionally skip reporting for a firm for personal or professional interests. For example, analysts choose not to report when there is a bad news happening to a given firm, which results in multiple missing values for that firm at the same time period, i.e., systemic missing values [105, 110]. On the other hand, analysts enter, exit, re-enter this job market, follow one firm, stop following, and re-follow, resulting in randomly missed values. The existence of systemic missing value justifies that analysts' forecast data is not Missing Completely

at Random (MCAR) but Missing at Random (MAR). The MAR assumption is built based on the premise that the missingness on the data can be estimated by the other analysts' available observations and the analyst's historical forecasts. One can argue that the systematic missing values may provide optimistic predictions of unseen values as analysts sometimes choose not to report a bad forecast to maintain a good relationship with management [128, 129]. These may qualify the data as Missing Not at Random (MNAR). However, these types of missing values are very few compared to the total number of missing values for a firm. In addition, practical applications show that MF can work reasonably well for MNAR data too [117, 130], and the incorporation of firm-level information into the missing data imputation process by CMF can tackle this systematic bias.

To augment highly sparse data of analysts' earnings forecasts with the auxiliary firm characteristics information in CMF, I adopt 19 widely used firm characteristics following the literature [33, 41, 120]. The firm characteristics data mainly come from two sources, quarterly accounting variables from COMPUSTAT and daily and monthly stock prices and returns from CRSP. I fix some recording errors such as negative stock prices and remove the observations with missing total assets. Table 2.2 provides a detailed description of each selected variable.

To evaluate the imputation performance of different imputation algorithms, I pre-select a small subset of these 2082 firms. For assessing the stability of imputation, I increase the missing values by randomly removing some observed values from the data. The goal is to select firms with highly populated data to ensure that I can still choose a reasonable amount of sample data when the percentage of missing values increases. To meet the requirement, I select firms with at least 1300 forecasts in the entire period. After applying the criterion, I obtain 51 firms for analyzing initial imputation performance.

Table 2.3 Hyperparameter Settings

Model	Hyperparameter	Values
MF and CMF	Latent factors (K)	3, 5, 10, 15, 20
	Weight (λ)	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
	Learning Rate (e)	1e-5, 1e-4, 1e-3, 1e-2
	Regularization (α)	1e-4, 1e-3, 1e-2, 1e-1
LASSO	Regularization (λ)	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
XGBoost	Learning Rate (e)	1e-5, 1e-4, 1e-3, 1e-2
	Regularization (λ)	e-4, 1e-3, 1e-2, 1e-1
SVR	Epsilon (ϵ)	1e-5, 1e-4, 1e-3, 1e-2
	Constant (C)	1, 2, 3, 5, 10

The performance of machine learning models often depends on hyper-parameters. To find the hyper parameters' optimal configuration for each model, I use 10-fold cross-validations and perform a grid search on hyper-parameters in Table 2.3. The potential grid values are selected according to the common practice in machine learning and the original model designer's choices (e.g., [45, 117]). For the rolling window analysis in Section 2.4.5, I perform the grid search for the first model of the rolling window, i.e., Q1-2015. The best parameters are then used for the remaining models throughout the rolling window process. The low-rank representation of data depends on data complexity. As a result, it is tricky to set the dimensionality of latent factors, a.k.a. rank K for MF and CMF. The computational complexity increases with a large K. For optimal performance, K must be substantially lower than the number of analysts and quarters. I apply a sampling approach and attempt to perform a grid search using the value on Table 2.3 on a small subset of firms. Although the actual rank required for different firms can vary, my assessment shows that the marginal benefits of a rank higher than ten are very low. Therefore, to ensure consistency, I select rank $K = 10$ for both MF and CMF analysis. Finally, to avoid overfitting, I use multiple random seeds, i.e., ten, for the train-test split and average the ten trials'

results. Using multiple random seeds also allows us to overcome the spurious effect of outliers in the test data.

2.4 Results and Discussion

There are two sequential objectives in my primary analyses: missing value imputation and earnings prediction. I first evaluate MF in imputing missing values along with other baseline methods. Next, I apply three machine learning models to both in-sample and out-of-sample predictions with MF imputed data. The in-sample prediction involves the imputation to the entire data sample and a prediction model with 90% training set and 10% test set. In-sample prediction intends to demonstrate the usefulness of imputing missing values without considering the prediction, and therefore, involves the entire sample in the imputation step. By contrast, the out-of-sample prediction focuses more on earnings forecast, i.e., it only imputes the up-to-date observations and then predicts one-quarter ahead earnings. Finally, to argue that single dataset is not enough for missing value imputation, I use the out-of-sample approach to impute missing values with a coupled matrix factorization (CMF) and predict one-quarter ahead earnings based on CMF imputed data.

2.4.1 Missing Value Imputation

In this section, I perform robustness tests to evaluate the proposed missing value imputation's efficiency and superiority to the other available techniques. To show this, I randomly remove entries from the available EPS forecast data and create a series of datasets with 70%, 75%, 80%, 85%, 90%, 95%, and 99% missing values for each firm. The randomly removed values from original data are set aside to use as the benchmark values to evaluate and compare the calculated values by the given imputation techniques. I compare the performances of the proposed MF with Zero imputation, mean imputation, MI, random-walk imputation, k-NNI, and i-SVD.

Table 2.4 Imputation Performance at Different Percentage of Missing Value

Metrics	Algorithm	70%	75%	80%	85%	90%	95%	99%
R^2	Zero	0.8253	0.6400	0.4378	0.2684	0.0653	-0.1307	-0.2857
	Random-Walk	0.7081	0.6801	0.4863	0.4526	0.3204	0.1106	0.0640
	Mean	0.9594	0.9215	0.8705	0.8289	0.7709	0.5588	**0.1550**
	MI	0.9954	0.9845	0.9756	0.9576	0.9129	0.4442	-1.0044
	iSVD	0.9915	0.9839	0.9564	0.8970	0.6881	0.2383	-0.2904
	k-NNI	0.9920	0.9885	0.9711	0.9570	0.8987	0.4070	-0.2787
	MF	**0.9934**	**0.9931**	**0.9824**	**0.9772**	**0.9607**	**0.8593**	-0.0134
MSE	Zero	0.0474	0.0987	0.1527	0.1990	0.2553	0.3081	0.3521
	Random Walk	0.0070	0.0165	0.0200	0.1434	0.1869	0.2807	0.3311
	Mean	0.0089	0.0182	0.0289	0.0385	0.0515	0.0764	0.1705
	MI	0.0006	0.0016	0.0037	0.0069	0.0213	0.0816	0.1840
	iSVD	0.0011	0.0030	0.0077	0.0222	0.0771	0.1947	0.3598
	k-NNI	0.0008	0.0019	0.0037	0.0065	0.0174	0.1282	0.3511
	MF	**0.0005**	**0.0010**	**0.0018**	**0.0026**	**0.0055**	**0.0283**	**0.1504**
MAPE	Zero	4.3210	9.0601	14.0404	18.3212	23.3001	28.3131	32.1421
	Random-Walk	2.6212	5.4829	9.1386	12.3305	15.1359	25.6746	34.3193
	Mean	4.5912	9.5592	14.9589	19.9721	25.6145	33.0763	40.4842
	MI	0.9437	2.1685	3.9593	6.9882	12.6171	34.3544	69.1211
	iSVD	1.0021	2.2858	3.9646	6.6562	12.7212	22.5952	31.9931
	k-NNI	1.0031	2.3412	4.1742	6.6457	11.6842	24.6332	32.5212
	MF	**0.7321**	**1.5702**	**2.5452**	**3.4637**	**5.1496**	**9.0416**	**27.8907**

Note: The columns represent the total percentage of missing value after randomly deleting some existing observations. R^2, MSE, and MAPE are calculated using Equation (2.11), (2.12), and, (2.13), respectively. Smaller MSE, MAPE and larger R^2 indicate better accuracy..

Table 2.4 reports the performance of the different algorithms with the same setting of evaluation scenarios where the best performing algorithm is in bold font. Almost in all data settings, MF outperforms other imputation techniques in all three performance metrics. Notably, the reduction in MSE by MF over the mean imputation is 94% for 70% missing values, 93% for 80% missing values, 89% for 90% missing values, 62% for 95% missing values, and 12% when 99% data is missing. Among all studied methods, the zero imputation and random-walk imputation have the worst performance. MI outperforming the mean imputation in all three performance metrics

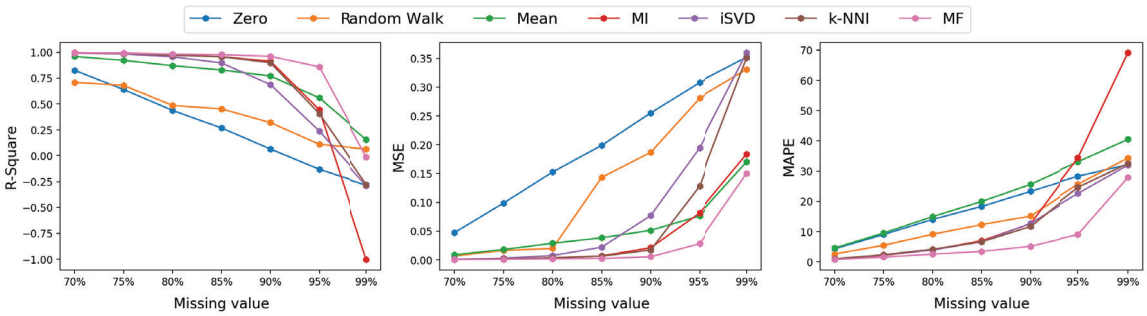
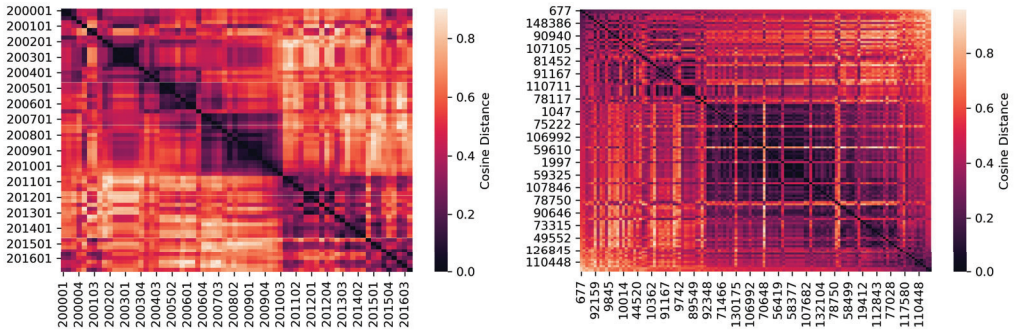


Figure 2.4 Imputation performance at different level of missing values.

for up-to 90% missing values, and under-performing with 95% and 99% missing values. The poor performance of zero imputation is understandable as replacing missing values with zero does not improve the information rather adds noises [112, 113]. The results from the random-walk model signify that the firm performance and EPS change over time, and consequently, the historical performance can not surrogate the current ones.

Table 2.4 and Figure 2.4 also show that most imputation techniques perform well up-to 90% missing value. Specially, when the missing percentage increases to 99%, data becomes extremely sparse, rendering the performance decline of all techniques. This findings show that without enough data, almost all sophisticated machine learning models fail to impute missing values effectively. It is also evident that with 99% missing values, MF retains its superiority over the mean and zero imputations in regards to MSE and MAPE. Other techniques, such as MI, iSVD, and k-NNI, lose their advantage over the mean forecast under extremely sparse conditions. Because these techniques approximate the missing values using a linear combination of known values and are sensitive to the number of known values. It might be interesting to note that, with 99% missing value, all imputation techniques have negative R^2, except the mean and random-walk imputation. However, the seemingly good R^2 performance of the mean imputation at a high percentage of missing value comes from the definition of R^2. R^2 compares the imputation performance with the mean value of the entire matrix. Therefore, the R^2 of mean imputation never falls below

zero. On the contrary, other methods including MF get a negative R^2. They suffer from the randomness and noises under such an extremely sparse dataset and cannot outperform a simple mean algorithm in term of R^2. This is an extreme case and might be unlikely for real data; meanwhile, it also suggests that missing value imputation methods for such sparse data might not be optimal. Overall, MF provides the best performance in all cases with just one exception.



(a) Quarter **(b)** Analyst

Figure 2.5 Pairwise cosine similarities among (a) 'Quarter' factors and (b) 'Analyst' factors learned from the 'Apple Inc.' analyst EPS forecast data. The precise dividing line in December 2009 between the two groups in the 'Quarter' factors represents the impact of the 2008-09 global financial crisis.

The superior performance of MF model in imputing missing value comes from the model's underlying architecture. MF solves the problem of missing value by first learning the bias of analysts and the seasonal behaviors from individual data instances, and then applying the learned factors to recover the remaining target values. The rationals come from the observations that individual analysts have a bias on corporate earnings and tend to over-/under-estimate their values. The ML-based algorithm encodes these biases in the embedding (vector representation) that represent individual analysts and can be used to measure the similarity among analysts, as shown in Figure 2.5b. Moreover, a firm's earnings might have seasonality that affects its actual earnings at different quarters and is subject to the global economic environment. Figure 2.5a shows that the machine learning algorithm also

Table 2.5 Predicting Firm Earnings

Method	R^2-Train	R^2-Test	MSE-Train	MSE-Test	MAPE-Train	MAPE-Test
Mean		0.7676		1.2607		41.3800
Random-Walk		0.2340		2.6447		114.2348
XGBoost	0.9943	-1.7070	0.0038	0.8527	29.2431	62.2416
MI+LASSO	0.6045	0.5777	0.4082	0.4205	45.9482	59.1836
MI+XGBoost	0.9987	0.7117	0.0016	0.2912	**10.257**	47.1694
MI+SVR	0.7784	0.7038	0.0682	0.3201	42.2457	48.6601
MF+LASSO	0.8419	0.7557	0.1439	0.5429	31.1921	41.4667
MF+XGBoost	**0.9997**	**0.8097**	**0.0003**	0.0873	20.2296	**24.4457**
MF+SVR	0.9139	0.7974	0.0492	**0.0866**	20.6630	24.6801

Note: This Table shows the training and testing performances. I use the values in the columns labeled "-Test" to evaluate the real model performance. R^2, MSE, and MAPE are calculated using Equation (2.14), (2.15), and, (2.16) respectively. Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term.

produces the latent representation of a firm's temporal behavior and displays two temporal regions in the heat maps of the pair-wise similarities among quarter factors. The two temporal regions have a boundary at 2009∼2010 and expose significant events in the local and global economic environment of a company.

2.4.2 In-Sample Prediction of Firm Earnings

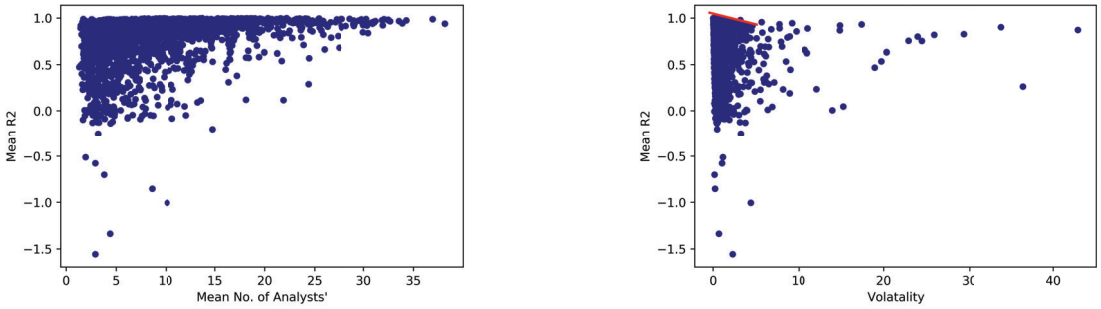
In this section, I show the necessity to impute missing values. I use the MF-imputed data along with three prediction methods, LASSO (MF+LASSO), XGBoost (MF+XGBoost), and SVR (MF+SVR) and compare the result with MI-imputed (Multiple Imputation) data using the same three prediction methods (MI+LASSO, MI+XGBoost, and MI+SVR), the analyst consensus forecast (Mean), historical EPS forecast (Random walk), and XGBoost on the original data without data imputation (XGBoost). The use of XGBoost helps us understand the effect of the imputation method on the same prediction algorithm. Table 2.5 reports the average prediction performance of all tested models in all 2082 studied firms. For each firm, I divide its data into the training and test subsets with a 90:10 split ratio and no overlapping between training and test data subsets.

The MF data imputation significantly improves the performance in all three evaluation metrics of R^2, MSE, and MAPE. For example, XGBoost with complete matrix, i.e., MF+XGBoost, makes almost 80% (in MSE) and 60% (in MAPE) improvements over a simple XGBoost alone. The XGBoost with missing value has the negative R^2, indicating it is worse than the mean prediction. Even for the current industry best practice, a.k.a., the mean prediction with available analysts forecasts in a given quarter (R^2= 76%), my imputation technique is still able to outperform it. MF+XGBoost improves performance by 5% in R^2, 93% in MSE, and 41% in MAPE. Among the two imputation methods (MI and MF), the superiority of MF is significant: MF+LASSO outperforms MI+LASSO by 30%, MF+XGBoost outperforms MI+XGBoost by 14%, and MF+SVR outperforms MI+SVR by 13%, all in R^2.

In Table 2.5, the two most sophisticated machine learning models, MF+XGBoost and MF+SVR have almost identical performance in all three metrices. Confirming that a complete matrix by MF already provides the majority of knowledge from data and the selection of a downstream prediction model is less important. Table 2.5 also demonstrates another important aspect of machine learning models, i.e., the overfitting of models. XGBoost with missing data has severe overfitting problem, e.g., 99% training R^2 with negative test R^2. However, with imputation, this overfitting problem is reduced significantly. In MF+XGBoost, the R^2, MSE, and MAPE of test data are greatly improved. I observe that the difference between SVR's training and testing performance is relatively small, signifying that SVR has a much smaller overfitting error than does XGBoost.

2.4.3 Impact of Analysts' Size and Volatility on Earning Prediction

To ensure the reliability of the results, I investigate whether the imputation performance varies across firms. Earlier studies reveal that analysts' attributes, firm



(a) No. of analysts' and performance **(b)** Volatility and performance

Figure 2.6 Forecast accuracy in relation to average analysts' and forecast volatility.

size, and the variance of analyst forecasts can affect the forecast accuracy [42,107,131]. Figure 2.6 depicts how the mean prediction accuracy is influenced by the number of analysts who follow a firm in each quarter (Figure 2.6a) and the variance of forecasts by the analysts for a firm (figure 2.6b). In Figure 2.6a, when the number of analysts for a firm increases, the volatility caused by the random errors of individual analysts will be averaged out, leading to higher prediction accuracy. For firms with less than five analysts per quarter, the R^2 values are scattered, some of which even have negative R^2. In contrast, the R^2 values of the firms with more than 15 analysts per quarter never fall below 0. Figure 2.6b shows the relationship between the variance of analysts forecast for a firm and the mean prediction accuracy. Ignoring the outliers in the image, the steep slope in the top-left corner (red line) of figure 2.6b depicts the adverse effect of volatility on the prediction accuracy. To study the impact of these phenomena discussed in Section 2.3, I partition firms with two grouping criteria: the number of analysts following and the analysts' forecast dispersion/volatility. For each criterion, I further sort firms into three groups: top 33%, bottom 33%, and remaining 34% and obtain nine groups. Table 2.1 elaborates on the data distribution in each group.

Table 2.6 presents the double sorted performance metrics of all methods in both grouping criteria. I sort firms into three by three groups independently based on the number of analysts following and the analysts' volatility. Not surprisingly,

Table 2.6 Impact of Analysts Size and Forecast Volatility on Earning Prediction

		High Analysts'			Moderate Analysts'			Low Analysts'		
	Method	R^2	MSE	MAPE	R^2	MSE	MAPE	R^2	MSE	MAPE
High Volatility	Mean	0.7826	0.2135	67.5901	0.6657	1.5503	79.3914	0.5159	0.6911	94.3921
	Random-Walk	0.2035	1.0273	192.0067	0.0057	3.6096	209.0410	0.0707	1.4846	207.4898
	XGBoost	0.5004	0.1209	85.1803	-3.9704	5.0733	113.1449	-3.4969	1.2255	139.5506
	MI+LASSO	0.7801	0.3306	71.9107	0.4871	0.9218	98.5722	0.0558	1.1696	135.2802
	MI+XGBoost	0.8201	0.1482	52.1654	0.7021	0.6348	87.3713	0.2272	0.9938	105.0198
	MI+SVR	0.8514	0.1952	54.9430	0.6970	0.6741	86.0145	0.2225	1.0959	116.9838
	MF+LASSO	0.8506	0.2353	61.2345	0.6886	1.4303	69.9521	0.5330	0.4022	77.5301
	MF+XGBoost	**0.8762**	**0.0544**	**27.3921**	**0.7562**	**0.1530**	**39.4568**	0.6016	0.1771	53.4145
	MF+SVR	0.8662	0.1322	27.4797	0.7424	0.1539	41.1751	**0.6093**	**0.1595**	**51.7828**
Moderate Volatility	Mean	0.8358	0.2373	29.9278	0.7742	0.8337	33.9312	0.6732	0.9551	37.1662
	Random-Walk	0.3021	0.9073	101.1423	0.2225	2.2389	107.6158	0.0242	13.6891	114.0054
	XGBoost	0.7023	0.1249	44.1773	0.5124	0.3345	64.8611	-1.6620	0.7404	61.4155
	MI+LASSO	0.7986	0.2824	28.1589	0.4340	0.3308	47.6859	0.3598	0.5949	69.7596
	MI+XGBoost	0.8695	0.0610	24.0066	0.7533	0.1839	34.6138	0.4103	0.5004	62.4213
	MI+SVR	0.8948	0.0455	21.3093	0.7713	0.1279	33.6817	0.4494	0.5230	66.5452
	MF+LASSO	0.8430	0.2306	25.1546	0.7937	0.8000	31.1773	0.5563	1.1749	41.7411
	MF+XGBoost	0.9150	**0.0273**	17.3154	**0.8191**	0.1075	23.4957	**0.7064**	0.2087	**28.3048**
	MF+SVR	**0.9168**	0.0298	**16.3782**	0.8077	**0.0744**	**23.3349**	0.6617	**0.1491**	30.4434
Low Volatility	Mean	0.9348	0.0236	8.7145	**0.9100**	**0.0098**	**9.9303**	**0.8159**	**0.0316**	**11.4230**
	Random-Walk	0.5577	0.0938	30.2582	0.5119	0.0774	31.8188	0.4212	0.1380	29.8386
	XGBoost	0.8958	0.0141	14.0510	0.6229	0.0181	15.3146	0.5327	0.0227	22.5111
	MI+LASSO	0.8962	0.0143	10.5973	0.8064	0.0567	21.1950	0.6055	0.1107	48.1015
	MI+XGBOOST	0.9405	0.0091	9.8323	0.8446	0.0319	17.8272	0.7570	0.0746	37.0430
	MI+SVR	0.9320	0.0100	7.0174	0.8578	0.0484	16.6871	0.6873	0.0744	35.3539
	MF+LASSO	0.9279	0.0429	10.2301	0.8158	0.0352	14.4540	0.6721	0.2444	16.7980
	MF+XGBoost	**0.9574**	0.0052	7.0121	0.8876	0.0100	10.6410	0.7681	0.0429	12.9903
	MF+SVR	0.9549	**0.0043**	**6.8800**	0.8809	0.0107	10.8807	0.7363	0.0651	13.8117

Note: R^2, MSE, and MAPE are calculated using Equation (2.14), (2.15), and, (2.16) respectively. Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term. The reported values are from test data set. Firms are sorted into 3 by 3 groups independently based on both number of analysts following and analysts dispersion/volatility.

when data are sufficient and consistent, almost all model performs exceptionally well, as shown in the group of high analysts and low volatility. In contrast, as I move from the lower-left panel to the upper right, the prediction accuracy deteriorates, with the worse performance in the group of low analysts and high volatility at the other end of the spectrum. In most cases, the combination of MF with advanced machine learning techniques (MF+LASSO, MF+XGBoost, and MF+SVR) performs significantly better than other models. The magnitude of improvement is significant when the analysts' forecasts have high volatility. The improvement can be further enhanced when a high number of analysts follow the firm and generate enough data.

The mean prediction performs better than the imputation based methods in the lower right panel of Table 2.6. I attribute the observation to the central limit theorem and confidence interval. When enough analysts make forecasts for a firm and the forecast variation is low, the sample mean closely approximates to the real mean. The R^2 value for all high-analysts group (93%) is higher than that of the moderate (91%) and low-analysts groups (81%), which is the evidence of the central limit theorem. In addition, the 95% confidence level for analysts forecast of firms for high-volatility, moderate-volatility, and low-volatility group is ±82.99%, ±32.03% and ±9.36% respectively. When the data volatility is low, the mean prediction has a narrow confidence interval to encompass the true earning. Therefore, without any additional information, it is extremely hard to surpass the mean prediction. These findings are also consistent for single sorted groups. The discussion on the single sorted groups are available in Appendix A.2.

Overall, I show that the MF+XGBoost and MF+SVR consistently perform well across different groups, less sensitive to the number of analysts following and the analysts' dispersion. MF imputation with advanced machine-learning techniques improves the forecast accuracy most when the data is very volatile. The experiment results confirm that the MF model with its latent factors mitigates the inherent volatility, reduces the uncertainty, and improves the information quality in forecast data.

2.4.4 Out-of-Sample Prediction of Firm Earnings

In-sample prediction analysis mainly confirms the necessity to impute missing values and the consistent out-performance of MF as the imputation technique. However, the issue of 'information leak' can be a concern for the earnings prediction in practice. Here, 'information leak' is referred to that in-sample prediction uses future information, which is not available at the moment to make a prediction. In the

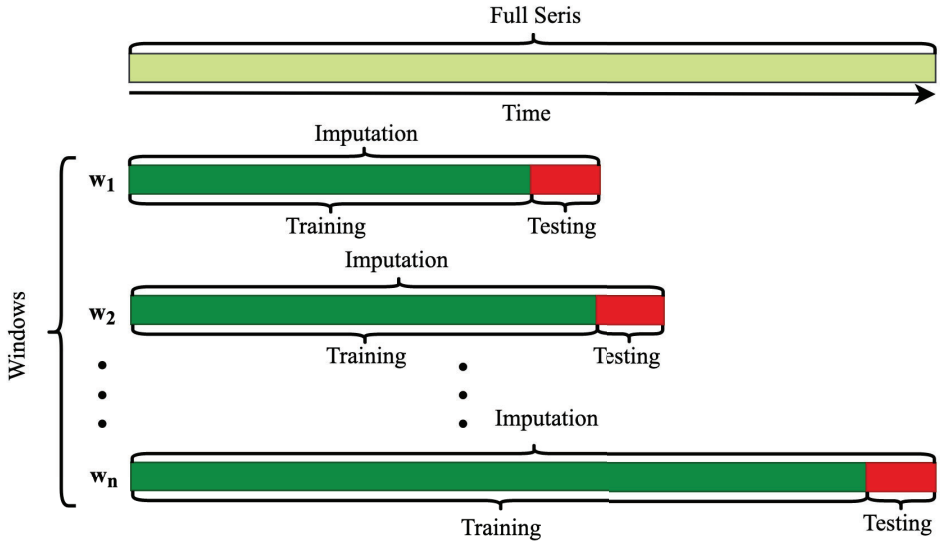


Figure 2.7 Rolling Analysis. The complete time series is divided into several overlapping windows (W). For each w first, I factorize and impute the total window and then use the first few quarters (green) for training and last quarter (red) for testing.

in-sample analysis, the prediction takes advantage of the data entries that are imputed with future information, thereby containing the leaked information not available to actual prediction and inferences. Any model trained with leaked information underestimates the generalization errors and becomes overfitting. Once I use the newest data, the model performance drops significantly. In the earlier settings, information leakage occurs in two ways: first, using matrix factorization for missing value imputation in the total data set might induce information leakage into the earlier quarters from the later quarters. Second, training the predictive model with mixed current and future quarters data provides an 'artificially' crafted good fit for model parameters [97].

To overcome the information leakage problem, I conduct the out-of-sample forecasts based on the rolling analysis, shown in Figure 2.7. First, I perform factorization and imputation to the first 60 quarters (Q1-2001 to Q4-2014) and then train the prediction model on the first 59 quarters and test the model performance only in the last quarter. I then iteratively increase the window size by one quarter,

impute data again, and then perform with the same training and test split. Instead of setting the constant window size, I increase the window size by one at each iteration to ensure the maximum utilization of the available data. Finally, I report the average result of the eight out-of-sample forecasts across firms. The iterative rolling process prevents information leakage by only employing the information in the past and present.

2.4.5 Coupled Matrix Factorization Results

Table 2.7 reports the rolling window analysis result with both MF and CMF. In the rolling window analysis, I factorize and predict for every firm at every quarter, resulting in high computational complexity. To overcome this problem in the out-of-sample analysis, I only consider the firms with more than 50 analysts and obtain a total of 117 firms. In the rolling window analysis, the performance of MF that solely relies on analysts' data decreases significantly. Except for LASSO, both MF+XGBoost and MF+SVR perform worse than the simple mean prediction. But once I integrate firm characteristics information with analysts' forecast by CMF, I observe significant performance improvements. Measured by the R^2 metric again, the three prediction algorithms based on CMF (CMF+LASSO, CMF+XGBoost, and CMF+SVR) outperform the mean prediction by 42%, 11%, and 8% respectively. Besides, the three algorithms also reduce MAPE by 60%, 34%, and 31%, respectively, in comparison with the mean prediction. Similar to in-sample prediction, both MF+XGBOOST and CMF+XGBoost outperform the simple XGBoost in all three performance measurements. Among the studied firms, simple MF based algorithms outperform the mean prediction in 67% (MF+LASSO), 47% (MF+XGBOOST) and 52% (MF+SVR) of firms, whereas the percentage of firms in which the CMF based algorithms outperform the mean prediction increase significantly, i.e., 95% (CMF + LASSO), 56% (CMF+XGBoost), and 68%(CMF+SVR).

In addition, I perform the paired two-sample t-test and evaluate whether the prediction improvement based on CMF imputed data is statistically significant compared to analysts' consensus forecasts. The results in Table 2.8 confirm that the CMF models can refine the quality of imputation and further improve firms' EPS prediction performance. The mean difference of MAPE between CMF + LASSO/XGBoost/SVR and analysts' consensus forecast (Mean) is negative and significant at least at five percent, which indicates that the forecast errors of algorithms decrease significantly. The significant mean difference between MF-LASSO and MI-LASSO indicates the proposed model's superiority over other imputation methods, i.e., MI. Moreover, I show that the mean difference of MAPE between CMF and MF using the same prediction models is also significantly negative. Once the analysts forecast data is imputed with auxiliary firm characteristics information, the accuracy of earnings prediction can be further increased. This finding shows the effectiveness of the proposed CMF and implies that the imputed values effectively absorb the firms' fundamental information that is not fully reflected by analysts' forecast data. My results are robust when using MSE and R^2.

In Table 2.7, LASSO performs better than both SVR and XGBoost when I apply coupled matrix factorization and extract high-dimensional data representations from two datasets. This finding is in contrast to the in-sample analysis on single dataset in Section 2.4.2. LASSO is a more straightforward approach than SVR and XGboost. Combined datasets unavoidably introduce data redundancy, inconsistency, noise, variances, and skewness. For predicting firms' EPS from the imputed data, LASSO mainly performs feature selection and eliminates data redundancy by enforcing ℓ_1-penalty. LASSO's simplicity allows for learning parameters with superior generalization capability, reflected in its better out-of-sample prediction performance. On the other hand, both SVR and XGBoost attempt to fit indiscriminately data and the associated noise with non-linear kernel functions and sophisticated models. When

Table 2.7 Earning Prediction Using Rolling Window Method

Method	R^2	MSE	MAPE
Mean	0.5809	0.0228	34.4740
Random-Walk	0.3146	0.2125	69.1041
XGBoost	0.3991	0.0340	27.2207
MI+LASSO	0.3955	0.0358	37.5226
MI+XGB	0.3709	0.0810	58.8945
MI+SVR	0.3934	0.0891	41.9782
MF+LASSO	0.7098	0.0151	16.5738
MF+XGBoost	0.4707	0.0378	31.1051
MF+SVR	0.3281	0.0486	35.4197
CMF+LASSO	**0.8275**	**0.0070**	**13.4689**
CMF+XGBoost	0.6451	0.0216	22.7810
CMF+SVR	0.6292	0.0223	23.9471

Note: Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term. The reported values are from test data set.

Table 2.8 Mean Difference Test of MAPE

Method	Difference
MF+LASSO - Mean	-51.92%***
MF+LASSO - MI+LASSO	-55.82%***
CMF+LASSO - Mean	-60.93%***
CMF+XGBoost - Mean	-34.92%***
CMF+SVR - Mean	-30.54%**
CMF+LASSO - MF+LASSO	-18.73%*
CMF+XGBoost - MF+XGBoost	-26.76%***
CMF+SVR - MF+SVR	-32.39%***

Note: The Difference is the percentage difference in MAPE between the groups on the earning prediction using rolling window method. *, **, and *** indicate statistical significance at the 10%, 5%, and 1% levels, respectively.

Table 2.9 Impact of Analysts' Forecast Volatility on Out-of-Sample Prediction

	Low Volatility			High Volatility		
Method	R^2	MSE	MAPE	R^2	MSE	MAPE
Mean	0.7384	0.0043	9.2907	0.4237	0.0419	59.7913
Random-Walk	0.4232	0.1041	46.1046	0.2062	0.3224	92.0814
XGBoost	0.5002	0.0146	11.5604	0.3070	0.0539	40.7714
MI+LASSO	0.5012	0.0105	10.1100	0.2912	0.0610	65.2445
MI+XGBoost	0.4581	0.0127	30.9800	0.2831	0.1501	86.6459
MI+SVR	0.5212	0.0254	14.8400	0.2666	0.1534	69.1087
MF+LASSO	0.7526	0.0030	5.5504	0.6605	0.0277	26.6010
MF+XGBoost	0.6901	0.0076	9.7446	0.2656	0.0681	51.4781
MF+SVR	0.5233	0.0122	9.9145	0.1782	0.0842	56.0801
CMF+LASSO	**0.8988**	**0.0020**	**4.8748**	**0.7574**	**0.0120**	**21.9246**
CMF+XGBoost	0.7426	0.0059	8.0746	0.5493	0.0370	37.2381
CMF+SVR	0.7185	0.0058	8.1649	0.6004	0.0663	39.4700

Note: Firms are sorted on descending order based on the average standard deviation of analysts earning forecast for a firm at a given quarter. Top 50% represents the high analysts' group and bottom 50% represents the low analysts' group. R^2, MSE, and MAPE are calculated using Equation (2.14), (2.15), and, (2.16), respectively. Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term. The reported values are from test data set.

I apply out-of-sample training, the number of available training data becomes much smaller than that of in-sample training. Data coupling adds complexity to data and causes advanced machine learning models to overfit. As a result, their performance drops significantly in out-of-sample predictions and when data coupling is used.

This finding suggests the best practice in training machine learning models: when training dataset is highly-skewed and highly-complex (the number of data points is much smaller than the dimensionality of data), simple linear models and regularization are adopted to improve the generalization capability. Once the training data size increases, which is the case reported in Tables 2.5 and 2.6, the complex models, such as XGBoost and SVR, gain their superiority over the simple LASSO

model. The relationship between the training data size and the model performance of different machine learning models is described in [132].

I also analyze the impact of the volatility in analyst forecasts on the out-of-sample prediction and report results in Table 2.9. I group the studied firms into two groups: high volatility (firms with an average standard deviation among analysts forecasts in each quarter being greater than or equal to 0.03) and low volatility (firms with an average standard deviation among analysts forecast in each quarter being less than 0.03). I find that the rolling prediction is consistent with the result discussed in section 2.4.3, i.e., the forecast volatility plays a vital role in performance improvement. For those firms with low volatility, some CMF-based algorithms barely outperform the mean prediction (CMF+LASSO = 22%, CMF+XGBOOST = 0.6%, CMF+SVR = -2%) while for high volatility, the performance improvement is nearly doubled: CMF+LASSO = 79%, CMF+XGBOOST = 30%, CMF+SVR = 41%. The benefit of including additional dataset on the high volatile firms is also evident in Table 2.9. Without the firm characteristics, the performance of MF+XGBOOST and MF+SVR deteriorates significantly for the out-of-sample prediction. Particularly, firm characteristics improve the performance by 15% (CMF+LASSO vs. MF+LASSO), 107% (CMF+XGBoost vs. MF+XGBoost), 236% (CMF+SVR vs. MF+SVR).

Overall, the findings confirm that it is important to impute missing values of individual analysts' forecast as it can reveal useful information which is orthogonal to the available analysts' forecast. Also, I show that single dataset is not sufficient and the proposed CMF technique helps to improve the imputation quality and further enhances the accuracy of EPS prediction by incorporating another dataset, i.e., firm characteristics.

2.4.6 Discussion

In this chapter, I conduct both in-sample and out-of-sample prediction. The intention of the two analyses is different. In-sample prediction mainly explains the importance of imputing missing values because the missed individual analysts' forecasts conceal useful information on firms' future earnings. Given that it analyzes the entire sample with sufficient observations, in-sample prediction shows the effects of size and volatility in analysts' forecast accuracy. It sheds light on how machine learning techniques help filter out anomaly analysts' forecasts and improve prediction accuracy.

On the contrary, out-of-sample prediction overcomes the 'information leakage' issue and is more applicable and practical. Naturally, the out-of-sample prediction performance does not match with that of in-sample prediction that takes advantage of both up-to-date and future information for missing value imputation. However, an out-of-sample approach generalizes better and is more suitable in real-world applications than the in-sample one because the available information for the out-of-sample training and inference is consistent.

There may be some concerns about the selection of firm characteristics. In this chapter, I focus on the data-driven approach, use firm characteristics, as a whole, as an example of one additional dataset for CMF imputation, and design an intelligent regularization mechanism to allow machine learning models to choose the most relevant features instead of manually analyzing each variable's impact. It might be interesting to examine whether different variables will generate different results and some are more important than others. However, given that this is not my focus and there can be numerous combinations of imputation techniques and firm characteristics, I select variables following three pieces of literature [41, 111, 120] (see Table 2.2). These papers are published in the well-acknowledged top journals in finance and accounting. I assume that they have already examined multiple variables and select those with high impacts on firms' earnings. With the help of the current

techniques, the useful information from firm characteristics is automatically and implicitly extracted from the learning and regularization procedure.

In terms of machine learning techniques, this chapter focuses on missing values and improves the quality of imputing one dataset (analysts' forecast) with another dataset (firm characteristics). It will be interesting to investigate whether and how multiple heterogeneous datasets will improve data imputation. Also, the explicit or hidden relationship and networks within and across datasets remain unexplored and leave great potentials for future research in designing new methodology and applying data imputation and machine learning in other finance areas. For example, applying tensor imputation instead of matrix imputation. Financial panel data is essentially a tensor. Tensor imputation incorporates the information from time dimension, different firms and analysts into the same framework. Another direction is to use the imputed data as the inputs to compute the optimal weights among analysts or to combine with other information, i.e., stocks prices.

2.5 Conclusion

Analysts' earnings forecast serves a vital role in the financial market. Both institutional and individual investors use information from analysts' earnings forecasts for augmenting their investment decision. A vast literature has been devoted to developing a better way to forecast firms' future earnings. In this chapter, I show the importance of imputing missing values and propose a novel way to impute missing forecasts with high quality, i.e., CMF. CMF significantly improves the quality of imputing missing analysts' forecast by incorporating additional dataset (firm characteristics). Combined with the data imputation technique, advanced machine learning algorithms provide a superior prediction of firms' earnings. Compared to the traditional mean predictions, my approach performs consistently well, less sensitive to data quantity and quality.

The findings are of great importance with several implications: first, the quality of input data, the selection of relevant datasets, and the way on how to integrate datasets are indispensable to predict a firm's future earnings. Second, the imputation of missing data and using external data, such as firm characteristics, to gauge the imputation process, are necessary. Third, even though analysts have access to the entire firm accounting information, their forecasts only utilize some aspects of such information, which leaves room for further improvement by ensemble methods. Fourth, advanced machine learning techniques have vital implications in finance research. Moreover, this chapter sheds light on a general research challenges, i.e., how to improve data quality. The improved data quality with CMF gives rise to the impressive accuracy of the downstream machine learning models for the prediction of firms' future earnings.

CHAPTER 3

PREDICTING FIRM EARNINGS USING NONLINEAR TENSOR COMPLETION ON HETEROGENEOUS DATA

3.1 Introduction

Finance studies often employ heterogeneous datasets from different sources with different structures. Some datasets are noisy, sparse, and unbalanced with missing values; some are unstructured containing text or networks; some with high frequencies, intraday and daily, whereas others with low frequency such as quarterly and annually. A simple combination of multiple datasets thus induces many challenges, including the curse of dimensionality, i.e., having more variables and features than the number of observations, neglecting interactions among data attributes, and suffering significant information loss when aggregating data from high to low dimension or from high to low frequency. In addition, conventional econometric analyses, such as regressions, require input data to be complete and balanced, which is often not true in real-world applications.

To overcome those challenges, this chapter proposes a nonlinear tensor coupling and completion framework (NLTCC). NLTCC uses (sparse) tensor (also known as multi-dimensional array) to represent input data, design machine learning techniques to disentangle the complex multi-way relationships among input data and decompose the input tensor into latent (often compact) vectors, and then employ neural networks to impute missing values and reconstruct the entire tensor for earning prediction.[1] The key novelty of NLTCC is to perform nonlinear tensor factorization on multiple datasets simultaneously and extract low-rank embedding representations. Compared to traditional linear tensor factorizations, i.e., CANDECOMP/PARAFAC (CP) [133] and Tucker decomposition [134] the neural network of NLTCC allows it to capture

[1]The basics and development related to NLTCC can be referred to section 3.3.

nonlinear interactions among datasets. As a result NLTCC extract more meaningful information to impute missing values and mitigate the curse of dimensionality.

To investigate the advantage and the usefulness of the proposed NLTCC, I use financial analysts' forecast of earnings per share (EPS) data as the experiment for two reasons: its importance in the business world and its complex data structure. EPS is the ratio of a firm's earnings (i.e., profits) to its number of common shares outstanding, it reflects a firm's performance and is one of the fundamental inputs for security pricing [23]. Market investors are highly interested in predicting a firm's future EPS to make their investment decision. Analysts' consensus (mean or median) forecast of EPS is considered as the most common and plausibly the dominant measure of market expectation. It serves as a benchmark for advanced yet complex prediction models and provides information to individual investors who do not have the necessary skills, knowledge, and time to conduct their own analysis. Studies in both finance and accounting show that analysts' earnings forecast is a better estimator of firm earnings than time series models because analysts incorporate their skills, experience, and timely information in their forecast [23, 28–30].

Besides the importance, several challenges exist in harnessing analysts' earnings forecast data. First, analysts' forecast of EPS is highly sparse and unbalanced. At a given time, multiple analysts follow one firm and generate individual reports. Analysts only track a limited set of firms and create reports for only these firms while skipping all other firms. Even for the limited group of firms, analysts occasionally miss reporting and change the firms that they follow. As a result, the EPS forecast dataset is highly sparse and has a high number of missing values in the dataset, e.g., over 99% in the original sample. Second, the EPS announcement only comes quarterly or lower frequency along with individual forecast of hundreds of analysts, resulting in more predictors (each analyst generates one predictor) than observations. Meanwhile, EPS can be affected or captured by factors other than analysts' forecasts

such as firm characteristics, stock markets, and macroeconomics. Those factors may have higher frequencies, monthly or even daily, and therefore contain more timely information.

Moreover, the EPS dataset has three dimensions, *time*, *firms* and *analysts*. The common practice to ease the data complexity is aggregating information from individual analysts to the firm-level and flatten the third-order tensor into a *firm-time* matrix, as in Figure 1.1a [31–33]. The *firm-time* matrix only preserves information from the time domain and neglects important inter-firm relationship in the spatial domain [35]. Previous studies suggest that complex relations exist among analysts, firms, and industries. Analysts from the same company or with a similar background tend to demonstrate nearly identical forecasting behaviors [37, 38]. Firms in the same industry or having similar characteristics also show similar patterns in their earnings growths [42]. Besides, analysts vary in their forecasting accuracy [36, 37] and may show a systematic bias [37, 38]. Aggregating all the efficient and inefficient analysts' forecasts indiscriminately will contaminate the information content of efficient analysts' forecasts. NLTCC, however, represents panel data in their natural multi-dimensional format, i.e., a third-order tensor (Figure 1.1b), to capture the inter-dependency of both the spatial and temporal domains and distinguish between efficient and inefficient analysts.

I show the superiority of NLTCC from three perspectives. First, I use NLTCC to impute missing values in individual analysts' forecasts based on various complementary data: firm characteristics and daily stock returns. NLTCC reduces the error of missing value imputation by 57% compared to the standard matrix factorization (MF) [117] and by 48% compared to the linear tensor factorization algorithm (CPWOPT) [135].

Second, given the high-quality imputed data, I apply advanced machine-learning (ML) techniques to capture nonlinear interaction among firms, analysts, and time,

learn powerful data representations, and then predict the next quarter's earnings. Results show that NLTCC improves the prediction accuracy by 5% for R^2, 65% for MSE, and 6% for MAPE compared with analysts' consensus forecasts. NLTCC can also distinguish influential analysts from ineffective ones and excel even more at the sectors with high heterogeneity and volatility.

Finally, to confirm the economic significance of NLTCC, I construct a long-short portfolio based on NLTCC predictions. I sort firms into decile at each fiscal quarter based on the difference between the NLTCC prediction and the consensus prediction, and then treat the top 10% as the "winner" group to take long position and the bottom 10% as the "loser" group for short position.[2] When NLTCC provides a more accurate prediction of a firm's next quarterly earnings, the positive (negative) difference between the NLTCC prediction and the consensus prediction might serve as a strong indicator for the under- (over-) estimating a firm's performance and generate a higher (lower) portfolio return. The average daily return of the long-short portfolio for three-day holding period is 0.75% (15% per month). This finding shows that NLTCC prediction is better at identifying stocks with good (bad) performance and therefore, offers more accurate foundation for profitable investment.

The main contributions of this work are as follows:

- I propose a nonlinear tensor framework (NLTCC) to address the challenges: heterogeneous data integration, missing values imputation, different data frequencies, and high dimensional complexity. This is more practical to both researchers and industry practitioners who use multiple datasets with different structures.
- I demonstrate how embedding learning with spatial-temporal regularization and state-of-the-art convolutional neural network can harness maximum information from heterogeneous data without compromising quality. By incorporating orthogonal regularization on temporal dimension and enforcing local similarity on the spatial dimension, I ensure high-quality embedding for data imputation and downstream prediction.
- The ablation studies confirm the stability of NLTCC and the benefits of integrating multiple datasets. My findings reaffirm that the success of advanced

[2]A long position is purchasing an asset with an expectation that it will increase in value. A short position is selling an asset on a borrowed account with an intention to buy it later at a lower price.

machine learning techniques depends on the quality of input data, a key area where NLTCC can contribute.

- My findings contribute to the earnings prediction literature in finance and accounting. Once NLTCC is applied to integrating multiple datasets, it enhances earnings-related information and improves prediction accuracy. This suggests that latent information such as interactions among firms and analysts captured by NLTCC is useful for firms earning prediction and nevertheless is omitted by conventional methods.

The rest of the chapter is organized as follows: Section 3.2 provides a brief introduction to the mathematical and machine learning techniques I use as the cornerstone for building the model. Section 3.3 presents the details of the NLTCC model. Section 3.4 discusses the data, experimental setup, and hyperparameters used for this study. Section 3.5 includes the detailed analysis and discussion of the experimental results. Section 3.6 discusses long-short portfolio strategies based on the proposed model. Section 3.7 provides sensitivity analysis regarding regularization and ranks, and discusses computational complexity of the model. Section 3.8 offers conclusions.

3.2 Background

In this section, I introduce notations, definitions, and machine learning techniques used throughout this chapter. I first introduce the baseline methodologies, including matrix-based factorization and missing value imputation techniques. Then I evaluate the extension of these techniques for higher-order tensors and how it can be used effectively for missing value imputation. Table 3.1 presents the symbols and notations used in this chapter.

3.2.1 PCA and Matrix Factorization

Principal Component Analysis (PCA) is widely used in finance and economics as the prominent factor analysis and dimension reduction technique. PCA uses the covariance matrix of variables to estimate the factors and their betas from the panel of the observed variables [136]. PCA linearly maps p variables into k factors where

$k \leq p$. In other words, for a data matrix $X \in \mathbb{R}^{n \times p}$, PCA decomposes it into $U \in \mathbb{R}^{n \times k}$ and $W \in \mathbb{R}^{p \times k}$ where U consists of the k principal components with the largest variance of X, and W is the factor loadings. U and W are orthogonal matrices.

Matrix completion gains significant attention for missing value imputation after the successful implementation of matrix factorization (MF) in recommender systems [117]. As discussed in Chapter 2 to estimate the missing values in $X \in \mathbb{R}^{n \times p}$, MF directly factorizes the original incomplete matrix into two low-rank factor matrices $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times p}$ with the following objective:

$$\mathcal{L} = \min_{U,V} L(X, U, V) = \min_{U,V} \|(X - U \cdot V) \odot \mathbb{1}_X\|_F \tag{3.1}$$

Herein, $\mathbb{1}_X$ is an indicator matrix, $(\mathbb{1}_X)_{i,j} = 1$, when $X_{i,j}$ has value, else $(\mathbb{1}_X)_{i,j} = 0$. MF can be applied to impute the missing values of a single firm. Both PCA and MF only handle matrix data and are inadequate for multi-way data arrays with higher order (≥ 3) [137]. To impute missing value for multiple firms in the panel data settings, MF inevitably has the problem of the undesired contamination of firm-level information when the forecasts from different firms are indiscriminately mixed as the columns or the rows of the resulting matrix.

3.2.2 Tensor Factorization and Completion

Tensor factorization/decomposition can be viewed simply as an extension of matrix factorization/PCA and low-rank approximations for higher-order data. The idea of tensor decomposition dates back to [138]. In recent years, tensor decomposition has become a prevalent dimensionality reduction technique in signal processing, computer vision, and graph analysis [139–141]. Figure 3.2 provides a graphical representation of the advantage of tensor factorization over PCA.

Table 3.1 Notations

Symbol	Description
x, x, X	a scalar, a vector, a matrix
$\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$	a tensor of order 3 and shape $d_1 \times d_2 \times d_3$
$\mathcal{X}_{i,j,k}$	tensor entry value at index (i, j, k)
$X_{i,:}, X:, j$	i-th row, j-th column of matrix X
$\mathcal{A} \odot \mathcal{B}$	element-wise tensor multiplication (Hadamard product)
$\|\mathcal{X}\|_F^2$	Frobenius norm of tensor $\mathcal{X}$
$\mathbb{1}_X$	indicator matrix, $\mathbb{1}_{X_{i,j,k}} = 1$, when $X_{i,j}$ has value, else $\mathbb{1}_{X_{i,j,k}} = 0$
$\mathbb{1}_\mathcal{X}$	indicator tensor, $\mathbb{1}_{\mathcal{X}_{i,j,k}} = 1$, when $\mathcal{X}_{i,j,k}$ has value, else $\mathbb{1}_{\mathcal{X}_{i,j,k}} = 0$
q, f, c, a, d	number of quarters, firms, characteristics, analysts, and days.

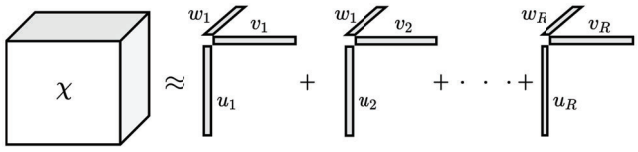


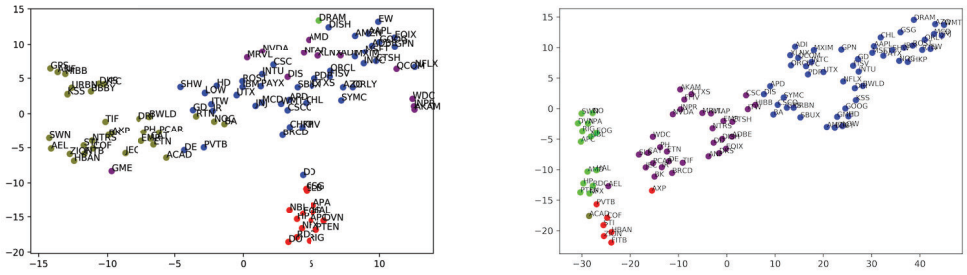
Figure 3.1 CP decomposition.

CP is one of the most popular low-rank tensor factorization models [133, 142]. Figure 3.1 shows that CP factorizes a tensor into a series of rank-one tensor and approximates the original tensor with the sum of the r rank-one component tensors. For a 3rd-order tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and a given rank r, CP factorization essentially consists of three-factor matrices $U \in \mathbb{R}^{d_1 \times r}$, $V \in \mathbb{R}^{d_2 \times r}$ and $W \in \mathbb{R}^{d_3 \times r}$ and is expressed as:

$$\mathcal{X} \approx [\![U, V, W]\!] \equiv \sum_{s=1}^{r} u_s \circ v_s \circ w_s \tag{3.2}$$

For a more comprehensive discussion on tensor decomposition, please see [139].

Tensor completion first applies factorization on a partially observed tensor $\mathcal{X}$ to learn low-rank factor-matrices from the observed entries, and then reconstructs missing entries and completes the target tensor $\hat{\mathcal{X}}$ from the factor matrices. I define



(a) PCA based embedding. **(b)** CP based embedding.

Figure 3.2 The comparison between the representation learning by PCA and CP on 100 randomly chosen firms from the service sector. Sub-Figure (a) represents spectral clustering based on the first ten principal components estimated from PCA and Sub-Figure (b) represent spectral clustering based on the latent factors learn from the CP decomposition.

the element-wise CP reconstruction as follows:

$$\hat{\mathcal{X}}_{i,j,k} = \sum_{s=1}^{r} U_{si} V_{sj} W_{sk} \tag{3.3}$$

The objective function tries to minimize the following equation:

$$f(U, V, W) = \|(\mathcal{X} - \hat{\mathcal{X}}) \odot \mathbb{1}_{\mathcal{X}}\|_F^2$$

I propose a coupled tensor method that jointly factorizes two tensors concurrently. The underlying assumption is that two Nth-order tensors $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_n}$ and $\mathcal{Y} \in \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_n}$ share at least one common dimension. Coupled tensor algorithms propagate the information from one tensor to the other by enforcing the same latent factor matrices for the shared dimensions during the factorization process [127, 143]. For two third-order tensors $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathcal{Y} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ with one common dimension d_1, the objective function for coupled tensor is to minimize the mean

square error of two tensor factorizations as follows:

$$f(U, V, W, Q, T) = \left\|\mathcal{X} - [\![U, V, W]\!]\right\|_F^2 + \lambda\left\|\mathcal{Y} - [\![U, Q, T]\!]\right\|_F^2$$

Here, λ is the hyper-parameter to adjust the relative importance between the two coupled tensors. The low-rank factor matrices are U, V, W, T, Q with U shared by both tensors. Similarly, the coupled tensor completion imputes a missing value in tensor $\mathcal{X}$ as $\hat{\mathcal{X}}_{i,j,k} = \sum_{s=1}^{r} U_{si}V_{sj}W_{sk}$ and in tensor $\mathcal{Y}$ as $\hat{\mathcal{Y}}_{i,j,k} = \sum_{s=1}^{r} U_{si}Q_{sj}T_{sk}$, where r is the tensor rank.

3.2.3 Neural Network

The neural network is a nonlinear predictive technique that gains considerable attention in recent years. Inspired by human brain, neural networks consist of highly connected neurons organized in multiple layers. The input layer takes 'raw features', and the output layer predicts the 'ultimate outcomes' with one or more hidden layers in between. A neural network can be designed either shallow or deep based on the task requirements by varying the hidden layers. An activation function in each layer adds non-linearity in the networks. Because of their flexibility in network construction and superiority as a predictive model, neural networks are widely used in computer vision, signal analysis, traffic prediction, game playing, natural language processing, and finance [97, 144–146].

I use a specific type of neural network for tensor reconstruction: Convolutional Neural Network (CNN). CNN is initially developed to overcome the overfitting problem by fully connected layers [1]. The connection between neurons in CNN resembles animals' visual cortex where individual neuron only responds for stimuli on the specific region of a receptive field [147, 148]. The hidden layers in CNN also perform convolution operation [148]. Figure 3.3 shows a typical convolutional neural

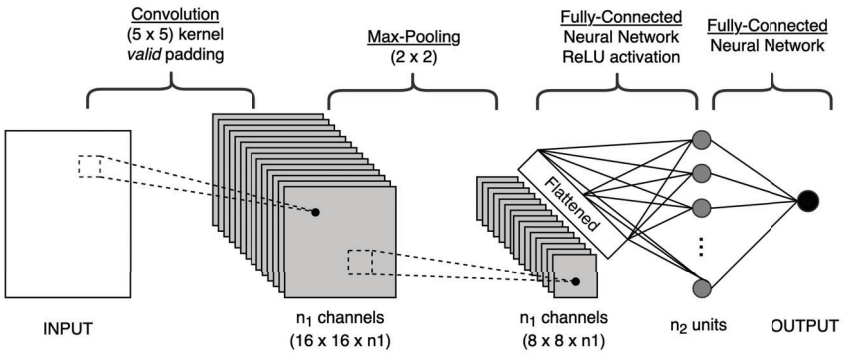


Figure 3.3 Convolutional neural network architecture.

network architecture, with each convolutional layer followed by a max-pooling layer and two fully connected layers. The filters in the convolutional layer scan input data, take a specific portion ($5 \times 5 \times 1$ for the first layer in Figure 3.3) of the tensor each time and perform a convolution operation with data within the receptive field. The subsequent pooling layer reduces dimensionality by pooling and extracting the most dominant features. Finally, the network flattens the convolution outputs and and uses it as the inputs to fully connected layers for classification or regression.

3.3 Methodology

3.3.1 The Model Architecture

The nonlinear tensor completion architecture for heterogeneous data (NLTCC) consists of four modules (a data fusion module, an embedding module, a nonlinear mapping module, and an aggregation module as shown in Figure 3.4), each of which implements one step in the forward propagation.

Data fusion module: In the data fusion module, the model takes three different tensors as input: firm characteristics $\mathcal{C} \in \mathbb{R}^{q \times f \times c}$, analysts' EPS forecasts $\mathcal{A} \in \mathbb{R}^{q \times f \times a}$, and firm daily return $\mathcal{R} \in \mathbb{R}^{f \times d'}$. One order – the number of firms (f) – is the same across the three tensors. The firm characteristics and analyst EPS forecast have the quarterly (q) dimension (the first order). The return data has a daily frequency, but

by folding the two-dimensional return tensor ($\mathcal{R} \in \mathbb{R}^{f \times d'}$) on the second-order ($d'$), I convert it into a three-dimensional tensor ($\mathcal{R} \in \mathbb{R}^{q \times f \times d}$), whereas $d = d'/q$. I partition the daily returns in quarters and represent the days within a quarter as features of that quarter, resulting in a third-order ($quarter \times firm \times day$) tensor. For example, if for firm XXX, the EPS announcement for the first quarter comes on March 31 and the second quarter comes on June 30, I consider all the trading day returns between March 31 and June 30 as features for the second quarter. Such folding operation will make the first order (quarter) of all three tensors identical. I set the maximum number of days in a quarter to 65.[3] I treat the remaining days as "missing" for quarters with less than 65 trading days and let our learned factor matrices from the model to guide the imputation on these days.

Firm characteristics are measured at different scales and skewed. Therefore, I apply data standardization on the firm characteristics tensor. Mainly, a feature-wise Yeo–Johnson transformation [149] is used to stabilize the variance and make the data distribution approximately normal.

$$x_i^{(\lambda)} = \begin{cases} \frac{[(x_i+1)^{\lambda}-1]}{\lambda} & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i+1) & \text{if } \lambda = 0, x_i \geq 0, \\ -\frac{[(-x_i+1)^{2-\lambda}-1]}{2-\lambda} & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases} \tag{3.4}$$

λ is estimated with a maximum likelihood method. I also standardize the return data. These data preprocessing procedures ensure heterogeneous data from three tensors within a standard range and help overcome unwanted effects due to large values in the imputation process.

[3]I only consider the trading days. On average, there are 21 trading days in a month.

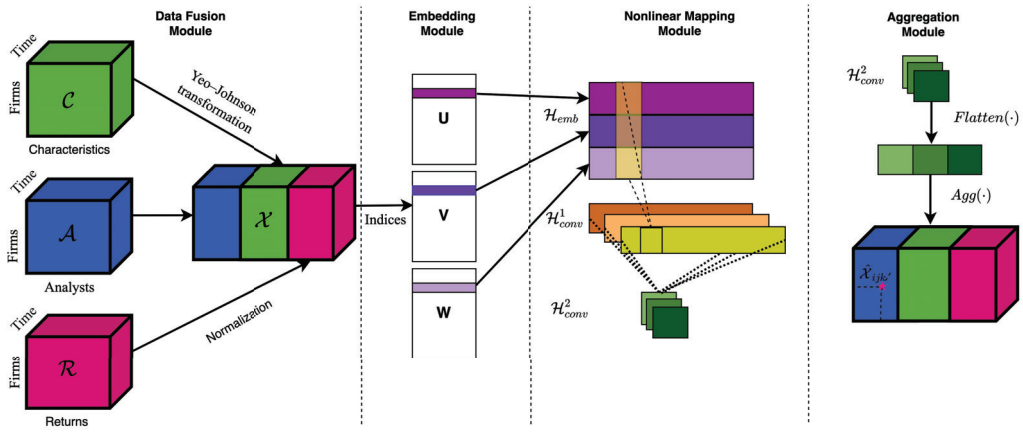


Figure 3.4 Model architecture nonlinear tensor coupling and completion framework.

All three 3rd-order standardized tensors: firm characteristics ($quarter \times firms \times characteristics$), firms' daily returns ($quarter \times firms \times daily\ returns$), and the analysts' earnings forecast ($quarter \times firms \times analysts$) have two identical dimensions and can then be concatenated along the third dimension of tensors. I treat the characteristics, returns, and analysts' earnings forecast in the concatenated tensor as the firm's unified features.

$$\mathcal{X} = \mathcal{A} \| \mathcal{C} \| \mathcal{R} \tag{3.5}$$

where $\mathcal{X} \in \mathbb{R}^{q \times f \times z}$, $z = characteristics + analysts + daily\ returns$, and $\|$ the concatenation operation. By doing so, I ensure the subsequent tensor completion uses complete knowledge from both daily return and characteristics data to regulate the imputation procedure performed on the analysts' forecast data. The tensor-based data representation allows a simple concatenation, and thus provides a convenient sampling and ensemble strategy on the firm-level data.

Embedding module: The success of any machine learning technique largely depends on learning the proper representation (embedding) of the data. The

embedding module in Figure 3.4 learns latent embeddings from multiple integrated tensors, imputes the missing values to be aligned with various data sources, and ensures the prediction of firm future earnings.

NLTCC first fuses three tensors into one third-order tensor $\mathcal{X} \in \mathbb{R}^{q \times f \times z}$ and uses it as training data to learn three factor matrices $U \in \mathbb{R}^{q \times r}, V \in \mathbb{R}^{f \times r}$ and $W \in \mathbb{R}^{z \times r}$ that consist of trainable parameters. Instead of performing a simple linear aggregation as in Eqn 3.3, I adopt the non-linear aggregation method proposed in CoSTCo [150], and design the neural networks that essentially implement the following parameterized function on the domain of tensor space, and map the indices of a tensor cell and associated embedding to the corresponding tensor element $\hat{\mathcal{X}}_{i,j,k}$:

$$\hat{\mathcal{X}}_{i,j,k} = f(i,j,k) = f(U_{i:}, V_{j:}, W_{k:}, \{\theta_i, \ldots, \theta_n\}) \tag{3.6}$$

where $1 \leq i \leq d_1$,$1 \leq j \leq d_2$, $1 \leq k \leq d_3$ and $\theta_i, \ldots, \theta_n$ are the weights of convolutional layers, dense layers, and regularization, respectively. Equation (3.6) defines an element-wise tensor completion based on the embedding matrices and the neuron weights. During network training, given an entry index i, j, k, the neural network first forward-propagates to obtain the function value and then back-propagates the loss to the embedding layers for updating the elements in U, V, W.

Nonlinear mapping module: NLTCC uses a neural network, in particular CNN, to perform element-wise tensor reconstruction and completion for the sparse input tensor. Liu et al. (2019) prove that convolutional layers are more efficient in terms of the number of parameters than the MLP-based neural networks, especially for learning high-quality nonlinear embeddings in factor matrices [150]. The subsequent reconstruction uses the embeddings to estimate a sparse tensor's unobserved entries with higher accuracy than other available linear/nonlinear tensor

completion methods. The nonlinear mapping module uses two 2-D convolutional layers with filter size of $(1, 3)$ and $(r, 1)$. The output of each convolutional layer is:

$$\begin{aligned} \mathcal{H}^1_{conv} &= \sigma(Conv(\mathcal{H}_{env} : (1, 3))) \in \mathbb{R}^{C\times 1\times 3} \\ \mathcal{H}^2_{conv} &= \sigma(Conv(\mathcal{H}^1_{conv} : (r, 1))) \in \mathbb{R}^{C\times 1\times 1} \end{aligned} \tag{3.7}$$

where C is the channel number and $\sigma(\cdot)$ is the nonlinear activation function ReLU $\sigma = max(\cdot, 0)$.

Aggregation module: The aggregation module first takes the second convolutional layer $\mathcal{H}^2_{conv}$'s output, flattens into a length-C vector and uses fully connected network layers to aggregate the vector into a scalar as the reconstructed entry i, j, k in the output tensor $\hat{\mathcal{X}}$. Because the tensor entries have both positive and negative values, the final output layer consists of a dense layer and a linear activation function.

3.3.2 Objective Function and Regularization

I first define the objective to minimize the mean squared loss between the observations in $\mathcal{X}$ and the reconstructed values in $\hat{\mathcal{X}}$. Then I back-propagate the gradient of the objective function to the embedding layer and update all affected parameters in $U_{i,:}, V_{j,:}$, and $W_{k,:}$. I define the objective function as follows:

$$\mathcal{L} = \min_{U,V,W} \|(\hat{\mathcal{X}} - \mathcal{X}) \odot \mathbb{1}_{\mathcal{X}}\|^2_F + \mathcal{R}(U, V, W, \theta) \tag{3.8}$$

$$= \min_{U,V,W} \sum_{(i,j,k,y)\in s} (f(i, j, k) - \mathcal{X}_{i,j,k})^2 + \mathcal{R}(U, V, W, \theta) \tag{3.9}$$

Here $s = \{(i, j, k, y_{ijk})|\mathbb{1}_{ijk} = 1\}$ represents the training and testing data. The Hadamard product $(\hat{\mathcal{X}} - \mathcal{X}) \odot \mathbb{1}_{\mathcal{X}}$ is applied to select the observed values because the

loss function only considers non-missing values and ignores the imputed values that have no ground truth for comparison. $\|\cdot\|_F$ is the Frobenius norm. Equation (3.9) is the loss function in the Tensor Factorization defined on the data s. The regularization term $\mathcal{L}$ in Equation (3.9) consists of the embeddings of factor matrices U, V, and W and the neural network parameters θ.

Regularization on time dimension: The main advantage of factorization based tensor completion frameworks is that the learned low-rank factor matrices contain condensed information. The success of such methods always lies in the ability to learn high-quality factor matrices. Simple concatenation among multiple tensors allows useful information to propagate from one tensor to another. However, the concatenation might also introduce noises, inconsistency, and redundancy with information. Similar to other data-driven models, tensor completion cannot eliminate all noises from the information. Explicit regularization is required to impose penalties on unwanted information (noise), redundancy, and discrepancies. One of the major goals for learning high-quality low-rank representation is that the individual features of the embedding matrix should be as different as possible. The regularization in the proposed objective function imposes the orthogonality constraints on the time dimension to eliminate the redundancy among learned features and associated undesired artifacts (high sensitivity and high variance) in the downstream learning tasks. When two features are orthogonal, they share no information, reduce redundancy and covariance, and facilitate the reliable induction of models. The orthogonal regularization ensures that $U_{:i} \perp U_{:j}$ when $U_{:i} \neq U_{:j}$. It also makes sure that all individual features in the embedding matrix are as different as possible.

The new objective function with orthogonality constraint is defined as follows:

$$\sum \|(\mathcal{X} - \hat{\mathcal{X}}) \odot \mathbb{1}_{\mathcal{X}}\|_2 + \lambda_1 \|U^\top U \odot (\mathbf{1}\mathbf{1}^\top - I)\|_{1,1} \tag{3.10}$$

where $\mathbf{1}\mathbf{1}^\top$ is the matrix of all ones. The $L_{1,1}$ norm regularizes the element-wise sparsity in a matrix. The second term of the objective penalizes the inter-dependency (correlation) among the latent features, i.e., the column vectors $U_{:i}$ and $U_{:j}$ in the quarterly matrix U. As a result, the embedding features in U are as distinct as possible. The rationale behind this is to capture as much significant information as possible from the temporal dimensions without inducing collinearity among features.

Regularization to enforce return similarity and locality: A high-quality embedding must preserve the internal data structure and locality property of the objects to be represented. In this case, the embeddings for closely related firms must be similar, while different from those unrelated firms. I use the similarity regularization in the embedding matrix to enforce this condition. For example, the embedding space of Google should be similar to that of Apple but different from that of TJ Maxx. The correlation between two firms' returns is a strong indicator of similarity and might be closely related to the explicit characteristics of firms (industry sector, firm size, revenues, and markets) and latent features to be learned during tensor completion. To build the similarity matrix, I compute the pairwise correlation between firms and build the cut-similarity matrix using a threshold function.

$$\rho(m,n) = \frac{cov(r^m, r^n)}{\sqrt{var(r^m)var(r^n)}}$$

$S_{m,n} = \rho(m,n)$ if $\rho(m,n) \geq p$ otherwise 0, where $S_{m,n}$ represents the similarity between firm m and firm n. Many values are close to zero in the original correlation and do not provide any useful information about firm similarity. Although I acknowledge the significance of negative correlations among firms, my goal is to develop an embedding matrix for firms that represents firms' similarity, and a negative correlation indicates difference. Therefore, for efficiency, I use a threshold $p = 0.30$

to make S sparse by eliminating both trivial and negative correlation. Finally, I normalize the cut similarity matrix as:

$$S \leftarrow D^{-\frac{1}{2}} S D^{-\frac{1}{2}} \tag{3.11}$$

where D is the diagonal matrix of $D = diag(d_1, ..., d_N), d_n = \sum_{l=1}^{N} S_{nl}$. Incorporating normalized-cut similarity matrix in the objective function helps enhance the total similarity between the related firms and dissimilarity between unrelated firms in the embedding matrices of firms. The new objective function with the orthogonal regularization on the time dimension and similarity on the firm dimension is defined as follows:

$$\sum \|(\mathcal{X} - \hat{\mathcal{X}}) \odot \mathbb{1}_{\mathcal{X}}\|_2 + \lambda_1 \|U^\top U \odot (\mathbf{1}\mathbf{1}^\top - I)\|_{1,1} + \lambda_2 \|S - VV^T\|_F^2 \tag{3.12}$$

3.4 Data and Experimental Details

3.4.1 Data

There are three primary sources for the data: quarterly earnings per share (EPS) and individual analysts' forecast data from Thomson Reuters I/B/E/S, firm characteristics from COMPUSTAT, and the stock information from CRSP. These data are available at https://wrds-www.wharton.upenn.edu/. I consider the period from Q1-2009 to Q4-2017.

To compare the performance of NLTCC with other benchmarks, I tested all models on a combined data set of 300 firms with 173 analysts following. In addition, I also analyzed the model performance in firm groups according to their industry sub-sector. I use the SIC (Standard Industrial Classification) to group firms into eight broad industry divisions. I perform a simple data cleaning procedure to make

Table 3.2 Data Description

Industry class	Firms	Analysts	Tensor Shape	Missing (%)	Analysts STD	EPS STD
Combined	300	173	(36, 300, 173)	96.65	2.8471	0.9232
Mining	57	152	(36, 57, 152)	88.36	3.6396	3.2796
Construction	18	81	(36, 18, 81)	87.29	2.0617	0.9084
Manufacturing	306	170	(36, 306, 170)	97.03	2.0969	0.8335
Transportation	85	62	(36, 85, 62)	88.66	3.8768	1.0187
Wholesale trade	21	20	(36, 21, 20)	78.41	2.2829	0.4074
Retail trade	84	119	(36, 84, 119)	90.07	3.2147	1.4486
Finance	192	159	(36, 192, 159)	95.50	4.2137	1.9697
Service	122	101	(36, 122, 101)	93.68	2.4531	0.9849

Note: To avoid the high computation costs of large tensors, I chose a random sample of 300 firms for the combined data. Analysts STD represents the standard deviation of individual analyst's EPS forecast of a firm at a given quarter and is then averaged over quarters for each firm. EPS STD represents the standard deviation of a firm's realized EPS over time for each firm. I then report the mean STD within the industry.

the imputation and the downstream prediction task meaningful and useful. I remove firms with time lapses in the complete time series and analysts who have predicted for less than four years or made less than 200 predictions in their entire career. This allows us to have consistent data sets for sound performance evaluation. Table 3.2 shows data description for the combined group and different industry sub-groups, including the number of firms (Firms), the number of analysts (Analysts), tensor shape, missing percentage of third-order tensors, the volatility of individual analyst's forecast (Analysts STD), and the volatility of realized EPS (EPS STD). To avoid the size effect, I scale the two standard deviations by their respective means.

Following previous literature [33,41,120], I incorporate nineteen firm characteristics and monthly and daily stock returns into the analysis. Some recording errors, e.g., negative stock price, are replaced with the absolute value. A detailed description of each characteristic variable are presented in Table 2.2.

Table 3.3 Hyperparameter Settings

Model	Hyperparameter	Values
MF	Rank (K)	5, 10, 15, 20
	Learning Rate (e)	1e-5, 1e-4, **1e-3**, 1e-2
	Regularization (λ)	1e-4, **1e-3**, 1e-2, 1e-1
CPWOPT	Rank (K)	5, 10, 15, 20
NLTCC	Rank (K)	5, **10**, 15, 20
	Learning Rate (e)	1e-5, **1e-4**, 1e-3, 1e-2
	Regularization (λ)	1e-4, **1e-3**, 1e-2, 1e-1
SVR	Epsilon (ϵ)	1e-5, **1e-4**, 1e-3, 1e-2
	Constant (C)	1, 2, 3, **5**, 10
XGBoost	Learning Rate (e)	1e-5, 1e-4, 1e-3, **1e-2**
	Regularization (λ)	1e-4, 1e-3, **1e-2**, 1e-1

Note: The final selected parameters are in bold front.

3.4.2 Experimental Setup and Hyperparameters

The NLTCC model is implemented using Keras [123] with the TensorFlow [151] at back-end. Keras is a popular deep learning framework for a cost-effective implementation of the Distributed Stochastic Gradient Descent (SGD). Keras has the built-in capability of supporting distributed machine learning models within each mini-batch and ensuring models scalability. For SGD, I use Adam. Adam is a gradient-based optimization of stochastic objective functions that continuously adjusts the learning rate based on the adaptive estimates of lower-order moments. As a result, it easily escapes saddle points while providing fast convergence [152].

I initially tune the hyperparameters (reported in Table 3.3) using a grid search on 10% validation data for the first window of the rolling window process, i.e., Q1-2009 to Q4-2015. The best parameter set (bold font in Table 3.3) is then used for the remaining windows. The initial values in the grid search are selected by following the common practice in academia and the values used in the original paper

of those models (e.g., [45,117,135,150]). For the neural network, the batch size is 128. The maximum training Epochs is set to 500 with the early stopping criteria where the program finishes training if the validation loss stops decreasing for ten Epochs. For consistency, I use the same grid configurations of tensor ranks (K), learning rate (e), and regularization hyperparameters (λ) for other benchmark models, i.e., MF and CPWOPT. Section 3.7 details the ablation studies regarding the impact of regularization and rank and presents the running time and convergence evaluations.

As discussed in Section 2.4.4 information leakage can significantly influence the outcome of any machine learning model. Therefore, to overcome the leakage problem, I conduct the experiments based on the rolling window with the window size of 28, shown in Figure 2.7. First, I perform the tensor imputation to the first 28 quarters (Q1-2009 to Q4-2015) and then train the prediction model on the first 27 quarters and test the model performance only in the last quarter. I then iteratively move the rolling window forward by one quarter, impute data again, and then perform with the same training and test split. Finally, I report the average value of the test quarters. The iterative rolling window process prevents information leakage by only employing the information in the past and present.

3.5 Results and Discussions

To investigate the superiority of NLTCC, I conduct a two-step analysis in this section. First, I evaluate multiple methods to impute missing values of individual analysts' forecasts with and without complementary datasets (firm characteristics and stock markets) and demonstrate that NLTCC consistently shows higher accuracy in the imputation than baselines. Second, I apply machine learning techniques on the NLTCC imputed data to predict the next quarter earnings (EPS) of the entire industry (prediction based on the combined group) and sector by sector (prediction based on individual industry-level subsamples).

Table 3.4 Tensor Completion Results

Additional Missing		10%			20%			40%			60%		
Total Missing		96.09%			96.52%			97.39%			98.26%		
Metric	Model/Rank	5	10	20	5	10	20	5	10	20	5	10	20
TCS	MF	0.0689	0.0553	0.0522	0.1680	0.1538	0.1532	0.3244	0.3322	0.3351	0.7575	0.7964	0.7815
	CPWOPT	**0.0495**	0.0315	0.0303	0.1371	0.1259	0.1592	0.3752	0.2579	0.4079	0.8353	0.6543	0.8942
	NLTCC(A)	0.0723	0.0430	0.0302	0.1462	0.1209	0.1120	0.2853	0.2572	0.2158	0.5702	0.5410	0.5143
	NLTCC	0.0646	**0.0292**	**0.0289**	**0.1362**	**0.1102**	**0.1009**	**0.2653**	**0.2002**	**0.1914**	**0.4011**	**0.3422**	**0.3212**
MSE	MF	0.0113	0.0092	0.0082	0.0280	0.0209	0.0208	0.0576	0.0580	0.0556	0.2067	0.1775	0.1774
	CPWOPT	**0.0107**	0.0062	0.0050	0.0156	0.0736	0.0559	0.3733	0.3573	0.6573	0.2833	0.8265	0.8308
	NLTCC(A)	0.0122	0.0104	0.0042	0.0181	0.0188	0.0106	0.0461	0.0359	0.0252	0.0769	0.0570	0.0436
	NLTCC	0.0109	**0.0056**	**0.0039**	**0.0161**	**0.0172**	**0.0101**	**0.0381**	**0.0309**	**0.0212**	**0.0569**	**0.0270**	**0.0236**

Note: MF represents matrix factorization; CPWOPT indicates CP based tensor factorization; NLTCC(A) is the proposed model that is applied only to analyst data. NLTCC is the proposed model for all available data combined, including analyst, firm characteristics, and return data. A lower value of TSC and MSE indicates better performance.

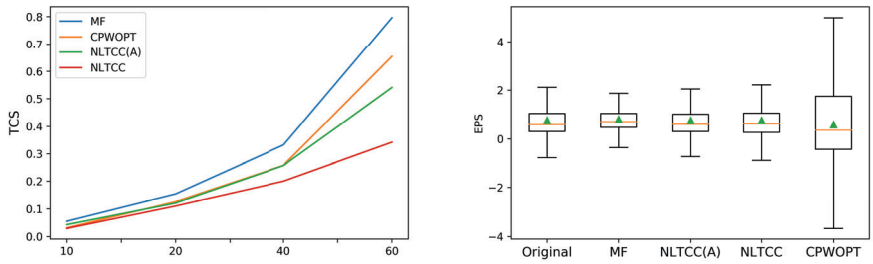
3.5.1 Tensor Completion

At first, I compare the imputation performance of the NLTCC against two well-known missing value imputation techniques; MF [117] and CP based tensor completion CPWOPT [135]. To evaluate the algorithms' robustness, I first randomly sample the available EPS forecast data by 90%, 80%, 60%, 40% to create a series of tensors with the additional missing values from an already sparse tensor and then evaluate data imputation on these tensors with an increasing level of sparsity. The performance is evaluated in terms of mean squared error (MSE) and tensor completion score (TCS) [135]. TCS is calculated as follows:

$$TCS = \frac{\|((\mathbb{1} - \mathbb{1}_{\mathcal{W}}) \odot \mathcal{X}) - ((\hat{\mathcal{X}} \odot \mathbb{1}_{\mathcal{X}}) \odot (\mathbb{1} - \mathbb{1}_{\mathcal{W}}\|}{\|(\mathbb{1} - \mathbb{1}_{\mathcal{W}}) \odot \mathcal{X}\|} \tag{3.13}$$

where, $\mathbb{1}_{\mathcal{X}}$ and $\mathbb{1}_{\mathcal{W}}$ are binary tensors with $\mathbb{1}_{\mathcal{X}_{i,j,k}} = 0$ representing original missing values, and $\mathbb{1}_{\mathcal{W}_{i,j,k}} = 0$ represents both original and randomly created additional missing values. TCS captures the relative error that is always nonnegative and stands for a good performance when the error value is small.

The tensor completion results are presented in Table 3.4. Both MF and CPWOPT only complete a single tensor; therefore, to get a fair comparison, I use two



(a) TCS (rank =10) **(b)** Data Distribution

Figure 3.5 (a) TCS at different levels of missing data. (b) The *original* data distribution and the data distributing after imputing with *MF*, *NLTCC*, and *CPWOPT*. Here, *NLTCC(A)* represent imputation with only analyst dataset, and *NLTCC* represent imputation with analyst, characteristics and return data. The green triangle represent the mean of the data.

versions of NLTCC. NLTCC(A) is the nonlinear tensor completion with only analysts' forecast data whereas NLTCC further incorporates firm characteristics and stock market information. All three tensor completion methods outperform the matrix completion by a significant margin. Table 3.4 shows that the performance difference among these models is minimal at 10% additional missing value. As the percentage of missing value increases, the superior performance of NLTCC becomes highly evident in both TCS and MSE. When 98% entries are missing, at rank 10, NLTCC outperforms MF by 57%, and CPWOPT by 48%. Even with only a single dataset, my approach NLTCC(A) outperforms MF and CPWOPT by 32% and 17%, respectively. The importance of using auxiliary information is visible in the performance difference between NLTCC(A) and NLTCC (Figure 3.5a). By fusing two additional datasets (firm characteristics and stock returns), I further improve tensor completion accuracy by 36%.

Besides, Figure 3.5b shows that the tensor completion with NLTCC also retains the original data distribution precisely comparing to MF and CPWOPT. Especially, CPWOPT inter-quartile range and whiskers are much wider than those of the original data. In comparison to NLTCC(A), NLTCC, after incorporating the other two

datasets, slightly widens the distribution of imputed data. However, this change reflects the heterogeneity among different datasets and is still much smaller than the variance incurred by CPWOPT.

3.5.2 Predicting Firms Earnings Across All Industries

To examine if NLTCC imputation can improve earnings prediction, I compare the prediction quality of the complete EPS data imputed by NLTCC against those imputed by matrix factorization and CPWOPT. I also directly compare the prediction model with the industry benchmark mean prediction.

Two prediction methods, Support Vector Regression (SVR) [46] and XGBoost [45], are applied in predicting future earnings. I integrate different imputation methods (including NLTCC) with these two prediction models. Because XGBoost has built-in sparsity awareness and handles missing values in datasets, I also use XGBoost to predict earnings from the original dataset with missing values directly. XGBoost allows us to compare the performance improvement with/without the tensor completion for data imputation. For evaluating model performance, I use three performance metrics, R^2, MSE, and MAPE as defined in Chapter 2 in Equations 2.11, 2.12, and 2.13, respectively.

Table 3.5 shows that both NLTCC+SVR and NLTCC+XGBoost outperform the mean prediction in all three performance measurements. NLTCC+XGBoost exceeds the mean prediction (Mean) in R^2 by 5%, in MSE by 65%, and in MAPE by 6%. Meanwhile, the other two imputation techniques MF and CPWOPT can not beat the mean prediction due to those models' inherent limitations (distortion to the original data distribution). Particularly, to impute data, MF collapses three-dimensional tensor into two dimensions by flattening time and firm into one dimension. Consequently, it fails to capture important information in the temporal dimension. Besides, firm-specific latent information is also lost when the data is

Table 3.5 Predicting Firms Earnings

A: Performance Evaluation			
Model	R^2	MSE	MAPE
Mean	0.9320	0.0498	55.1690
XGBoost	0.8712	0.0502	71.5423
MF+SVR	0.8451	0.0611	64.0114
MF+XGBoost	0.8526	0.0521	62.3044
CPWOPT+SVR	0.7616	0.0819	94.5147
CPWOPT+XGBoost	0.8133	0.0749	89.8420
NLTCC+SVR	0.9665	0.0195	54.1246
NLTCC+XGBoost	**0.9765**	**0.0172**	**51.6342**

B: Mean Difference Test of R^2	
Model	Difference
NLTCC+XGBoost - Mean	0.0445**
NLTCC+XGBoost - XGBoost	0.1053***

Note: This table reports the average value of three performance measures over time for each model. Mean indicates consensus forecast, MF, CPWOPT, and NLTCC indicate matrix factorization, CP based linear tensor completion, and the proposed nonlinear tensor completion model, respectively. Smaller MSE, smaller MAPE, and larger R^2 indicate better accuracy. Values in MAPE are in percentage term. **, ***, represent significant at 5%, and 1% level of significance, respectively.

flattened across firms into a matrix. Similarly, CPWOPT is a linear model and fails to capture nonlinear interactions among analysts and firms. As shown in Figure 3.5b, the wider the data distribution, the higher the variance of imputed values, the less accurate the model predictions, which explains why the prediction using CPWOPT is worst among all models.

The advantage of imputation with NLTCC and data fusion is evident in the performance comparison between XGBoost and NLTCC+XGBoost. XGBoost, with its internal data imputation alone, can not outperform the simple mean. Nevertheless,

Table 3.6 F-Test and P-Value Significance Test

Model	Adj R^2	F-Value	P-value	Significant Analysts
MF	0.647	69.02	1e-5	94
CPWOPT	0.101	5.158	3e-5	20
NLTCC	0.916	405.0	1e-5	87

with NLTCC based imputed data, the same machine learning model with similar hyper-parameters increases the prediction performance by almost 12% (R^2). Unlike XGBoost, MF+XGBoost, and CPWOPT+XGBoost, the XGBoost combined with the imputation model NLTCC (NLTCC+XGBoost) is the only one better than the mean prediction. This finding suggests that the advantage of a prediction model is conditional on the quality of input data, i.e., it is crucial to impute missing values with high quality where the NLTCC plays an important role.

To test the imputed data's quality, I perform a multivariate panel regression of a firm's actual EPS on the imputed values of analysts' forecasts. The results of the F-test and P-values reported in Table 3.6 show that the information gain of the tensor completion with NLTCC is significant. Even a simple panel regression with the imputed analysts' earnings forecast explains the 91% variance of actual earnings, whereas MF explains only 65% and CPWOPT explains only 10%. Besides, 87 analysts out of 173 are significant (effective) at 1% by NLTCC, whereas MF has 94 significant analysts. A larger R^2 with a lower number of significant features (analysts) suggests that the proposed model does not impute data indiscriminately. Instead, it distinguishes efficient analysts from inefficient ones and only keeps efficient forecasts while filtering out noises, resulting in better imputation accuracy than linear models.

3.5.3 Predicting Firms Earnings in Individual Industry Sectors

Tensor completion on all analysts' forecasts for all firms may cause some issues. First, on average, an individual analyst only focuses on eleven firms for a given period,

Table 3.7 Earnings Prediction for Industry Groups

Industry	Model	R^2	MSE	MAPE
Mining	Mean	0.8061	0.1306	103.5411
	XGBoost	0.7527	0.1578	137.0901
	NLTCC+SVR	**0.8732**	**0.0509**	51.5701
	NLTCC+XGBoost	0.8660	0.0689	**25.1641**
Construction	Mean	0.9297	0.6754	54.5707
	XGBoost	0.9482	0.5560	57.0514
	NLTCC+SVR	0.9519	0.5265	49.3445
	NLTCC+XGBoost	**0.9692**	**0.4015**	**48.5945**
Manufacturing	Mean	**0.8927**	**0.0689**	**24.4744**
	XGBoost	0.6977	0.1888	106.6645
	NLTCC+SVR	0.8918	0.0922	32.5341
	NLTCC+XGBoost	0.8854	0.1023	39.6142
Transportation and public utilities	Mean	0.7973	0.0845	24.3018
	XGBoost	0.8116	0.0666	29.1281
	NLTCCC+SVR	0.8954	0.0589	13.5846
	NLTCCC+XGBoost	**0.9352**	**0.0501**	**13.5108**
Wholesale trade	Mean	**0.9468**	**0.0111**	**13.9946**
	XGBoost	0.9329	0.0268	18.7945
	NLTCC+SVR	0.8783	0.0355	21.3545
	NLTCC+XGBoost	0.8919	0.0370	23.4045
Retail trade	Mean	0.8939	0.2120	62.3345
	XGBoost	0.8538	0.2143	69.5145
	NLTCC+SVR	0.9298	0.1944	57.3745
	NLTCC+XGBoost	**0.9465**	**0.1675**	**55.5448**
Finance, insurance and real estate	Mean	0.5221	0.4799	24.9981
	XGBoost	0.4168	0.6443	59.2015
	NLTCC+SVR	0.6998	0.3315	23.8848
	NLTCC+XGBoost	**0.7206**	**0.3293**	**20.4585**
Service	Mean	0.9005	0.1128	52.2945
	XGBoost	0.9134	0.0986	55.6301
	NLTCC+SVR	0.8935	0.1203	41.2801
	NLTCC+XGBoost	**0.9323**	**0.0951**	**40.4948**

Note: Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term. Boldface indicates the best model.

which leaves the three-dimensional tensor of all analysts and all firms over the entire history excessively sparse. I observe the EPS tensor has a high percentage of missing

data, i.e., over 99%. Second, the computational cost of tensor completion increases exponentially with the tensor order and each order size. Third, NLTCC attempts to learn the embedding for time, firms, and analysts from the relevant time, firms, and analysts. Previous literature suggests that firms in different industries likely are experiencing different business cycles. For example, firms in the manufacturing industry are expanding with high growth rates during a given time, whereas firms in the service industry are not, and vise versa. Therefore, combining different types of firms in the same imputation tensor introduces data inconsistency and adds complexity in learning.

To overcome the first two problems, I have to perform a rigorous filtering procedure to select only a small subset of firms that ensure a high number of valid observations with the small tensor shape. However, it is still not sufficient to tackle the last problem of data inconsistency. Therefore, I group firms into their related industry sectors following the Standard Industrial Classification (SIC) codes and perform industry-wise tensor completion. It ensures the consistency of information among firms that share the same earning trend. The strategy also helps to overcome the curse of dimensionality and reduces the percentage of missing values. Table 3.7 reports the result of predicting the next quarter earnings for firms in each industry group. There are several interesting findings as follows.

First, the performance of NLTCC is relatively stable and consistent across groups. Except for the manufacturing and wholesale trade groups, the imputation based models (NLTCC+SVR and NLTCC+XGBoost) outperform the mean and simple XGBoost predictions in all three performance metrics for all other groups. For the manufacturing industry, the mean prediction is better than NLTCC+SVR and NLTCC+XGBoost. However the performance difference is negligible, as the R^2 is 0.8927 (Mean), 0.8918 (NLTCC+SVR), and 0.8854 (NLTCC+XGBoost).

Second, the performance improvement of NLTCC over the mean prediction is more prominent for groups with high analysts' dispersion. For example, the analysts' dispersion of the mining, transportation, and finance industries is 3.64, 3.87, and 4.21, respectively (based on the Analysts STD in Table 3.2), all of which are higher than others. Those industries enjoy more improvement from the proposed prediction model than others. In terms of R^2, the gain is 8% for mining, 17% for transportation, and 38% for finance. This result can be explained with the central limit theorem and confidence interval. It is challenging to surpass the simple yet, effective mean prediction because when the data volatility is low, the mean prediction has a narrow confidence interval to encompass the actual earnings. By contrast, for industries with a high analyst dispersion because of insufficient information or information being too complicated, individual analysts have different interpretations and, therefore, are less likely to agree with others. Under these circumstances, NLTCC helps select efficient information from analysts without introducing too much noise. As a result, it improves the quality of imputed data and further enhances the prediction accuracy by machine-learning techniques.

Third, the performance improvement of tensor completion is positively correlated to firms' realized EPS volatility. For firms with less variation in their actual earnings, the mean algorithm attains perfect prediction, as shown in Figure 3.11, and the sophisticated ML models with data imputation hardly exceed the mean prediction. The wholesale trade has the lowest earnings variance among all groups (0.4047) (The last column in Table 3.2), and as a result, they have an excellent mean prediction ($R^2 = 0.95$). It is also true for other stable industry groups, such as construction and manufacturing. These industries are mature, and in the last few years, companies in these sectors experience relatively steady growth. When the growth rate is stable, analysts can easily estimate future earnings at a high accuracy level. The value added by any sophisticated prediction model for these categories is marginal. These

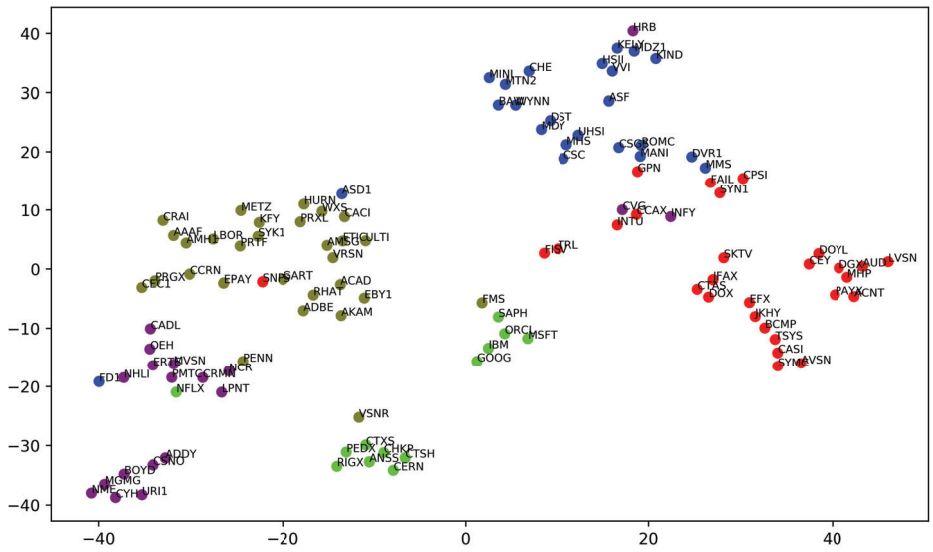


Figure 3.6 The t-distributed stochastic neighbor embedding (t-SNE) of the spectral clustering based on firm learned latent factors. All the firms in the service industry are grouped into five clusters. The adopted tensor completion model learns meaningful embeddings for firms according to their size, service type, and the client groups they serve. Both color and space represent differences in embedding.

experiments suggest that the advanced imputation model (such as NLTCC) and machine learning models (such as SVR and XGBoost) deliver the benefits to firms whose standard deviation of earnings is over 1.00 and nevertheless, have marginal improvements for firms with a standard deviation of less than 0.80. The area with variance ranging between 0.80 to 1.00 is somewhat mixed where the gain from NLTCC is reasonable, yet less convincing than that in the highly volatile section

Forth, with the high-quality data preprocessed by NLTCC imputation, two prediction models (NLTCC+XGBoost and NLTCC+SVR) have a small performance difference. The finding confirms that data preprocessing based on data fusion and imputation plays a vital role in data analysis, reveals the majority of knowledge from data, and eases the selection of downstream prediction models. In addition, the performance of XGBoost without explicit data imputation fluctuates significantly across all sectors. For some industries, such as construction, transportation, public utilities, and service, XGBoost is better than the mean prediction but worse than the mean prediction for all other sectors. Compared with my proposed

Table 3.8 Data Description of Manufacturing Sub-Sectors

Sub-sector	Firms	Analysts	Missing (%)	Analysts STD	EPS STD
Food	21	55	81.38	3.1452	1.0493
Textile	32	75	91.28	2.6940	0.9587
Papers	12	29	73.49	2.3089	1.9054
Chemicals	57	146	94.55	2.7495	1.6615
Glass and metals	60	140	94.08	2.5267	0.7339
Computers	72	89	93.22	3.0194	1.4772
Automobile	56	86	92.75	2.2095	1.2185

Note: Analysts STD represents the standard deviation of individual analyst's EPS forecast of a firm at a given quarter and is then averaged over quarters for each firm. EPS STD represents the standard deviation of a firm's realized EPS over time for each firm. I then report the mean STD within an industry.

NLTCC+XGBoost, the simple XGBoost is inferior in all industries except the wholesale trade. This observation signifies the importance of data imputation effort and nonlinear tensor completion model.

Fifth, as I claim in Section 3.3, the factor matrices obtained from NLTCC factorization contain meaningful embedding vectors for the respective dimensions, i.e., quarter, firm, and analyst. High-quality latent embedding of NLTCC will greatly benefit the downstream machine learning prediction models. Figure 3.6 provides a useful demonstration of this. I use the t-distributed stochastic neighbor embedding (t-SNE) to visualize the learned latent factor and their embedding space. For brevity, I use the clustering result of the service sector as an example. NLTCC learns multi-dimensional embedding for firms based on their size, service type, and the client groups they serve. For example, Citrix Systems, ANSYS, Check Point Software Technologies are IT companies and belong to the same group as Microsoft, Google, IBM, Oracle, and SAP, but their embedding space in the lower green group is slightly different from the large IT firms in the upper green group. High-quality embedding offers better grouping and classification accuracy and provides useful information

Table 3.9 Sub-Sector within Manufacturing Industry

Industry	Model	R^2	MSE	MAPE
Food	Mean	0.8585	0.0471	49.9463
	XGBoost	0.8979	0.0435	50.7263
	NLTCC+SVR	0.9243	0.0237	47.7413
	NLTCC+XGBoost	0.8719	0.0473	42.1763
Textile	Mean	0.8613	0.0483	66.1538
	XGBoost	0.8330	0.0443	61.9275
	NLTCC+SVR	0.8887	0.0225	51.9138
	NLTCC+XGBoost	0.8513	0.0410	65.4938
Paper	Mean	0.8570	0.0132	21.7800
	XGBoost	0.8189	0.0202	18.4225
	NLTCC+SVR	0.8746	0.0100	18.7638
	NLTCC+XGBoost	0.8760	0.0101	17.3750
Chemicals	Mean	0.8204	0.7047	0.6498
	XGBoost	0.8157	0.2947	0.7465
	NLTCC+SVR	0.8601	0.1502	0.5068
	NLTCC+XGBoost	0.8540	0.1598	0.5221
Glass and metals	Mean	0.9296	0.0219	35.5488
	XGBoost	0.8041	0.0615	80.6163
	NLTCCC+SVR	0.9038	0.0420	59.9600
	NLTCCC+XGBoost	0.8982	0.0424	58.4575
Computers and electronics	Mean	0.9062	0.0389	30.5213
	XGBoost	0.8756	0.0350	52.0375
	NLTCC+SVR	0.9137	0.0318	28.5525
	NLTCC+XGBoost	0.9283	0.0279	26.2750
Automobile and air-crafts	Mean	0.9078	0.0536	29.0275
	XGBoost	0.8517	0.0754	44.7750
	NLTCC+SVR	0.9386	0.0472	24.7925
	NLTCC+XGBoost	0.9259	0.0512	26.5388

Note: Mean indicates consensus forecast, NLTCC indicates the proposed model. Smaller MSE, MAPE and larger R^2 indicate better accuracy. Values in MAPE are in percentage term.

about firms' inherent structure and hidden representations that are otherwise difficult to capture with simple data mining techniques.

3.5.4 Robustness Test

To check the robustness of my findings within industry groups, I further investigate one industry at a micro-level. Compared to other sectors, manufacturing initially comprises of a large number of firms. I hypothesize that the under-performance of NLTCC in this sector results from heterogeneous information in the firm dimension. I further divide all manufacturing firms into seven subgroups based on their SIC codes, i.e., food, textile, paper, chemicals, glass and metals, computers and electronics, and automobile and air-crafts. Table 3.8 presents the details of sub-sectors. The prediction results with tensor completion for each sub-sector are reported in Table 3.9. The results of the sub-sectors support my initial hypothesis: once firms are grouped into a more refined category, the tensor completion and subsequent prediction models on sub-sectors perform much better than the simple mean method. NLTCC takes advantage of consistent knowledge in sub-sectors to improve performance over the initial grouping with a coarse granularity of industry sectors. It also reaffirms the understanding that the prediction with complete tensor has a much higher prediction accuracy than the mean method and XGBoost for the cases in which the analysts' forecast variance and the standard deviation of actual earnings are high.

3.6 Portfolio Analysis

3.6.1 Portfolio Based on the Difference Between NLTCC and Mean Prediction

To evaluate the proposed NLTCC model's investment feasibility, I build portfolio strategies based on the assumption that bias exists in analysts earning forecast. I hypothesis that over-optimistic analysts' forecasts for a firm will lead the current market price of its share higher than the true value, and once actual earnings announcement comes, its share price will go down to reflect the fair price; and on the other hand, over-pessimistic forecasts for a firm will lead the current price of its share to be lower than the true value, and once actual earnings announcement comes,

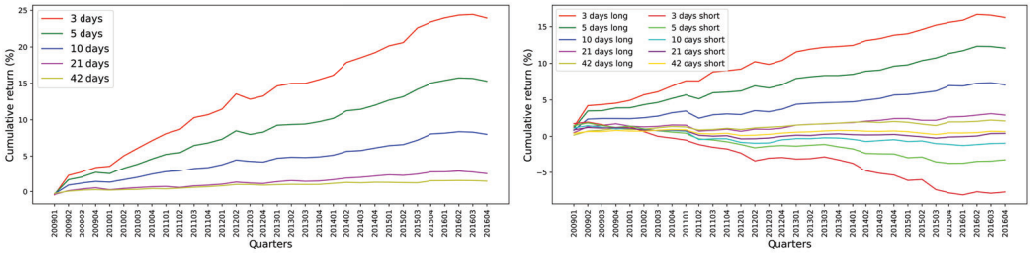
Table 3.10 Returns on Long-Short Portfolio Constructed Based on the Difference Between NLTCC and Mean Prediction

Holding periods	3 days	5 days	10 days	21 days	42 days
Average (%)	0.7487	0.4777	0.2500	0.0780	0.0451
STD (%)	0.7499	0.5039	0.3230	0.1882	0.1086
Sharpe Ratio	15.8474	15.0489	12.2834	6.5806	6.5971

Note: Average is the daily average return and STD is the daily standard deviation of return of long-short portfolios. I adopt the formula from Morningstar, Inc. to calculate the yearly Sharpe ratio from daily returns. $SR = \frac{\sum_{d=1}^{D} PR_d}{\sigma_{PR_d}} \times \sqrt{252}$; where PR_d is daily portfolio return and D is the total number for days.

its share price will go up to reflect the fair price. If these upward or downward biases in analysts' forecasts can be detected with high confidence, investors can earn a high return by designing their portfolios in accordance to the predicted over-estimation or under-estimations. NLTCC prediction is more accurate than the mean EPS forecasts and provides an excellent indicator to these earning expectation bias. To confirm the superiority of the NLTCC prediction, I construct a long-short portfolio based on the difference between NLTCC predicted EPS and the mean consensus forecast. At each quarter, I first calculate the difference between NLTCC and the mean estimates for each firm and scale the difference by the respective share price on the prior day $t-1$ ($\Delta = \frac{NLTCC_{iq} - Mean_{iq}}{P_{t-1}}$) where t stands for the date of the earnings announcement for each firm at a given quarter. I take a long position on the top 10% firms with the most positive difference and take a short position on the bottom 10% stocks with the most negative difference. I hold the position for three trading days (t to $t+2$), five trading days (t to $t+4$), 10 trading days (t to $t+9$), 21 trading days (t to $t+20$), and 42 trading days (t to $t+41$), respectively, where 21 (42) trading days is equivalent to one (two) month holding period.

Machine learning based NLTCC model provides a low bias estimation of a firm's future earnings [153] whereas analyst forecast tends to have a high bias [37,38,40,154,



(a) Long-short portfolio. **(b)** Long side and short side separately.

Figure 3.7 Cumulative returns on long-short portfolio build based on the difference between the EPS prediction by NLTCC and Mean. Subfigure 3.7a is the long-short portfolio as a whole, and subfigure 3.7b is the individual component of the long-short portfolio.

155]. The positive (negative) difference indicates that those firms are under- (over-) estimated by analysts. During the time window over earnings announcement, the stock price increases (decreases) correspondingly when a firm's realized earning beats (misses) the analysts' consensus forecasts. Table 3.10 shows that the NLTCC portfolio earns a positive return with impressive Sharpe ratios in all five holding periods. The Sharpe ratio is the highest for the three-day holding period followed by the five-day holding period.

One interesting finding is that as the holding period increases, the average daily return decreases. This is intuitive: upon the earnings announcement, investors respond to the new information, resulting in volatility in stock price. Afterwards, the positive and negative information is gradually reflected by stock prices, and the magnitude of price changes (returns) diminishes. My results are consistent with the literature that the financial market is efficient to some extent and suggests that a short-term investment strategy is more profitable. Figure 3.7a depicts the cumulative return from the long-short portfolios of different holding periods and shows a similar pattern. The cumulative return declines over a more extended holding period, i.e., the cumulative return earned from a three-day holding is 25% compared to 2% for forty-two days.

I are also interested in understanding whether the outperformance mainly comes from the long-position, short-position, or both. I construct the long-side and the short-side portfolio separately. The results in Figure 3.7b indicate that the long position contributes more to superior performance than the short position does. For example, the average daily return during the three-day earning announcement window for a long position is 0.5075 (68%), whereas that for a short position is only 0.2411 (32%). For a 42-day holding period, the average daily of a short position is even negative. One plausible reason is that the U.S. stock markets' restriction makes short selling more difficult than long positions and incurs higher costs and extra requirements. Consequently, investors choose not to take a short position even though there exists a profitable opportunity.

3.6.2 Continuous Trading Strategy

In Subsection 3.6.1, I construct a long-short portfolio strategy and show that without considering any transaction cost and practicability, the proposed model offers high profit potential by identifying "winner" and "loser" stocks. Nevertheless, the different announcement dates for each stock make the implementation of this portfolio strategy hardly practical.

Therefore, I propose a practical trading strategy to hybrid the S&P500 index and the "winner" or "loser" stocks identified by the NLTCC prediction. I initially hold the S&P500 index ETF until appropriate "winner" ("loser") stocks are identified, and then take long (short) positions on selected stocks during the designated holding window around earnings announcement dates. At the end of the assigned holding window, I will cash out the positions, reinvest on S&P500 and repeat the same process. One day before each announcement day (t-1), I categorize stocks into "winner", "loser", and the rest, based on the difference between NLTCC prediction and mean forecast scaled by the share price at t-1. When the scaled difference of a stock is

higher than 0.5%, it is considered a "winner", and on the other hand, a 'loser" when the difference is less than -0.5%. The daily return for each stock is calculated as:

$$r_{it} = \begin{cases} log(P_t + dividend) - log(P_{t-1}), & \text{if } \Delta > 0.005 \\ log(P_{t-1}) - log(P_t + dividend), & \text{if } \Delta < -0.005 \\ NA, & \text{otherwise.} \end{cases} \tag{3.14}$$

where $\Delta = \frac{NLTCC_{iq} - Mean_{iq}}{P_{t-1}}$ is the scaled difference of each stock. I calculate the portfolio return as follows:

$$PR_t = \begin{cases} r_{S\&P,t} & \text{if } \forall r_{it} = NA \\ \frac{\sum_i^p r_{it}}{n}, & \text{otherwise.} \end{cases}$$

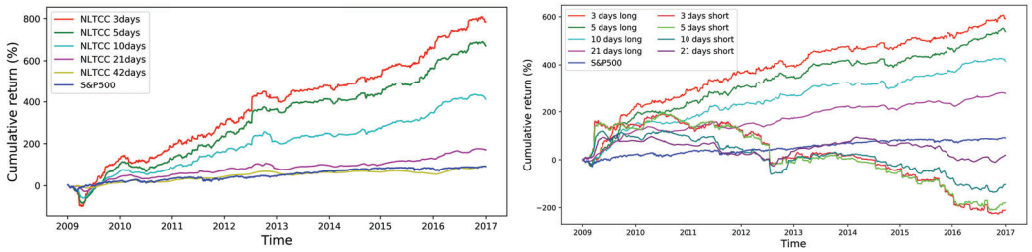
where p is the number of stocks in the "winner" or "loser" group based on Equation (3.14). If multiple stocks meet the investment design for any given day, I hold an equally weighted position across all of them. No transaction cost is considered: whenever a new stock is added, the old portfolio's weights are rebalanced to accommodate the new stock(s) with the same weights for all stocks. For example, the previous portfolio has four stocks with equal weights of 25% each. After identifying a new stock, the portfolio consists of five stocks with equal weights of 20% each.

Table 3.11 reports the results of the continuous portfolio for three-, five-, ten-, 21-, and 42-day holding period, respectively. I compare my mixed portfolio returns to S&P500 return. The mixed portfolios for the five holding periods generate much higher returns than the simple holding on S&P500. Similar to Section 3.6.1, the continuous portfolio also demonstrates that the majority of the above-average return comes from the information advantage on and right after the announcement day.

Table 3.11 Continuous Portfolio

Holding periods	S&P500	3 days	5 days	10 days	21 days	42 days
Average (%)	0.0451	0.3893	0.3334	0.2049	0.0845	0.0459
STD (%)	1.0996	2.3019	1.5496	0.8790	0.4245	0.2980
Sharpe Ratio	0.6507	2.6846	3.4159	3.7015	3.1602	2.4470

Note: Average is the daily average return and STD is the standard deviation of daily return of long-short portfolios over the time period. Average and STD are daily, where Sharpe ratio is annualized. I adopt the formula from Morningstar, Inc. to calculate the yearly Sharpe ratio from daily returns. $SR = \frac{\sum_{d=1}^{D} PR_d}{\sigma_{PR_d}} \times \sqrt{252}$; where PR_d is daily portfolio return and D is the total number for days.



(a) Long-shot portfolio. **(b)** Long side and short side separately.

Figure 3.8 Cumulative returns on long-short portfolio hybrid with S&P500. In this trading strategy, I hold S&P500 unless I find any undervalued or overvalued security in the market. I take long positing in the under-estimated security and short positing in the over-estimated security. Subfigure 3.8a is the long-short portfolio as a whole, and subfigure 3.8b is the individual component of the long-short portfolio.

Holding stocks for a longer period does not necessarily provide more additional benefits. Instead, the return for long holding periods shows a mean reversal trend. For 42 days holding period, the portfolio return is almost the same as S&P500.

Figure 3.8 depicts the cumulative returns over the studied period from the hybrid long-short portfolios as a whole (Figure 3.8a) and long side and short side separately (Figure 3.8b). Initially, the hybrid portfolio earns a lower return than S&P500, and the low return is mainly associated with the negative return in the short-side of the portfolio. Figure 3.8b shows that, from the beginning of 2009 to almost the end of 2012, the short portfolio return is above zero, indicating that I receive a loss in short-selling. This is intuitive: during this period, because the market

recovered fast, the stock price had a significant upward trend. Nevertheless, over time, both the short and long positions of the portfolio become profitable and earn significantly higher returns than S&P500. This also explains why the contribution of the short-side is lower than that of the long-side in the long-short portfolio on average. In the studied period (2009-2018), the US market experienced high growth, a recovery from the great financial crisis, and the bull market afterward. As a result, the long positions in any stocks became more profitable compared to the short positions.

3.7 Ablation Study

3.7.1 Impact of Regularization and Ranks

In this subsection, I analyze the impact of combining multiple data sets, regularization, and different ranks on the earnings prediction performance and evaluate the impact of hyperparameters on the robustness of data integration results. A variety of financial datasets are related to firms and their quarterly earnings: stock prices capture the market responses to a firm's information, accounting data document a firm's fundamental information, finance news reports any major event of a company, and analysts consensus forecast reveals the general market expectation of firms' future earnings. These datasets offer different views concerning the same underlying firm, and any single source alone might not provide the complete representation of a firm. NLTCC delivers an efficient approach to combine multiple datasets and build a comprehensive model for earning prediction. I evaluate several versions of the proposed NLTCC model on the 300 firms analyzed in Section 3.5.2. These models include only financial analysts data (NLTCC(A)), financial analysts and firm characteristics data (NLTCC(A+F)), financial analysts, firm characteristics, and return data with no regularization (No_reg), all three datasets with the orthogonal regularization on the temporal dimension (Time_reg in Equation (3.12)), all three datasets with similarity clustering regularization on the firm dimensions (Sim_reg),

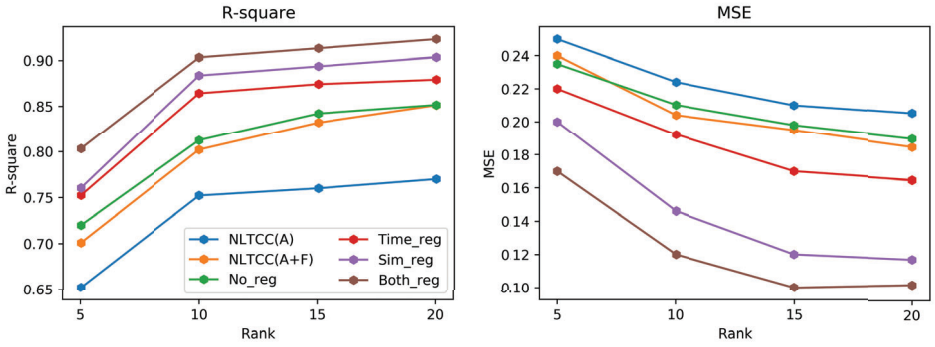


Figure 3.9 R^2 (left panel) and MSE (right panel) for different models at different ranks. *Mean* represents consensus forecast; *NLTCC(A)* is the proposed model with only analysts forecast data; *NLTCC(A+F)* with both analysts forecasts and firm characteristics data; *No_reg* with analysts forecast, firm characteristics and return data without any regularization; *Time_reg* with all three datasets and the orthogonal regularization on the time dimension; *Sim_reg* with all three datasets and the similarity regularization on the firm dimension; and finally, *Both_reg* with all three datasets and two regularizations on both the time and firm dimensions.

and all three datasets with both orthogonal and similarity clustering regularization (Both_reg).

Figure 3.9 signifies the importance of using auxiliary information from multiple datasets and applying regularization on earning prediction. The ML-based prediction with only the sparse and noisy EPS forecast data, even enhanced by advanced data imputation method and sophisticated predictive models (i.e., XGBoost, SVR), cannot beat the simple mean prediction. Combining firm characteristics and return information improves prediction performance, but it is still inferior to the mean forecast due to the discrepancy, noise, and inconsistent quality in heterogeneous datasets. Figure 3.9 also demonstrates that the performance improvement of adding third datasets, i.e., stock return data, along with analyst forecast and firm characteristics, is insignificant. This observation confirms that simple data integration can not maximize data value, and the reconciliation on different datasets is indispensable. Once I introduce regularization on the time and firm dimensions to reconcile the data heterogeneity and discrepancy, the performance improvement

becomes significant. Notably, the similarity clustering regularization on the firm dimension results in the highest reduction in forecasting error by 30%. Compared to those models without regularization, the orthogonal regularization on the time dimension reduces the prediction error by 9%, and the regularizations on both firm and time dimensions further reduce the prediction error by 43%.

Similar to the low-rank configuration on many representation learnings, the choice of rank has a significant impact on NLTCC's performance. Figure 3.9 illustrates the effects of different ranks on the earning prediction of different versions of NLTCC. With a small rank (= 5), all models' performance is low. Once the rank increases to ten, the performance improves significantly. Nearly all versions of the proposed model reach the peak performance with a rank between 15 and 20. Simultaneously, a higher rank incurs more computational complexity. In addition, once the performance peaks, further increasing rank has no benefit, which is consistent across models.

Overall, these findings signify that advanced ML methods can be applied to finance applications to reap the benefits of heterogeneous information from multiple datasets. However, before using these techniques, we must carefully consider the restriction and limitations of these techniques. A simple concatenation of heterogeneous data might not work because of the different measurements scales, noise, and inconsistency. Regularization enforces consistency in the data structure, separates the unwanted noise from the signals, and induces meaningful embeddings for the downstream modeling and predictions.

3.7.2 Impact of EPS Volatility on Prediction

As discussed in Section 3.5.3 that the performance improvement by NLTCC is marginal for firms that have low dispersion in available analyst forecast or firms with very stable historical earnings. I plot the EPS data distributions and demonstrate why it is difficult for advanced machine learning models to outperform the consensus

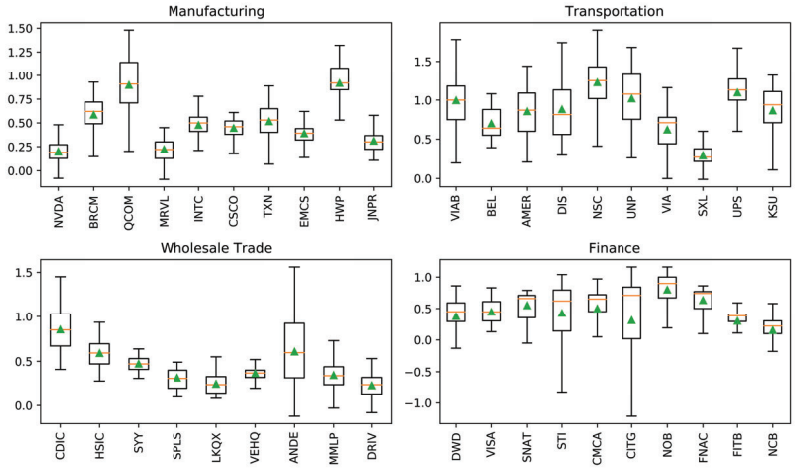


Figure 3.10 The EPS data distribution of the top ten most frequently analyzed firms in four major sectors.

forecast regarding some firms. Figure 3.10 shows the EPS data distribution of the top ten most frequently analyzed firms in four major sectors, i.e., manufacture, transportation, wholesale trade, and finance. For firms in *Manufacturing* and *Wholesale trade* sectors, the median of analysts forecasts almost equals the mean value that serves as a simple yet highly accurate prediction model for earnings. In contrast, for firms in the *Finance* industry, the analysts forecast distribution is highly spread, and the mean deviates from the median, which indicates the potential benefits of the ML-based prediction.

Figure 3.11 represents the actual EPS along with the analysts forecasted EPS for each quarter of some large companies i.e., *Nvidia* (manufacturing), *CSX* (transportation), *Henry Schein* (wholesale trade), *Morgan Stanley* (finance), and *Netflix* (service). It is challenging to outperform the mean prediction for firms whose actual EPS almost always aligns well with the analysts' EPS forecast, e.g., Nvidia or Henry Schein. In contrast, incorporating auxiliary information (firm characteristics and return) improves the prediction performance for firms whose analysts' forecasts deviate significantly from actual EPS, e.g., Morgan Stanley or Netflix.

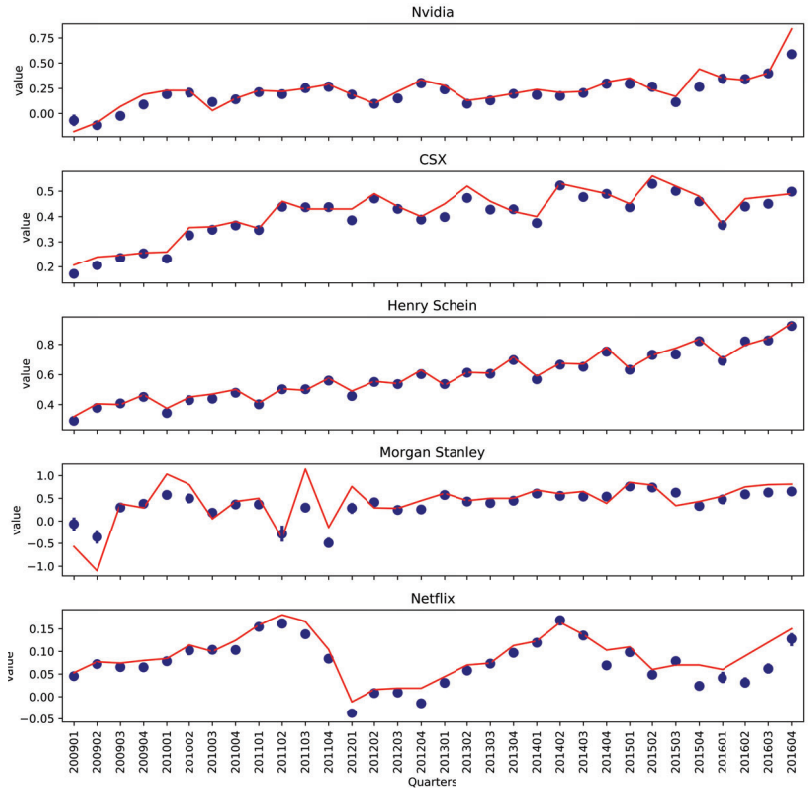
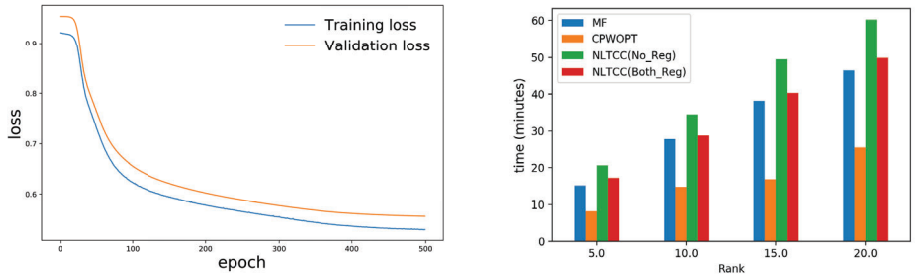


Figure 3.11 The actual EPS (red line) along with the analysts forecasted EPS (marked with blue dots) for each quarter of several large companies from different sectors.

3.7.3 Running Time Comparison and Convergence

Machine learning models often computationally expensive for large training datasets. However, the convolutional neural network (CNN) used for building NLTCC is renowned for faster convergence. Unlike traditional tensor algorithms, CNN does not use heavy operation steps, such as the Kronecker product or Gram matrix [150]. As a result, NLTCC does not incur steep computation/memory costs in the training process. Figure 3.12b presents the running time of two benchmark machine learning models and NLTCC for different tensor ranks. Among these models, CPWOPT is the most computationally efficient. It is expected because CPWOPT performs a simple CP decomposition for latent representation learning rather than involving a complex deep learning model. Another observation is that Both MF and NLTCC



(a) Convergence plot **(b)** Running time comparison

Figure 3.12 (a) Convergence plot of NLTCC (b) Running time of different tensor completion algorithms at different ranks. $NLTCC(No\_Reg)$ is with all three data set but without any regularization, $NLTCC(Both\_Reg)$ is with both orthogonality and similarity regularization.

incur similar computation time while MF is slightly faster than NLTCC. It also shows that regularization reduces the computation time for NLTCC because regularization facilitates a rapid learning process. With a Linux system with 16GB GPU memory, 3854 CUDA cores, and 40 CPU cores, it took NLTCC almost 50 minutes to converge for rank 15. In addition, the mini-batch gradient descent based on Keras's deep learning framework allows the model to be trained without explicitly storing the whole tensor. In [150], Liu et al. proved that the convolutional neural network-based tensor factorization has a fast convergence rate and a tight bound on generalization error [156] by showing the connection of their model with one shallow but efficient neural net. Figure 3.12a shows the convergence plot in training the NLTCC model. With a learning rate of 0.0001, the model converges around about 400 epochs.

3.8 Conclusion

This chapter presents a convolutional neural network-based nonlinear tensor coupling and completion framework, named NLTCC. NLTCC integrates firm-level characteristics, market return data, and analysts' earnings forecast to overcome the data quality problems of noise, sparsity, and heterogeneity. I first apply NLTCC to impute missing values in individual analyst's forecasts and then predict the firm's next

quarter earnings with popular machine-learning algorithms (SVR and XGBoost). The accuracy of NLTCC-based earnings prediction is significantly higher than the analysts' consensus forecasts. The majority of prediction improvement comes from the superior performance of NLTCC for data pre-processing and imputations, data integration and noise removal, and the discrepancy reconciliations among heterogeneous datasets. Extensive experiments on industry sectors and subsectors confirm the effectiveness of NLTCC model. Particularly, NLTCC works exceptionally well in industry sectors with high variance in analysts' forecasts, indicating a volatile market and involving complex information for forecasts and predictions. The findings imply the importance of data quality (data quality is the value added by missing values imputation) and the advantage of NLTCC. The backtesting shows that the long-short portfolio based on NLTCC prediction generates much higher returns than that of the S&P500 index and the portfolio strategy based on analysts' consensus forecast. The experiments also reveal that ML models have limitations in some circumstances. A straightforward technique, such as the mean (analysts' consensus) forecast, is effective in those stable sectors. Accordingly, I propose a hybrid deployment strategy for industry practitioners to reap maximum benefits: using mean prediction for firms with ≤ 0.8 earnings volatility and switching to advanced techniques for firms with higher earning volatility.

CHAPTER 4

LATENT FACTOR MODEL FOR ASSET PRICING

4.1 Introduction

Recent advancements in machine learning (ML), and the success stories of applying ML models to solve many real-life problems, encourage financial economists to apply ML in asset pricing. The ML-based model also enables researchers to tackle the asset pricing prediction problem from different aspects, e.g., working with high dimensional and high-frequency data [157, 158], incorporating news and text-based information [159], and considering behavioral aspects [160] into asset pricing models. Evidence suggests, compared to traditional econometrics models, e.g., ARIMA, VAR, and GARCH, ML-based models are superior in predicting the future price of an asset [161–164]. Studies also show that portfolios developed based on ML models could earn positive alpha [165–167]. Most of this work focus on applying ML methods directly to predict the future price while paying less attention to the interpretability of models for explaining risk premia.

Machine learning models, especially those based on artificial neural networks (ANN), are often criticized for their high obscurity. In neural networks, the weight and biases are discovered automatically; as a result, the explanation of how the network functions remain obscure. In addition to superior performance, the interpretability of the model is imperative for the successful real-world application of any model. In the business decision-making process, investors need interpretability of the model to understand what they are doing; policymakers want to understand the causal relations and know its implications. The traditional linear models are both easily interpretable and empirically examined; therefore, they are reliable and have more widespread use. For ML models, investors also need a similar level of interpretability

and understanding behind their investment decisions. The inability of earlier ANN-based models of asset pricing in explaining positive alpha motivates this work. To address the burgeoning requirements and facilitate the adoption of advanced ML models and neural networks in finance, I propose to identify why ML methods work better or what significant hidden relations they uncover for earning positive alpha. In contrast to other contemporary ML applications in this field, here I explain the risk premia of an asset-based on the factor model. I propose a latent factor model to explain the return difference among a group of similar assets.

I use an ANN-based model - *autoencoder* - to learn latent representation from the return data [61]. Over time, autoencoder becomes a very successful tool for nonlinear principal component analysis [61]. The autoencoder generated latent factors is used to identify representative and non-representative stocks in an index. The measure of representativeness - *communal difference* - is the L_2 norm of the difference between the original and constructed data from the latent factors [95]. This communal difference is then used to define communal assets (representative stocks) and non-communal (non-representative) assets in an index. I combine the latent factors with Fama-French characteristic factors, develop 18 portfolios, and define the communal factor as the daily average return difference between six non-communal and six communal portfolios.

In addition to explaining the risk premia of assets, the latent factor model derived from autoencoder also has a better prediction performance. I apply the estimated next-month latent factors from a rolling window of two years in the next-day return prediction. The resultant model outperforms both Fama-French factor models. The Sharpe ratio of the latent factor model exceeds traditional factor models and justifies the economic significance of the proposed model. The excellent predictive power of the latent factor model comes from the communal factor. To check the robustness of the proposed model, I conduct several multivariate regression analysis

on the communal factor along with the Fama-French five factors for each index. The result confirms that the communal factor has a statistically significant impact on stock returns. The panel regression analysis also shows that the effect of communal factors is no less than that of many other characteristics based factors. In some cases, the inclusion of the communal factor reduces the artifacts of the characteristics based factors.

Although the latent factor model involves considerable complexity while a simple model like Sharpe ratio [168] can provide similar risk-return trade-off information, the latent factor model is still superior to the Sharpe ratio in retaining information form the cross-section of assets. Sharpe ratio is based only on the historical risk and return of individual assets [168]. It fails to consider the covariances of the cross-section of assets return, and for different assets the Sharpe ratio alone cannot provide any relative information. The ANN-based model captures the covariances and any nonlinear interaction within the cross-section of all returns in a high dimensional latent space. It gains an edge in interpreting the risk and return of an asset concerning the other assets in the index.

This works contributes to asset pricing and finance literature in the following ways. First, traditional factor models, including PCA, are unable to predict the time-varying volatility in the profit return structure. The latent factor model based on autoencoder overcomes this issue by capturing both nonlinearity and time-varying volatility in the assets returns. Second, the proposed model uses daily data with a two-year rolling window and accurately detects the short term variation in stock returns. On the other hand, all other factor-based models for asset pricing use monthly data. While they can efficiently identify the long-term trend in stock return, nevertheless, they fail to detect short-term variations. Third, this work combines multiple latent factors into a single evaluative factor – communal factor – to better explain the return variances among the member stocks in an index. Fourth, the

superior performance of non-communal stocks come from the high risk associated with these stocks. Investors can use the communal factor to design a trade-off strategy between diversifying their assets portfolio and focusing on a few stocks from the index to outperform the index return. The proposed model helps an investor in improving his/er portfolio performance while reducing the transaction cost by investing in a small number of stocks.

The remainder of the chapter is organized as follows: Section 4.2 discusses traditional asset pricing theories followed by a brief discussion on the developments of machine learning based model in Section 4.3. Section 4.4 describes the autoencoder model architecture along with the data and hyperparameters settings. Section 4.5 provides experimental results and discusses the model implications for asset return prediction and portfolio optimization. Finally, Section 4.6 summarize the findings and concludes the chapter.

4.2 Traditional Asset Pricing Theories

CAPM [52, 169–171] provided the original and straightforward explanation of the return difference between assets. Based on Markowitz's portfolio theory [172], CAPM asserts that the expected return of an asset is a function of market risk premium and sensitivity of its return to the return on the market portfolio. However, to overcome the simplistic assumption of – the existence of mean–variance efficient market portfolio – and provide flexibility to investors, [53] proposed arbitrage pricing theory (APT). According to the APT, asset return can be expressed as a linear function of a variety of macroeconomic, market, and security-specific factors. APT allows users to include multiple risk factors and serves as an effective model for managing a portfolio and evaluating its performance concerning various factors. However, one major problem of the APT is there is no specification on which factors to use. In theory, an infinite number of factors can affect the risk premium of an asset, but it is impossible to

know the most significant factors upfront. A straightforward solution is to include the highest number of factors available to maximize the model fit. However, including a high number of factors without a proper justification can introduce multicollinearity and overfitting. Therefore, the success of applying APT in portfolio management depends on the analyst's experience, judgment, prior knowledge, and some random bias.

Historically, researchers use two primary approaches to overcome this issue. First, pre-specify factors based on prior knowledge, and second, analyze all available factors and identify latent factors. The collection of factors that impact an asset's expected return is vast and diverse. Over the years, hundreds of factors have been reported in empirical asset pricing literature. Researchers have identified factors based on empirical evidence, market influence, personal belief, and specific research needs. The most successful application of this approach is the Fama–French factor models [49, 173]. Initially, Fama and French proposed that the market, size, and book-to-market equity can successfully capture the cross-sectional variation in average stock returns [173]. The proposed three-factor model expands the CAPM model by adding the size risk and the value risk to market risk.

$$E_{it} = r_{ft} + \alpha_i + \beta_{i1}(E_{mt} - r_{ft}) + \beta_{i2}SMB_t + \beta_{i3}HML_t + \epsilon_{it} \tag{4.1}$$

In Equation (4.1), SMB is the risk premia for small stock and can be calculated as the return on a diversified portfolio of small stocks minus the return on a diversified portfolio of big stocks. HML is the value risk premia and is equal to the average return on the value portfolios minus the average return on the growth portfolios. In later empirical tests, it is found that these three factors β_1, β_2, and β_3 are unable to capture a significant amount of variation in the average return [174–176]. Therefore,

it appears to be an incomplete model for measuring risk premia. In 2015, Fama and French offered a new five-factor model that includes the profitability and investment factors to the three-factor model [49].

$$E_{it} = r_{ft} + \alpha_i + \beta_{i1}(E_{mt} - r_{ft}) + \beta_{i2} SMB_t + \beta_{i3} HML_t + \beta_{i4} RMW_t + \beta_{i5} CMA_t + \epsilon_{it} \quad (4.2)$$

where RMW is the difference between the return on the portfolio of robust versus weak profitability stocks and CMA is the difference between returns on the portfolio of low versus high investment stocks.

One major shortcoming of these factor models is that the prior knowledge used to identify relevant factors came from empirical analysis on the average returns from historical data. Factors are assumed to be fully observable from historical data, but in reality, historical return at best provides partial observation. The choice of factors is made mostly based on practical experience and is somewhat arbitrary [177]. In addition to Fama–French models, some other factor models are available, including multifactor-model [178], Carhart four-factor model [179],[1] and the six-factor model [180].[2] These models are nevertheless the extensions of Fama–French models and essentially have the same limitations as the Fama–French factor models.

The second approach to overcome the problem of APT's high-dimensional factor space adopt a factor analysis approach to reduce the dimensionality of the observed factors and derive latent variables. The most common technique for identifying latent factors is the Principal Component Analysis (PCA) that simultaneously estimates the latent risk factors and their betas from the panel of the realized returns [136]. PCA is much simpler in terms of basic concepts and computation costs and provides an

[1]The four-factor asset pricing model incorporates the momentum in Fama–French three-factor model [179].

[2]The six-factor asset pricing model introduced the human capital component to the Fama–French five-factor model [180].

approximate factor structure without requiring any prior knowledge. PCA uses the covariance matrix of returns to develop a factor model. Until recently, PCA was the most sophisticated dimension reduction technique for big financial data. PCA linearly map N individual returns into K factors where $K < N$. However, PCA still fails to capture nonlinearity among those observed factors. Furthermore, PCA only accommodates the static loadings (beta) and lacks the flexibility to incorporate additional data beyond the initial returns for constructing the covariance matrix. On the other hand, asset return is highly volatile in time, requires a dynamic model to incorporate the conditioning information continuously and incur a prohibitive cost to many static models, including PCA.

To overcome the problem associated with the factor-based modes and the factor analysis model (PCA), [48] recently proposed the instrumented PCA (IPCA). IPCA incorporates the idea of instrumental variables from the generalized method of moments (GMM) [181]. It uses observable asset characteristics (L) as an instrumental variable for latent conditional loading. As a result, the factors beta partially depends on the characteristics of observable asset and establish a relationship between the asset characteristics and expected return. IPCA also assumes that the mapping from L asset characteristics to K betas is linear. However, the assumption of linearity rarely holds in financial data. Many leading asset pricing models predict nonlinearity in the return dynamics [182–184]. Nonlinearity causes significant bias on the prediction of all asset pricing models mentioned above – CAPM, APT, Fama–French factor models, PCA, IPCA – as these models assume linearity and normality in the return distribution. Relaxing the linear restriction from the model will significantly improve model performance. For example, [55] outperforms the original linear model and achieves a zero intercept portfolio by incorporating the CDS spread and its associated quadratic term in the model. By incorporating nonlinear predictors, these models will substitute Fama–French's linear size and value risk exposure. Besides, it is

evident that financial time series exhibit time-varying volatility and regime-switching behavior [185–188]. Therefore calculating only one beta from the regression of historical data will not capture the dynamic risk exposure of predictors on the asset return.

4.3 Machine Learning in Asset Pricing

Machine learning (ML) based approaches can tackle the issues related to traditional asset pricing models discussed in Section 4.2, i.e., reducing dimension and modeling the time-varying volatility. ANN can perform nonlinear transformations on the input predictors, learn the effective data representation in a latent space, and offer several advantages over traditional linear asset pricing models. The application of ML-based model in asset pricing is not new; it can be traced back to the early 90's [189–192]. The use of ANN was also common in that period. ANN is applied to recognize patterns in the 'candlesticks' chart [189], to develop a composite synthesized rule for trading S&P-500 future contract [190,191], neural network-based genetic algorithm for feature selection and topology optimization [192], and rule-based neural network for trading S&P-500 future contracts [193]. In this earlier stage, most studies are conducted to predict the direction of the future price movement [164, 194, 195]. Studies are mainly focused on predicting the binary outcomes in the next time step, i.e., '0' indicating next time step price will be lower than current time step, and '1' meaning the opposite and developing investment strategy based on the prediction [191, 192, 194, 196].

Studies also use support vector machine (SVM) to predict the daily direction of price change [194,195,197]. Kim [194] uses both polynomials and the Gaussian Radial Basis Function as SVM kernels to predict the direction in the Korea composite stock index price, compare the performance with ANN and case-based reasoning of [193], and conclude that SVM performs better, but is sensitive on hyperparameter settings. Huang et al. [195] analyze the performance for NIKKEI 225 index and compare it

with linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and Elman backpropagation neural networks (EBNN). The result of these two efforts signify the importance of selecting hyperparameters.

In asset pricing, the task essentially boils down to predicting the final price or return; therefore, regression-based approaches are preferable over classification-based approaches. Recent advancement in ML techniques allow researchers to design regression-based approaches that can directly predict return or price. The use of neural networks is especially noticeable for predicting actual price [161, 198–200]. These works widely vary based on purpose, sampled data, and variables used. Efforts include, feature selection, optimization [199, 201], explaining curve dynamics [202], and application of reinforcement learning that supports prescriptive analysis and decision making in the automated trading scenario [203]. Kondratyev [202] use multi-layer perceptrons to capture the dynamic of the curve in nonlinear space. The model is tested on 300 monthly Brent crude oil forward price curves and 250 monthly USD swap curves. The testing results confirm that ANN offers superior prediction capabilities on the long term curve transformation that is impossible with PCA.

The majority of these models attempt to predict the price or the price movement of only one or a couple of indices. Gu et al. [47] first addresses the need for an overreaching generalized model, incorporates all the stocks listed in all three major US exchanges: NYSE, AMEX, and NASDAQ, and compares the performance of ANN-based approaches with traditional techniques (i.e., linear regression, generalized linear regression, principal component regression (PCR), partial least square (PLS), LASSO, and random forest). They conclude that the ANN-based approach is superior in explaining the return behavior of assets.

Autoencoder, a successful ANN approach for non-linear dimension reduction, nevertheless is rarely used in asset pricing. Two rare, but successful examples that apply autoencoder for better prediction and portfolio management strategy

are [95, 96]. An autoencoder based deep portfolio management strategy is proposed in [95]. Inspired by Markowitz's classic portfolio theory, [95] outline a four-step deep portfolio routine: encode, calibrate, validate, and verify. The deep portfolio theory first reconstructs the values of the stocks in an index, ranks the stocks according to the degree of communal information[3], and then creates deep portfolios based on this ranking. Similar to PCA, the standard autoencoder proposed in [95] only depends on the returns and does not leverage the conditioning variables.

Gu et al. [96] proposed an augmented conditional autoencoder that uses information from covariates to reduce the dimensionality directly. Following [48], they augmented the standard autoencoder with the asset-specific covariates in the factor loading specification. Their conditional autoencoder uses two input layers: the first input layer takes individual asset characteristics and the second input layer receives individual asset returns. In the final step, the conditional autoencoder multiplies the output betas from the first network and the factors of the second network and produce the estimation of each asset return.

Inspired by the successes of autoencoder in [95] and [96], I use the autoencoder to calculate the communal information and create a latent factor model similar to to [49, 177]. As a result, the proposed model captures both nonlinear interaction and time-varying volatility without depending on any observable asset-specific characteristic.

4.4 Latent Factor Model

4.4.1 Autoencoder

It is common in finance to estimate asset returns based on the linear latent factors and static loadings [204, 205].

$$r_t = \beta f_t + u_t' \tag{4.3}$$

[3]Same as the communal difference defined in 4.1, the l_2 norm differences between every stock in an index and their reconstructed returns generated by the autoencoder.

Where r_t is the vector of excess return, f_t is the K-dimensional vector of factor returns, β is an $N \times K$ matrix of factor loadings, and u_t is the $N \times 1$ vector of idiosyncratic errors. The matrix representation of Equation (4.3) in the PCA analysis is $R = \beta F + U$. The factor loading is estimated via the singular value decomposition (SVD) on returns [206, 207]:

$$R = P\Lambda Q + \hat{U} \tag{4.4}$$

where P is the $N \times K$ matrix of the left singular vectors, Q is the $K \times T$ matrix of the right singular vectors and $\hat{U}$ is an $N \times T$ matrix of residuals.

Section 4.2 states that a linear factor model, for example, PCA, cannot capture the nonlinearity and time-varying volatility in time series data. Autoencoder is a neural network model for learning nonlinear representation of latent factors. The close connection between autoencoder and PCA is well recognized in finance literature [96, 208]. Autoencoder learns to encode and decode data while moving data from its input layer to the output layer through a bottleneck. Autoencoder with one hidden layer and no activation function is essentially equivalent to PCA. The simplest autoencoder is a feedforward neural network with one input layer, one output layer, and one or more hidden layer(s). The input layer takes input variables and passes them to a small number of the hidden layer(s) (bottleneck) to generate a compressed representation of inputs (encoding). The output layer decodes the compressed codes into the closest possible representation of the original data (decoding). The input and output layers of an autoencoder have the same number of nodes.

I follow the common practice in the finance ML literature to use the cross section of excess return as the input in the input layer of the network. Let $X_{i,t}$ be the price of stock i at time t, where $1 \leq i \leq N$ is the stock index and $1 \leq t \leq T$ is the time index. The return for stock i at time t is defined as $r_{i,t} = log(X_{i,t}/X_{i,t-1})$. For N

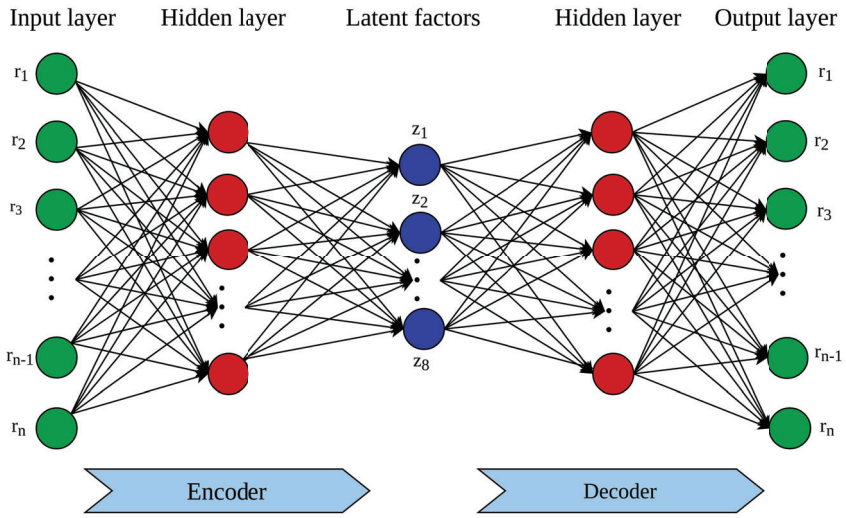


Figure 4.1 Autoencoder model.

stocks in an index, the input layer takes $r^0 = r = (r_1, \ldots, r_N)$, the recursive output of each neuron at layer $l > 0$ is $r_k^l = f(b^{(l-1)} + r^{(l-1)^\top} W^{(l-1)})$, with the final output as latent factors:

$$F(r, b, W) = b^{(L-1)} + R^{(L-1)^\top} W^{(L-1)} \tag{4.5}$$

where $W^{(l-1)}$ is $K^{(l)} \times K^{(l-1)}$ matrix of the weight parameters and $b^{(l-1)}$ is the $K^{(l)} \times 1$ vector for bias parameters. For the purpose of this work, I use three hidden layers and the rectified linear unit (ReLU)[4] as the nonlinear activation function.

Figure 4.1 shows the architecture of the autoencoder. For each time period t, the model takes return r for N stocks as both input and output and learn the latent factors z's. An autoencoder contains two components, encoder and decoder. During the encoding stage, the encoder maps the input vector $r \in \mathbb{R}^N$ to a latent representation $z \in \mathbb{R}^H$ (H is the dimensionality of latent vector.) with a deterministic mapping function $z = f_\theta(r) = ReLU(Wr + b)$ parameterized by $\theta = \{W, b\}$. During the decoding state, the decoder network transforms the latent representation back to

[4]A rectifier linear unit (ReLU) is a nonlinear activation function that allows a model to capture non-linear interactions among input variables. It is defined as: $f(x) = \max(0, x)$, for more details please see [209, 210]

a reconstructed vector $y \in \mathbb{R}^N$, where $y = g_{\theta'}(z) = ReLU(W'z + b')$ parameterized by $\theta' = \{W', b'\}$. During the training stage, I apply the built-in training algorithms to optimize parameters θ and θ' with the following objective of minimizing the average reconstruction error:

$$\arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} \|r_i - g_{\theta'}(f_\theta(r_i))\|^2 \tag{4.6}$$

4.4.2 Data and Optimization Techniques

I analyze three indices, S&P-500, NASDAQ-100, and RUSSELL-3000, from the U.S. stock market for this study and use the daily data for all the stocks in the indices from January 1st, 1990 to December 31st, 2018. I collect daily stock returns and company-specific characteristics from Bloomberg Terminal. To calculate excess returns, I use U.S. treasury bill rates. I use Fama–French factors data from the Kenneth French Data Library. S&P-500, NASDAQ-100, and RUSSELL-3000 are three major indices in the U.S., considered as the representatives of the U.S. stock market, and offer enough diversification opportunities for investors. As the model's primary goal is to identify representative and non-representative stocks among a group of stocks, applying to stocks on an individual index serves the purpose better than using all the listed stocks irrespectively.

Table 4.1 presents different network structures used in experiments, each of which have different parameter settings. The performance difference among many parameter configurations is insignificant except when batch size = 100 and epoch = 10. I report the results generated from the following experiment configuration: batch size = 20, the numbers of neurons in three hidden layers are 32-8-32, and epochs = 100.

Table 4.1 Parameters

Parameters	Level			
Batch size	10	20	50	100
Number of hidden node	8-8–8	16-8–16	32-8–32	56-8-56
Epochs	10	50	100	200

Training deep neural networks to reach an acceptable minimum with the vanilla version of the stochastic gradient descent method is time consuming. It uses a fixed learning rate and is not efficient in reaching the training target [211]. Adam is a gradient-based optimization of stochastic objective functions and continuously adjusts the learning rate based on the adaptive estimates of lower-order moments. To train the deep neural network, I used Adam that easily escapes saddle points while providing fast convergence [152].

4.4.3 Prediction Using Rolling Window

Predictive models for financial data use time-sensitive information and are prone to information leakage. Information leakage might occur when training data have unexpected access to information that is available during training, nevertheless not available during real-world deployment and use. For time series, even carefully designed train and test data may have information leakage if time continuity is not explicitly considered. Only can the present and past states determine the state of a future period. If the model already knows the future state (future return), it can easily identify the features from future observations for the training phase and make estimation and prediction under information leak. In the current setting, information leakage can occur in two ways, using future return data for generating latent factors and developing the predictive model using future data points. Training the latent factor models with mixed current and future return data can provide an "artificially" crafted good fit for model parameters. To overcome the leakage problem in identifying

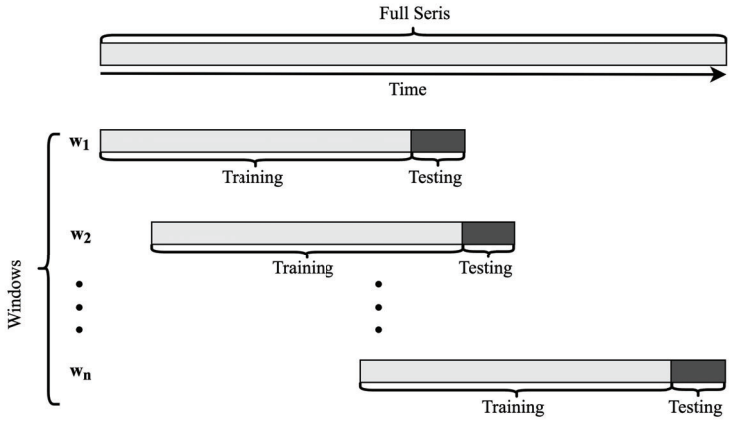


Figure 4.2 Rolling window for asset pricing model, with w = 504 days.

latent factors, I conduct a rolling window analysis of 24 months on the daily data. I use the first 24 months (504 days) as the training dataset and the next month (21 days)[5] as the test dataset (See Figure 4.2). I conduct a two-phase prediction function based on the auto-encoder framework.

In the rolling window system, first I train the autoencoder using the 24 months training data and estimate one month ahead latent factors Z_t to Z_{t+21} based on the trained weights. After that, I use the latent factors Z_t to predict the next day's return for each assets $r_{i,t+1}$. I define the predicted return as follows:

$$r_{i,t+1} = \alpha_i(r_{m,t}) + \sum_{j=1}^{8}(\beta_{i,j} z_{j,t}) + e_{i,t+1} \tag{4.7}$$

where r_m is the CAPM factor of the market risk premium $(R_m - r_f)$ and $e_{i,t+1}$ is the zero mean residual. I regress the future return on the eight latent factors z's of training data to estimate the eight β's, and then use these β's along with one month ahead latent factors to predict the next day's return. I use the predicted returns by linear regression to compare the performance of the latent factor model and the

[5]Following common literature, I use 21 as the number of trading days in a month

established factor models: Fama–French three-factor model (FF-3), Fama–French five-factor model (FF-5) and PCA.

4.4.4 The Communal Factor

In addition to developing a predictive model, I also explain the reason behind the superior performance of the predictive model. Once the autoencoder model is developed and latent factors are identified, I combine these latent factors into one evaluative factor based on the latent factors' ability to explain return variance. Latent factors constitute a representation of *communal information* for the stocks in the index. I follow the terminology used in [95] to define a stock communality (communal or non-communal) as its ability to be reconstructed from the latent factors Z. The communal difference CD is calculated as the l_2 norm of the error between each stock's returns between $t = 1$ and T (the input of autoencoder) and the reconstructed version (the output of the autoencoder).

$$CD_i = \sum_{t=1}^{T} \|r_{i,t} - F(r_{i,t}, b, W)\|^2 \tag{4.8}$$

At the beginning of each month, each stock's communal difference is calculated based on the last two years' latent factors and rank all stocks based on it. As [95] points out, the stocks in the same index share some common characteristics. The communal information has some connection to the shared characteristics: stocks that have many common features in an index demonstrate less communal differences, whereas stocks with less common characteristics will have higher communal differences. I categorize the stocks sorted on ascending order of communal differences into three categories: communal (top 20%), non-communal (bottom 20%), and moderate (remaining 60%).

Table 4.2 Daily Out-of-sample Stock-level Prediction Performance (Percentage R^2_{oos})

Index	FF-3	FF-5	PCA	Latent Factors
S&P-500	0.0718	0.0817	0.0855	0.0968
RUSSELL-3000	0.0653	0.0667	0.0589	0.0854
NASDAQ-100	0.1186	0.1298	0.1275	0.1492

Finally, similar to the Fama–French factor models [49, 177], I introduce the return difference between the noncommunal stocks and communal stocks as *noncommunal minus communal* (NCMC) factor. NCMC represents a stock's property of constructibility concerning the index return. I also evaluate the performance of the latent factor model by integrating them with the renowned Fama–French three-factor model.

$$E(r)_{it} = \alpha_i + \beta_{i1}(E_{mt} - r_{ft}) + \beta_{i2}SMB_t + \beta_{i3}HML_t + \beta_{i4}NCMC_t + \epsilon_{it} \qquad (4.9)$$

Equation (4.9) also enables us to test the latent factors' ability to explain the return difference among assets. A significant value for β_4 will indicate that the latent factors contribute to the expected return of an asset and reduce the unexplained variance. As non-communal assets are less representative than the communal assets, they should demonstrate a higher risk than communal assets while being rewarded with a higher return. Therefore, I hypothesize that the coefficient (β_4) of NCMC is significant, and its sign is positive.

4.5 Results and Discussion

4.5.1 Prediction Performance

The latent factor model has a better predictive capacity compared to traditional factor models in predicting the next day's returns. I compare the proposed model in

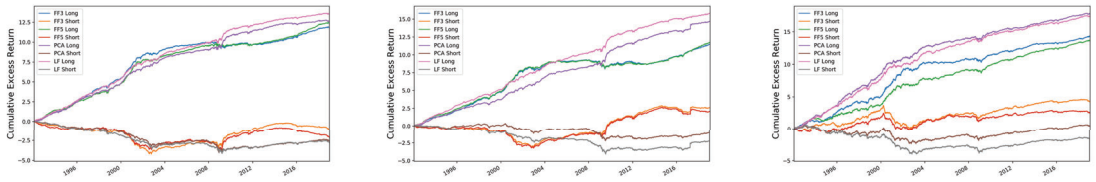
Equation (4.7) with the Fama–French three-factor (FF-3) model in Equation (4.1), the Fama–French five-factor (FF-5) model in Equation (4.2), and PCA. I evaluate the prediction performance in terms of the out-of-sample predictive R^2 (R^2_{OOS}) and Sharpe ratio. Following [48], the out of sample R^2 is calculated as follows:

$$R^2_{OOS} = 1 - \frac{\sum_{(i,t)\in OOS} \left(r_{i,t} - \hat{\beta'}_{i,t-1}\hat{f}_t\right)^2}{\sum_{(i,t)\in OOS} r^2_{i,t}} \tag{4.10}$$

The R^2_{OOS} indicate that the metrics only assesses the fits on the testing sub-samples that are never used to train/estimate the model. Table 4.2 shows that the latent factor model outperforms the widely used linear factor model (FF-5) by 18%, 28%, 15%, and the PCA approach by 13%, 45%, and 17% in S&P500, Russel-3000, and NASDAQ, respectively.

I compare the economic significance of the latent factor model with the FF-3, FF-5, and PCA. I build a decile long-short portfolio sorted on the model-predicted returns. During the testing period, I apply the proposed model on each day's return r_t to predict the return of all stocks on the next day r_{t+1}. I develop a zero investment equally weighted long-short portfolio. For each day, I short \$100 from predicted worst-performing assets and invest this \$100 on predicted best-performing assets. I construct the portfolio as follows: I first derive the predicted return on each specific day for each stock and then sort stocks in the decreasing order of predicted values. I choose the top 10% of stocks with the highest predicted returns to construct the sub-portfolio with a long position, where each stock has an equal investment. On the other hand, I short the bottom 10% of stocks with the lowest predicted returns, each of which has the same short amount from each stock.

Figure 4.3 presents the cumulative return of the long-short portfolio constructed by the four models for the three indices over 26 years. In all three chosen indices, I



(a) S&P-500 **(b)** RUSSELL-3000 **(c)** NASDAQ-100

Figure 4.3 Cumulative excess return on long-short portfolio (1992-2018).

observe that the latent factor model's performance is better in both long and short portfolios. The better performance of the proposed latent factor model is quite evident for RUSSELL-3000 and NASDAQ-100. Although the performance improvement is marginal for S&P-500, I observe a clear trend, i.e., after 2007–2008 financial crisis, both the long only and short only portfolios based on latent factors outperform the other two factor models. This trend confirms the idea of high uncertainty associated with non-communal stocks. During and after the financial crisis, a significant reshuffle took place in these indices. Several major companies, e.g., Lehman Brothers and Bear Stearns, went bankrupt and were removed from the S&P-500 index and several new companies, e.g., Mastercard, Visa, Salesforce, Netflix, entered into the index. Over the last decade, these newly registered firms demonstrated superior performance. As a result, their return structure was entirely different from existing ones. Because the proposed model tries to detect return structure based on common return characteristics, these new and strong stocks do not have these characteristics yet and are identified as the non-communal stocks. The superior performance of these newly entered firms in the index made the latent factor model outstanding after the 2007–2008 financial crisis.

The long-short portfolio created based on the latent factors model earns not only the highest return but also shows significant improvement in the Sharpe ratio in

Table 4.3 Long-Short Portfolio Sharpe Ratio

Index	FF-3	FF-5	PCA	Latent Factors
S&P-500	2.031	2.489	2.320	2.704
RUSSELL-3000	1.718	1.853	2.328	2.932
NASDAQ-100	1.096	1.253	1.877	2.094

all three studied indices. In Table 4.3, the portfolio Sharpe ratio is calculated as:

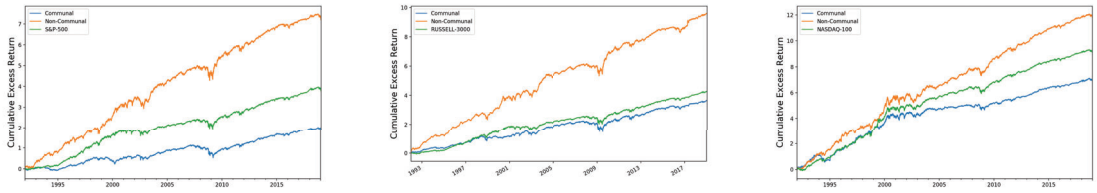
$$SR = \frac{\sum_{(t=1)}^{T} (P_{r_t}) \times 252}{\sigma_{P_{r_t}} \times \sqrt{252}} \tag{4.11}$$

I adopt the formula from Morningstar, Inc. to calculate the yearly Sharpe ratio from daily return. For S&P-500, the improvement of the Sharpe ratio in the proposed latent factors model is 33%, 9%, and 17% over FF-3, FF-5, and PCA, respectively. For RUSSELL-300, the improvement is 71%, 58%, and 26% over FF-3, FF-5, and PCA, and for NASDAQ-100 the improvement is 91%, 67%, and 12% over FF-3, FF-5, and PCA.

4.5.2 The Latent Factor

In this section, I try to evaluate what contributes to the better predictive performance of the latent factor model. At the beginning of each month, I use the last two years' latent factors to calculate the communal differences of each stock. Stocks sorted on communal differences are categorized communal (top 20%), non-communal (bottom 20%), and moderate (remaining 60%). I hold the stocks in each of these three groups for the whole month and calculate the average cumulative return earned by the stocks in each group. For example, I calculate the cumulative return for communal stocks as follows:

$$C_{i,t}^{c} = \frac{1}{S} \sum_{i=1}^{S} \sum_{t=1}^{T} r_{i,t} \tag{4.12}$$



(a) S&P-500 **(b)** RUSSELL-3000 **(c)** NASDAQ-100

Figure 4.4 Cumulative excess return of communal and non-communal portfolio (1992-2018).

Table 4.4 Communal and Non-Communal Portfolio

	S&P-500			RUSSEL-3000			NASDAQ-100		
Variables	C	M	NC	C	M	NC	C	M	NC
Mean	0.029	0.048	0.067	0.073	0.055	0.113	0.103	0.121	0.146
STD	0.872	1.157	1.597	1.854	1.186	2.536	1.818	1.575	2.007
Sharpe	0.528	0.658	0.665	0.625	0.736	0.707	0.899	1.219	1.155

Note: C denote communal stocks, M denote moderate stocks, and NC denote non-communal stocks in the index.

The daily data shows that in all three indices, non-communal stocks outperform both their index and communal stocks by a significant margin in Figure 4.4. The communal stocks, on the other hand, perform worse than the index return. This observation can be explained by the nature of the definition of communal and non-communal stocks: communal stocks are robust with a long history of being in the index and have less divergence from the index; in contrast, non-communal stocks have high variation in yield and show significant deviation from the other stocks in the index. As a result, predicting future returns of these stocks is difficult. The higher uncertainty involved in these non-communal stocks is rewarded by superior return.

Table 4.4 reports the average daily return, standard deviation, and Sharpe ratio of communal, moderate, and non-communal stocks of all three studied indices. Although the non-communal portfolio demonstrates a high average excess return than communal and moderate stocks, it also has a high standard deviation. In addition,

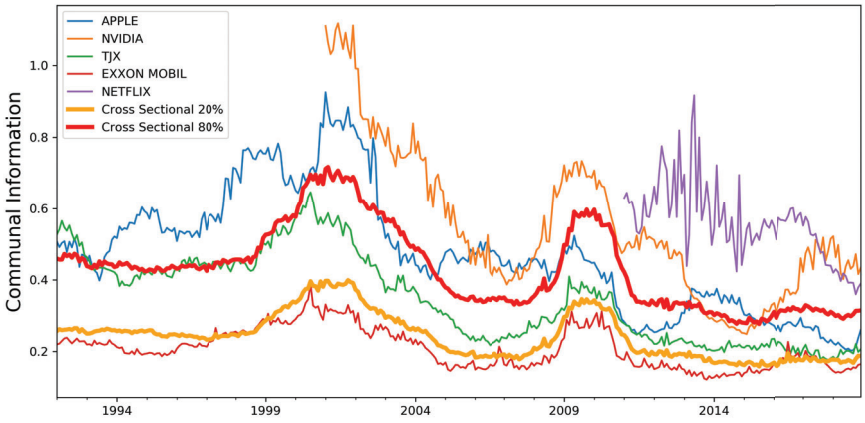


Figure 4.5 Timeline of stocks existence in different portfolios.

the improvement in the Sharpe ratio is marginal. This observation signifies that the high average return does not come from the superior performance of the stocks in the portfolio; instead, it is the compensation of high risk associated with these non-communal stocks.

In Figure 4.5, I evaluate some selected stocks from S&P-500 and the timeline of being in the communal and non-communal portfolio. The dynamic nature of the latent factor model assigns stocks in the communal, moderate, or non-communal portfolio at the beginning of each month. A closer look reveals this assignment is closely related to the life-cycle of the stocks and its market performance. The thick orange and red lines in Figure 4.5 represent the cross-sectional 20% and 80% of communal differences at any given time, respectively. Stocks that are a member of the S&P-500 index for a long time and have stable earning patterns are more likely to belong to the communal or moderate group, i.e., TJX (from 1985) and Exxon mobile (from 1964). They closely mimic the cross-sectional mean. Upon its recent introduction to the S&P-500 index, TJX was initially in the non-communal group but eventually, moved into the moderate group, and remained there.

Table 4.5 Ability to Explain Return Difference

	S&P-500		RUSSEL-3000		NASDAQ-100	
Variables	C	NC	C	NC	C	NC
Mkt-Rf	-0.0648***	0.016	-0.0337***	0.2272***	-0.0612***	0.0203**
	(0.02)	(0.004)	(0.013)	(0.066)	(0.009)	(0.010)
L1	0.0004***	0.002	0.002	0.0057**	-0.0008	0.0010
	(0.000)	(0.000)	(0.001)	(0.003)	(0.001)	(0.001)
L2	0.0008***	0.0004	0.0021***	00026	0.0010***	0.0014*
	(0.000)	(0.000)	(0.001)	(0.003)	(0.000)	(0.001)
L3	0.0005***	0.0006**	-0.0004**	0.0018	0.0019***	0.0002
	(0.000)	(0.000)	(0.000)	(0.002)	(0.001)	(0.001)
L4	0.0003**	0.0019***	-0.0003	-0.0024	0.0017**	0.0029***
	(0.000)	(0.000)	(0.001)	(0.003)	(0.001)	(0.001)
L5	-0.0011***	-0.0007**	-0.0001	0.0034	-0.016**	-0.0020***
	(0.000)	(0.000)	(0.000)	(0.003)	(0.001)	(0.001)
L6	-0.0001	0.0008***	0.0006***	-0.0039	0.0020***	0.0010
	(0.000)	(0.000)	(0.000)	(0.003)	(0.000)	(0.001)
L7	0.0009***	0.0008***	0.0005***	-0.0046***	0.0007	0.0006
	(0.000)	(0.000)	(0.000)	(0.001)	(0.001)	(0.002)
L8	0.0000	0.0028***	0.0017***	0.0055**	0.0010***	0.0021***
	(0.000)	(0.000)	(0.001)	(0.003)	(0.001)	(0.001)
r^2	0.005	0.001	0.007	0.005	0.004	0.001

Note: C denote communal stocks and NC denote non-communal stocks in the index. Values in parenthesis indicate standard error. *, **, ***, represent significant at 10%, 5% and 1% level of significance respectively.

On the other hand, growth stocks that enter the index at a later time and have significant fluctuation in return tend to stay in the non-communal group, i.e., Nvidia, Netflix. Netflix has been in the non-communal group for nearly its entire lifetime in the index. Besides, it also has significant variation in its communal information. The impact of market performance on my categorization is visible in Apple and Nvidia. The below-par performance of Apple in the early '90s and the return of Steve Jobs in the late '90s are highlighted in the two spikes in the Apple communal information line. The introduction of iPod (2001) and iPhone (2007) is also reflected in Apple's

Table 4.6 Average Daily Return on Different Factor Portfolios

	Big/High	Big/Medium	Big/Low	Small/High	Small/Medium	Small/Low
Communal	0.0489	0.0519	0.0306	0.0830	0.0667	0.1116
Moderate	0.0739	0.0736	0.0633	0.1090	0.0963	0.0927
Non-Communal	0.1208	0.1239	0.1286	0.1371	0.1345	0.1388

Note: Individual indices average factor return are available at Appendix B.

communal information. Following these two events, Apple rose to the non-communal group temporarily while otherwise remaining in the moderate group. Apple's final appearance in the non-communal group was the result of its record-breaking revenue for the first quarter of 2014 [212]. A similar pattern is also evident for Nvidia. Most of the time, Nvidia remains in the non-communal group except in 2014. In 2014, Nvidia experienced a decline in revenue and net income. However, Nvidia's drive to AI and deep learning from 2015 brought it back to the non-communal group.

Table 4.5 clearly explains why non-communal stocks generate higher return than communal stocks. I regress the returns of communal or non-communal stocks on the latent factors. In all three indices, non-communal stocks have lower r^2, and fewer statistically significant factors than communal stocks. This finding signifies that the non-communal stocks are less stable, hard to predict, and have a higher risk. The superior return, therefore, is the reward for bearing these highly uncertain stocks.

4.5.3 Which Factors Matter?

To analyze whether the latent factors from Autoencoders have some advantage over traditional factors in the Fama–French model, I develop 18 ($2 \times 3 \times 3$) portfolios based on the size,[6] value,[7] and communal difference. Table 4.6 shows the daily average returns on these factor portfolios. I find a significant difference in average returns

[6]Following [177], in the size portfolio, stocks are identified as either big or small based on the cross-sectional median size.

[7]Following [177], in the value portfolio, stocks are identified as either high value (top 30%), low value (bottom 30%), or medium value (remaining 40%) based on the cross-section of the book value of equity and the market value of equity. See details in [49, 173]

Table 4.7 Average Daily Return Difference

Factor	S&P-500	RUSSELL-3000	NASDAQ-100	Average
SMB	0.0221	0.0436	0.0309	0.0322
HML	0.0223	0.0194	0.0087	0.0168
NCMC	0.0562	0.0611	0.0327	0.0500

Note: SMB small minus big, indicate difference between size sorted portfolios; HML is high minus low, indicate value sorted portfolio; NCMC is Non-communal minus communal, indicate difference between communal information based sorted portfolio.

between the six high communal portfolios and six non-communal ones. I find that the average daily return on nine small portfolios is higher than that of nine big ones, which confirms the Fama–French factor model. It is also true for high versus low stocks. On average, the six high-value portfolios have higher average returns than the six low-value ones do. The six non-communal portfolios are outperforming both communal and moderate portfolios.

Interestingly, the small/low/non-communal portfolio has the maximum earnings. All three small/non-communal portfolios have a high daily average return; the difference among them is marginal. The observation leads to the conclusion that among the latent and characteristics factors, size and communality explain the most significant variance in return among the stocks in an index. The inclusion of latent factors minimizes the return difference between the high-value vs. low-value stocks. The finding also appears in the return difference among the factor portfolios in the three studied indices reported in Table 4.7. In S&P-500, the return differences in the size-sorted and value-sorted portfolios are almost identical (0.02%). In contrast, the return difference between non-communal versus communal is the highest (0.05%). However, in both RUSSELL-3000 and NASDAQ-100, the return difference in the value sorted portfolios is very low. In addition, in NASDAQ-100, the average return difference on the size and communal information sorted portfolios are almost similar (0.03%).

Table 4.8 Factor Significance

Variables	S&P-500	RUSSEL-3000	NASDAQ-100
Const	0.0005***	0.0008***	0.0010***
	(0.000)	(0.000)	(0.000)
$R_m - R_f$	-0.0511***	-0.0640***	-0.0247***
	(0.002)	(0.007)	(0.000)
SMB	0.0188***	0.0586***	0.0300***
	(0.0.003)	(0.013)	(0.000)
HML	-0.01805***	-0.0688***	-0.0588***
	(0.003)	(0.012)	(0.000)
RMW	0.1582***	0.0653***	0.0964***
	(0.004)	(0.019)	(0.000)
CMA	0.0556***	0.0089	-0.0506***
	(0.005)	(0.021)	(0.000)
NCMC	0.0817***	0.0621***	0.0245***
	(0.002)	(0.008)	(0.005)
R^2	0.011	0.008	0.010

Note: Values in parenthesis indicate standard error. *, **, ***, represent significant at 10%, 5% and 1% level of significance respectively.

Finally, I test the statistical significance of the proposed latent factors with the Fama–French factor model by conducting a panel regression of individual stock returns on the latent factors combined with the existing Fama–French factors Equation (4.9). The results are reported in Table 4.8 concerning all three indices. I find that the NCMC is as significant as any other factor in explaining the returns in all three indices. Among the indices, the communal factor has the highest impact on S&P-500 stocks. In addition, I find that both market factor ($R_m - R_f$) and value factor (HML) have a negative impact on these three indices return. Similar to the Fama–French factor model, I am unable to find an arbitrage-free model as the constant in all three models is statistically significant. The impact of constant is low compared to that of the other factors.

Table 4.9 Factor Significance in Communal and Non-Communal Stocks

	S&P500		RUSSEL-3000		NASDAQ-100	
Variables	C	NC	C	NC	C	NC
$R_m - R_f$	-0.0585***	-0.0118	-0.0377*	0.2904***	-0.0323**	0.0410*
	(0.014)	(0.024)	(0.023)	(0.097)	(0.012)	(0.045)
SMB	0.0635***	0.0292	00615***	0.3653**	0.0712***	-0.0268
	(0.023)	0.043	(0.020)	(0.181)	(0.020)	(0.045)
HML	-0.0181	-0.1238***	-0.0708***	0.3494**	0.0194	-0.1231***
	(0.024)	(0.040)	(0.021)	(0.167)	(0.042)	(0.047)
RMW	0.1245***	0.1263**	0.1180***	0.5106**	0.1875***	0.0852
	(0.035)	(0.062)	(0.041)	(0.251)	(0.054)	(0.073)
CMA	0.0421	0.0684	0.0122	-0.4549*	-0.0884	-0.0585
	(0.040)	(0.071)	(0.066)	(0.277)	(0.065)	(0.073
NCMC	0.0427***	0.0769***	0.0345***	0.2526***	0.0323***	0.0202*
	(0.016)	(0.029)	0.011	(0.101)	(0.012)	0.011
R^2	0.010	0.004	0.004	0.012	0.008	0.003

Note: C denote communal stocks and NC denote non-communal stocks in the index. Values in parenthesis indicate standard error. *, **, ***, represent significant at 10%, 5% and 1% level of significance respectively.

It is interesting to note that the impact of CMA[8] varies among different indices, i.e., CMA has a positive impact on S&P500, a negative impact on NASDAQ-500, and is not statistically significant for RUSSELL-3000. This finding is also obvious in evaluating the impact of Fama–French and latent factors concerning communal and non-communal stocks (Table 4.9). Except for non-communal stocks of RUSSELL-3000, CMA clearly shows no statistical significance to the returns of both communal and non-communal stocks in all remaining three indices. Table 4.9 also sheds some light on why non-communal stocks demonstrate higher average return than communal stocks do. In all three indices, non-communal stocks have lower r^2 and fewer statistically significant factors than communal stocks. Even for S&P-500, the market excess return is not significant for non-communal stocks,

[8]Conservative Minus Aggressive is the average return on the two conservative investment portfolios minus the average return on the two aggressive ones, for more details see [49]

whereas, for NASDAQ-100, it is only significant at the 10% threshold value. Besides, for non-communal stocks in S&P-500 and NASDAQ-100, the size factor is also not statistically significant, whereas it is statistically and economically significant for communal stocks. The observation confirms the claim in Section 4.5.2 that non-communal stocks are risky and difficult to predict; thus, they must generate a high return in compensating their uncertainty.

4.6 Conclusion

Asset pricing models are designed to identify risk measures and assign an appropriate reward for bearing those risks. Over the years, ML has received considerable success in predicting asset risk premia. In this paper, I use autoencoder to extract latent factors for explaining and predicting risk premia. The proposed communal factors help us understand the return variance among the stocks in an index. This work confirm that stocks in an index could be categorized based on their ability to share information, and stocks with low mutual information should earn a higher return for the associated higher risk. Investors will take advantage of this findings to design their trading strategies, i.e., (i) to beat the index by only investing in non-communal stocks while accepting high risk, (ii) to diversify their portfolio by investing both communal and non-communal stocks, or (iii) to avoid transaction costs by investing only a small number of stocks while replicating the average index return.

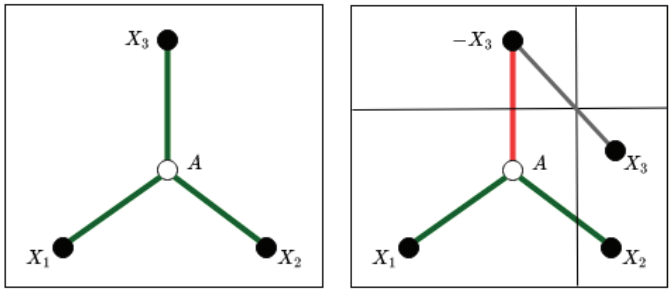
CHAPTER 5

THE NETWORK FACTOR OF EQUITY PRICING: A SIGNED GRAPH LAPLACIAN APPROACH

5.1 Introduction

Firms are connected through multiple types of networks. These network interconnections play an important role in how information and shocks transmit from one firm to another or to the whole system [56, 58, 59]. A single firm may generate a large impact on the market, such as the collapse of Lehman Brothers during the 2008 financial crisis. Similarly, market changes can affect individual firms as well as the interconnection between firms. For example, the COVID-19 pandemic has accelerated the adoption of digital technologies and changed supply-chain interactions. The underlying network dictating those interactions is not static, instead dynamically changing over time. Furthermore, the direction and magnitude of the exposure to network changes vary across firms. Empirical evidence suggests that firms' network exposures are associated with systematic risk, can improve return prediction [213], reduce diversification power [90], and warrant a centrality risk premium [91]. Meanwhile, most traditional asset pricing models mainly focus on firm-specific and market/macro factors, overlooking the indispensable interconnection among firms. In this work, I propose a novel approach to capture network information and to incorporate it into asset pricing models.

To evaluate the importance of network changes in equity prices, the first step is to be able to model and quantify the network change. The network representation inspired by graph theory is popular with a broad applicability across many domains, such as computer science and information system [72, 75, 214]. Unfortunately, those techniques are mainly for static and unsigned graph. Directly applying those techniques to model equity market network may produce suboptimal results. Firms

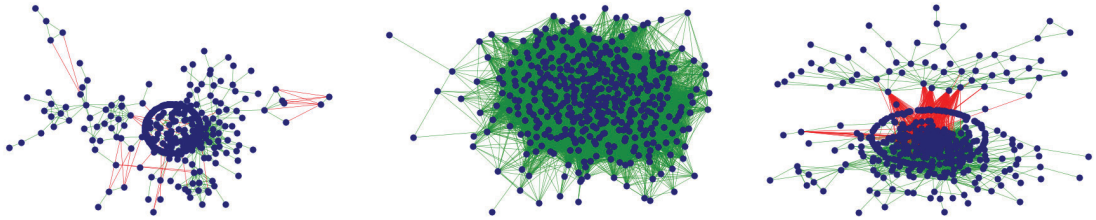


(a) Unsigned graph **(b)** Signed graph

Figure 5.1 Node representation in relation to the neighbors' proximity and antipodal proximity. (a) In unsigned graph, firm A's embedding should be the mean of its neighbors X_1, X_2 and X_3. (b) In signed graph firm A's embedding should be the mean of its positive neighbors X_1 and X_2 and antipodal points $-X_3$ of its negative neighbors X_3.

enter into or exit from the market and change their business models, capital structures, and supply chains. As a result, relationships among firms continually evolve. In addition, market shocks can affect the structure of the equity market network. A static graph built by aggregating all available information cannot capture such time-varying information. Previous studies use the correlation of firms' historical returns on a rolling basis to capture network dynamism [80, 215–217]. However, these efforts adopt unsigned networks that ignore the positive/negative signs by either using a distance function or using the absolute values of the correlation. Such application overlooks the core idea that a positive correlation indicates similarity and co-movement, while a negative correlation indicates the opposite. As shown in figure 5.1, when learning representation for a firm on an unsigned network, treating all of its neighbors indiscriminately (for a weighted network according to their edge weight) can produce an informative embedding. However, for a signed network, an informative representation of a firm should be similar to its positive neighbors and antipodal to its negative neighbors [218].

Figures 5.2 and 5.3 demonstrate the importance of modeling dynamic network changes and incorporating positive and negative connections in representing the

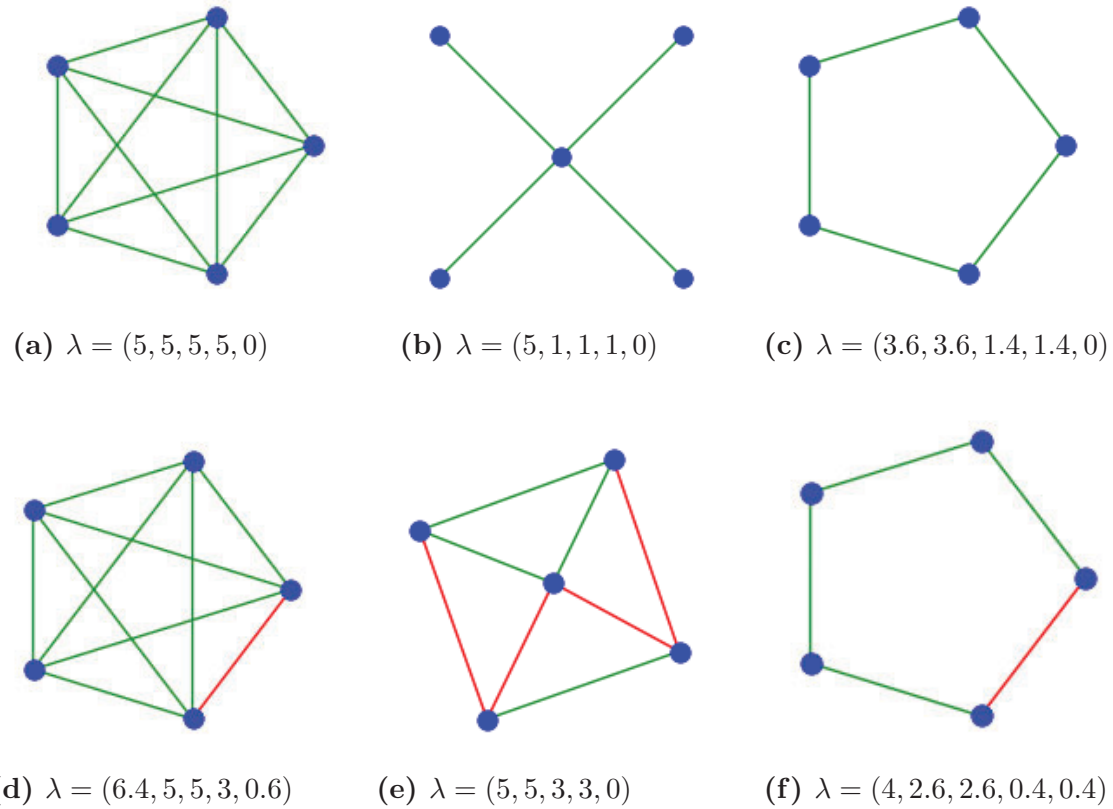


(a) December 2019 **(b)** April 2020 **(c)** December 2020

Figure 5.2 The network structure of S&P-500 stocks surrounding the COVID-19. Green (red) links represent the positive (negative) edge between two firms.

equity market network. During the normal economic period, the market consists of a mixture of positive and negative edges (Figure 5.2a and 5.2c). However, on the dawn of the Covid-19 pandemic (Figure 5.2b), most negative edges disappear, positive edges multiply, and the market forms a ball-shaped structure. The negative edges conveys information that an unsigned graph cannot reveal. For example, Figure 5.3(a)-(c) show that the smallest Eigenvalue (the rightmost λ in the parentheses) of all unsigned graph Laplacians is zero and fails to identify different structures of networks. Differently, for the signed graph Laplacian, only the balanced networks in Figure 5.3e (network with two sub-clusters) contain the zero Eigenvalue. Therefore, incorporating negative linkage helps to partition a network into multiple sub-networks (clusters), with positive edges signifying intra-cluster cohesion and negative edges serving as inter-cluster bridges. As shown in Figure 5.2c, the negative links are able to bisect S&P-500 stock network into two subgroups and positive links bond firms within the group.

In this chapter, I apply a generalized Laplacian matrix of [219] with the modified Laplacian for a signed graph to handle both positive and negative network connections. Particularly, the diagonal degree matrix in the modified Laplacian sums the absolute values of each firm's pairwise correlations with all others, and the affinity (weight) matrix contributes to retaining the signs of correlations (details in Section



(a) $\lambda = (5, 5, 5, 5, 0)$ **(b)** $\lambda = (5, 1, 1, 1, 0)$ **(c)** $\lambda = (3.6, 3.6, 1.4, 1.4, 0)$

(d) $\lambda = (6.4, 5, 5, 3, 0.6)$ **(e)** $\lambda = (5, 5, 3, 3, 0)$ **(f)** $\lambda = (4, 2.6, 2.6, 0.4, 0.4)$

Figure 5.3 Graph representations based on the Laplacian spectrum. Green (red) links represent the positive (negative) edge between two nodes. λ denotes the Laplacian spectra (the Eigenvalues of the Laplacian matrix) of the graph. Intuitively, different network topologies have different connectivities and are accompanied by a distinct set of Eigenvalues.

5.2.1). This modification ensures the positive-semidefinite property of the Laplacian matrix and the benefits therein.[1] I then use the signed Laplacian spectra (the Eigenvalues of the Laplacian matrix) to encode the network structure of the equity market at each time point. The Laplacian spectra of a graph essentially represent the frequency domain of discrete networks and directly link to the global structures, properties, and motif of a network [220–222]. As a result, the encoded representation of the network structure also incorporates the changing market condition, economic

[1]A detailed discussion of using Eigenvalues and spectrum to represent heterogeneous networks and the advantages can be found in [220].

environment, and uncertainties in a macro context. Moreover, to capture the dynamic evolution and detect change points in networks, I construct the network factor "Z-score" by measuring the difference between the current network state and network states of previous months.

I evaluate the importance and implication of the network factor in equity pricing by examining the factor in relation to the market, macro, and other asset pricing factors. The empirical analysis provides several interesting findings. First, Z-score aligns well with significant market events, such as the 1987 Black Monday, 2008 financial crisis, and COVID-19. Those major events generate considerable and asymmetric impacts on different firms, significantly changing the network structure. Moreover, the correlations of Z-score with VIX and EPU are only 0.15 and 0.19, respectively. This finding suggests that Z-score, to a large extent, reveals information different from volatility and uncertainty.

Second, I incorporate the proposed network factor into conventional asset pricing models and show that network is an important equity pricing factor. Following the literature, I compare multiple factor models to assess whether adding network factor can improve pricing models [223–225]. The two-pass cross-sectional regression [226] and three-pass estimator [225] produce significant and negative risk premia for the network factor. The time-series R^2 for the network factor is much higher than that of other nontradable factors, such as the macro-finance factors of [227] and the consumption growth factor of [228]. The Wald-test rejects the null hypothesis that network is a weak factor, confirming that network is a vital pricing factor for stocks. In addition, time-series tests show that the network factor can enhance the return predictability and reduce mispricing. Results are robust for different asset portfolios with multiple performance metrics.

Third, cross-sectional analysis shows that firms with the positive (negative) sensitivity to network changes have lower (higher) future returns. When a given firm

reacts to the network changes in the same direction, this synchronization suggests that this firm is capable to adjust quickly and less vulnerable. Investors, therefore, treasure such adaptability. As a result, the demand for those stocks increases, prices increase, and the expected returns decrease. Moreover, I discover that this negative relation is more significant when the market network experiences substantial changes.

The proposed methodology for the network factor and its associated empirical findings contribute to the existing literature in a number of ways. First, this chapter proposes a novel way to capture and quantify the network dynamics in financial markets. Current works predominantly use the historical return correlation [60, 80, 217], industry sector similarity [59], and customer-supplier network [229] to represent financial networks and mainly analyze how information flows among firms and institutions during crisis and pandemic situations [81–84, 230, 231]. By contrast, I apply the state-of-the-art Laplacian spectrum analysis technique in the equity market to represent the network and construct the global network factor. Compared to the unsigned network embedding approach, the proposed signed Laplacian approach better represents the information property of the equity network, generates higher risk premia, and better explains the return difference among assets with improved R^2. The general framework for the signed Laplacian spectrum detailed in this chapter can also extend for performing graph cuts, identifying proper equity market clusters, and analyzing other financial markets' network structures where negative connections exist.

Second, I explicitly construct a network index Z-score to track the aggregated changes at the market level rather than the pairwise correlations at the firm level. Z-score provides direct information to market participants about the network state of the equity market. As a result, Z-score can be considered as a complementary to the existing macro indices and asset pricing factors.

Third, my empirical tests show that network is a vital pricing factor and uncover how firms respond to network changes. The result also suggests that firms' exposure to network changes is more significant when the market network is volatile. Long-short portfolio analysis shows that the difference between the smallest network β and highest network β has an inverted U shape characteristics during different market conditions. It is significantly negative when the market network is stable with very small change and is significantly positive when the market network is volatile with very large change. The result is robust after controlling for market, size, value, and momentum. These findings shed light on the portfolio diversification opportunity to investors.

The remainder of this chapter proceeds as follows. Section 5.2 first presents the method of learning equity market network representation for the signed graph and then details the construction of the network factor. Section 5.3 discusses the data and the results from empirical analysis. This section provides a thorough examination of the network factor's application and validity, and evaluates its significance in equity pricing. Finally, Section 5.4 summarizes the findings and discuss future direction in regard to this method.

5.2 Methodology

5.2.1 Equity Market Network Representation with Dynamic Signed Graph

The network structure of the U.S. equity market at time t can be represented by a weighted graph $G_t = (V_t, E_t, W_t)$, where each node $v \in V_t$ represents a firm, each edge $e_{ij} \in E_t$ shows a connection between firms i and j at time t, and $W_t \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix representing the quantitative proximity among firms at time t with weights w_{ij} for all $e_{ij} \in E_t$, otherwise $w_{ij} = 0$. The network structure in

each time point (t) represents the market state at that time.[2] A proper representation of this network and the efficient extraction of network information are necessary to understand the equity market as a whole [60,217]. An efficient equity market network representation over time will reveal how information dissipates along with networks and how firms react in times of crises and shocks, i.e., major economic and political events.

Graph Laplacian and the network spectrum based analysis are heavily applied in computer science and social science to represent and extract information from networks. It has its origin from the graph spectrum theory [232] and can reveal the connectivity, local and global structures, and motifs of networks [220]. Encouraged by their success, in this work, I apply the Laplacian spectrum to extract valuable properties of the equity network. For a simple (unweighted, undirected with no multiple edges incident to the same two vertices) graph $G = (V, E)$: the set of nodes V with $|V| = N$, edges $E \subseteq V \times V$, and adjacency matrix $A \in \mathbb{R}^{N \times N}$ given by $a_{ij} = 1$ if i and j are adjacent and $a_{ij} = 0$ otherwise, the Laplacian matrix L is defined as follows:

$$L_{ij} = \begin{cases} deg(i) & i = j \\ -1 & if\ i\ and\ j\ are\ adjacent\ (i \sim j) \\ 0 & otherwise \end{cases}$$

where $deg(i)$ is the degree of node i, i.e., the number of edges incident to node i. Let, $D \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the diagonal elements as $(deg(1), ..., deg(n))$, L is rewritten as

$$L = D - A \tag{5.1}$$

[2]I drop the time subscript for brevity when skipping time subscript does not influence the basic understanding of the described procedure.

The eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}$ of the Laplacian matrix L constitute graph's spectrum. L is symmetric and positive-semidefinite; that is, $\lambda_i \geq 0$ for all i [232]. The eigenvalues of the Laplacian matrix reveal the critical properties of the graph. The smallest non-zero eigenvalue of L represents the first spectral gap. The second smallest eigenvalue of L represents the algebraic connectivity of the graph [221]. When the graph is connected, the algebraic connectivity (Fiedler value) is the same as the first spectral gap. The eigenvectors associated with the K smallest eigenvalues of the graph Laplacian capture critical information content and provide a low dimensional graph embedding [222].[3]

A weighted undirected graph G defines *weight matrix* $W \in \mathbb{R}^{N \times N}$, where $w_{ij} = w_{ji}$ if i and j are adjacent, and $w_{ij} = 0$ if i is not adjacent to j in G. $D_{ii} = \sum_{j \sim i} w_{ij}$ is the degree of i. The unnormalized Laplacian matrix of a weighted graph G is similar to equation (5.1), as $L = D - W$. The positive-semidefinite property of the graph Laplacian spectrum also holds when the weighted graph is non-negative [232].

Following the literature, I use the correlation of historical returns to measure firms' proximity and capture the U.S. equity market network [233, 234]. Unlike previous studies, instead of removing negative edges or using absolute values, I use both positive and negative correlations to construct a signed network (see details about network construction in the Appendix C.1). I calculate the modified signed Laplacian matrix following [218] and [219] as $\bar{L} = \bar{D} - W$, where $\bar{D}$ is:

$$\bar{D}_{ii} = \sum_{j=1}^{N} |W_{ij}| \tag{5.2}$$

Figure 5.4 shows a simple example of how the adjacency matrix, degree matrix, and Laplacian matrix are constructed from a signed graph based on the method.

[3]Graph embedding is the process of representing an entire graph in a vector space.

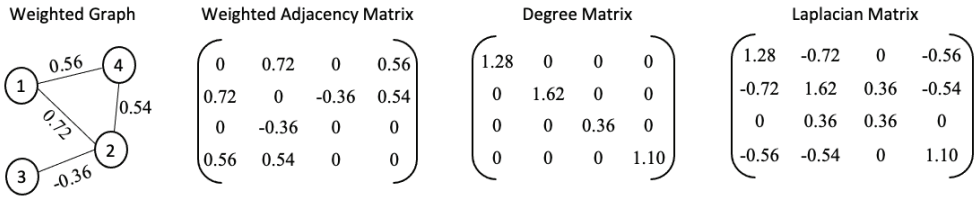


Figure 5.4 An undirected weighted signed graph, its adjacency matrix, degree matrix, and Laplacian matrix calculated based on the proposed model.

The modified diagonal degree matrix $\bar{D}$ ensures that in a signed, weighted, undirected graph G, the modified signed Laplacian matrix $\bar{L}$ is positive-semidefinite. That is, the eignevalues $\lambda_0 \cdots \lambda_{N-1}$ of $\bar{L}$ is non-negative. This can be proved with the incidence matrix of the signed G. For a signed graph with weights, I follow [235] to define the $|E| \times |V|$ oriented incidence matrix of an weighted graph as follows:

$$S_{i\sim j,v} = \begin{cases} +\sqrt{|W_{i,j}|} & if\ v = i \\ -sgn(W_{ij})\sqrt{|W_{ij}|} & if\ v = j \\ 0 & otherwise \end{cases} \tag{5.3}$$

The diagonal and off-diagonal entries of the product $S^\top S \in R^{V\times V}$ are:

$$\begin{aligned} (S^\top S)_{ii} &= \sum_{j\sim i} |W_{ij}| \\ (S^\top S)_{ij} &= -W_{ij} \end{aligned} \tag{5.4}$$

where $S^\top$ is the matrix transpose of S. This shows $S^\top S = \bar{L}$. The eigendecomposition of $\bar{L}$ generates N pairs of eigenvectors $x_i \in \mathbb{R}^V$ and the corresponding eigenvalues λ_i

with the following condition:

$$\begin{aligned}\lambda_i &= x_i^\top \bar{L} x_i \\ &= x_i^\top S^\top S x_i \\ &= (S x_i)^\top (S x_i)\end{aligned} \tag{5.5}$$

Essentially, λ_i is the inner product of Sx_i with itself. $\forall 1 \leq i \leq N, \lambda_i \geq 0$. Therefore, Equation (5.5) ensures that the Laplacian of weighted graph $\bar{L}$ is also positive-semidefinite with non-negative eigenvalues.

I take two steps to construct the network representation for equity market at each time point t: first solve the generalized eigenvalue problem $\bar{L}x = \lambda x$, and then choose the k lowest eigenvalues of $\bar{L}$ to create the Laplacian spectrum $\Lambda_t \in \mathbb{R}^k$ of G_t for representing the network. In comparison to PCA or the spectrum of the graph adjacency matrix, the spectrum of the Laplacian matrix gets reordered [236]. That is, instead of larger end, the lowest-end of the eigenvalues spectrum represent the highest information property. An efficient representation of the network structure does not require all N eigenvalues. [221] and [222] show that, a sufficiently large enough k lowest eigenvalues of the Laplacian matrix are sufficient to capture all necessary graph properties for building an efficient embedding. In addition, the Laplacian eigenmaps theory suggests that the eigenvectors associated with the largest eigenvalues encode high-frequency changes among nodes and noise. Therefore, ignoring the $(N - k)$ largest eigenvalues disentangles the noise from the useful network information [222].

The value hyperparameter k depends on the structure and rank of the data. In order to identify the k, I use the eigengap heuristic from spectrul clustering literature [237–239]. In this process, the goal is to identify the k that ensures $\lambda_i, ..., \lambda_k$ are very

small but λ_{k+1} is relatively large.

$$k = \arg\max_{i \leq k \leq N} |\lambda_{i+1} - \lambda_i|$$

This denote identifying the largest eigengap in the laplacian spectrum and using the eigenvalues before that to represent the network. The rational behind this is the perturbation theory for identifying k completely disconnected cluster. More details can be referred to [232, 238, 239]. For robustness test, in addition to using eigengap heuristic, I also check with hyperparameter $k = 10, 15$, and 20. The sensitivity tests of the network factor with respect to k are provided in the Appendix C.2.

5.2.2 Constructing the Network Factor Z-score

The equity market network embedding at each time point provides a snapshot of the market structure of that time. To track the dynamic evolution of the network over time, I compare the current network embedding with the benchmark, i.e., normal graph behavior during a context window (m), and the difference is captured as Z-score. In computer science literature, Z-score is a widely used measurement for the change point detection [214,240,241]. To calculate Z-score, I perform the following three-step procedure on the Laplacian spectra in the context window m:

- For each time period t, construct a context matrix $C_t \in \mathbb{R}^{k \times m}$ by concatenating previous m spectrum vectors of time t

$$C_t = [\Lambda_{t-m}\ \Lambda_{t-m+1}\ \cdots\ \Lambda_{t-1}], \tag{5.6}$$

  where m is the window size and $\Lambda_t = \sigma[\lambda_i \cdots \lambda_k]^\top$ represents the normalized Laplacian spectrum at time t with the k lowest eigenvalues of L and the normalization operator σ.

- Apply Singular value decomposition (SVD) to decompose $C_t = U_t \Sigma_t V_t^\top$ and obtain the left singular matrix U_t, singular value matrix Σ_t, and the right singular matrix V_t at time t. Use the first left singular vector $(U_t)_{:1}$ as the current context vector $\overline{\Lambda_t} = (U_t)_{:1}$. The current context vector represents the network's normal behavior in the time window ending at t. This process is also equivalent to obtain the weighted average of m vectors in the rolling window [240].

- Calculate the difference between current network structure Λ_t and normal network structure $\overline{\Lambda}_t$ as:

$$Z_t = 1 - \frac{\Lambda_t^\top \overline{\Lambda}_t}{||\Lambda_t||_2||\overline{\Lambda}_t||_2} = 1 - \Lambda_t^\top \overline{\Lambda}_t = 1 - cos\theta \tag{5.7}$$

Equation (5.7) shows that Z-score essentially calculates the cosine distance between Λ and $\overline{\Lambda}$ with a bound $[0 \leq Z \leq 1]$. A high Z-score reflects a significant evolution of the current spectrum from the normal spectrum. The choice of m depends on the application objective and the property of a network. A large m captures the impact of a long market evolution and business cycle, whereas a small m helps to identify the impact of shocks in the market. A sensitivity analysis of different m is presented in the Appendix C.2.

5.3 Empirical Results

This section details the network factor constructed by the proposed signed Laplacian-based methodology (see Section 5.2) and shows its implication in the U.S. equity market. First, I provide the data description and display the time-series evolution of the network factor during the period from 1960 to 2020. Second, I examine the relevance of the network factor in the broader macroeconomic environment and, with a battery of tests, show its significance in equity pricing. Finally, I evaluate firms' exposure to network risk and discuss the implications.

5.3.1 Data

To capture network structure and compute monthly Z-score, I use the daily returns from the Center for Research in Security Prices (CRSP) of all stocks listed in the NYSE, AMEX, and NASDAQ. The time period starts from January 1960 to December 2020, totaling 720 months. Z-score is computed at the end of each month based on the daily returns of all available stocks during the month. For all periods, I remove the assets with missing values to ensure that all return data in the given window

have complete values. After this filtering, there are about 30,000 firms in total and on average, about 3,800 firms in each month. In my sample, the minimum number of firms for a given month is March 1960 with 716 firms, and the maximum one is December 1997 with 5400 firms. In all analyses, equity returns are after risk free rates, which is the one-month Treasury bill rates.

For constructing the initial signed network, I use the correlation among firms' historical market returns. For each month, I calculate the end-of-month correlation ρ_{ij} using daily returns in that month. The top panel in figure 5.5 shows the average correlation coefficient among studied stock returns in a given month. On average, the mean correlation among stocks is 0.10. However, in more recent years, the mean correlation coefficient increased to almost 0.16. The middle and bottom panels in figure 5.5 show the time-trend for the fractions of positive edges and negative edges, respectively. The spikes in positive edges are associated with the spikes in the mean correlation. Figure 5.6 shows the distribution of positive and negative edges during the studied period. In the equity network, among all possible connections (positive, negative, and no-connection), on average, the percentage of positive edges is 12% and negative edges is 2.5%. Compared to negative edges, the distribution of positive edges has a long tail. In some months, the percentage of positive edges reaches 70%.

The significance of the proposed network factor is evaluated using multiple factor models. Following the asset pricing literature, I include multiple important and well-acknowledged factors in my analysis. These are: Capital Asset Pricing Model (CAPM) of [52, 170] that use the value-weighted market return; Fama and French three-factor (FF3) model that extends CAPM with the size (SMB) and value (HML) factors [177]; Carhart four-factor (FFC) model that incorporates momentum factor to the FF3 [179]; Pástor and Stambaug liquidity-factor (FFPS) model that adds liquidity to the FF3 [242]; Fama and French five-factor (FF5) model that adds operating profitability (RMW) and investment (CMA) to the FF3 [49]; industrial

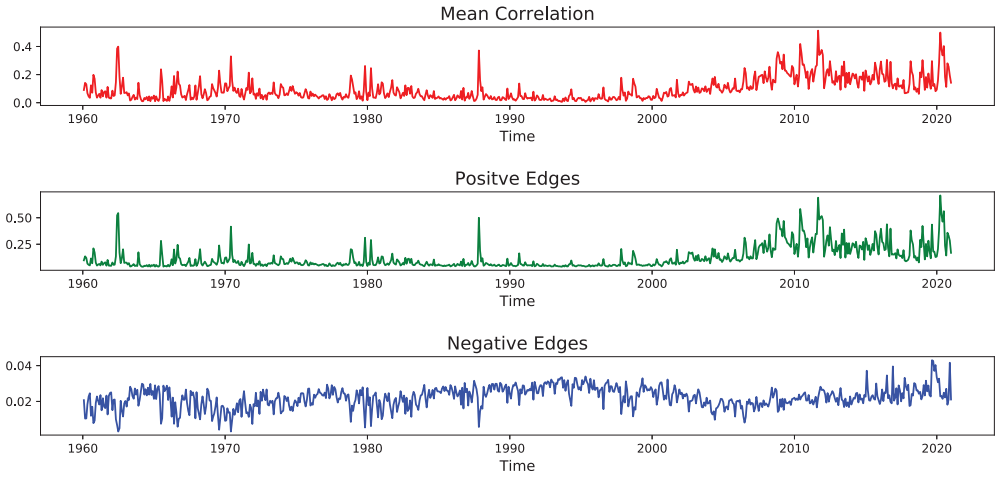


Figure 5.5 The timeline of mean correlation between equity returns, the fraction of positive edges in a given month, and the fraction of negative edges in a given month, respectively.

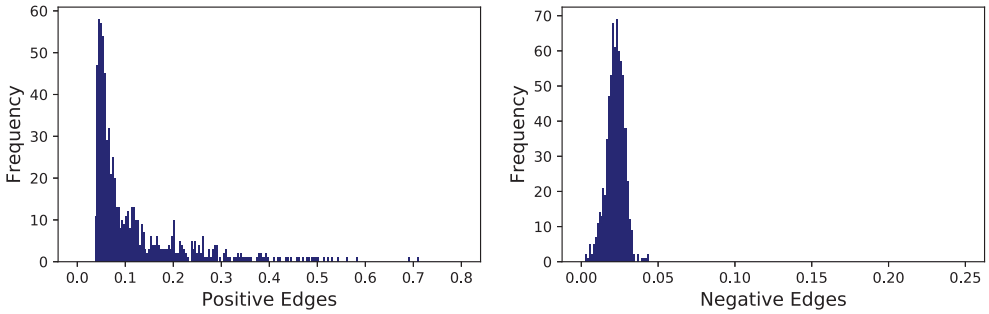


Figure 5.6 Histogram of the fraction of positive and negative edges between January 1960 and December 2020.

production growth (IP); Ludvigson and Ng macro-finance factor (LN) that uses principal components of 279 macro-finance variables [227]. In addition, I take more factors into the consideration, including the first three principal components following [225], intermediary capital factor by [243], and consumption based factor by [228].[4]

[4]The data for CAPM, FF3, FF4, and FF5 are obtained from Kenneth French's website; for liquidity, from Lubos Pastor's website; for IP, from the Federal Reserve Bank of St. Louis; for LN, Sydney Ludvigson's website; for intermediary capital, from Asaf Manela's website; for consumption based factor, from Toby Moskowitz's website.

To test the asset pricing models, I consider 173 anomaly portfolios. These anomaly portfolios are the value-weighted monthly excess returns: the 30 IND (industry) portfolios, 25 size-AC (accruals) portfolios, 25 size-β (market beta) portfolios, 25 size-RVar (residual variance) portfolios, 35 size-CI (abnormal capital investment) portfolios, 25 size-NI (abnormal profitability) portfolios, and 8 D10-1 (high minus low decile) portfolios. These portfolios capture a vast cross-section of return anomalies, pose a great challenge to existing asset pricing models, and are often used as the benchmark portfolios for evaluating and comparing asset pricing models [223–225]. In addition, the data for these portfolios is easily accessible from Kenneth French's website.[5]

To evaluate the significance of network factors in relation to the macroeconomic environment, I compute its correlation with multiple market and macroeconomic indicators. These include the monthly S&P-500 index returns (S&P), monthly Russell-3000 index returns (RUT), Chicago Board Options Exchange's volatility index (VIX), monthly industrial production total index (INDPRO), monthly consumer price index (CPI), monthly unemployment rate (UNRATE), and monthly U.S. economic policy uncertainty index (EPU).[6]

5.3.2 Network Factor and Macroeconomic Environment

The graph Laplacian provides essential information about the network structure at a specific time point. The Z-score from the graph Laplacian spectrum aligns with major events in the past fifty years. Figure 5.7 exhibits the Z-score over time in solid lines, major events labeled by the red star, and financial crisis periods are highlighted in gray blocks. The Z-score spikes during 1971 President Nixon's announcement to break

[5]For a detailed description of all portfolio construction methodology, please see Kenneth French's website: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

[6]The data for S&P, RUT, and VIX are obtained from Yahoo Finance; for INDPRO, CPI, and UNRATE, from the Federal Reserve Bank of St. Louis; and for EPU, from the economic policy uncertainty website: https://www.policyuncertainty.com/.

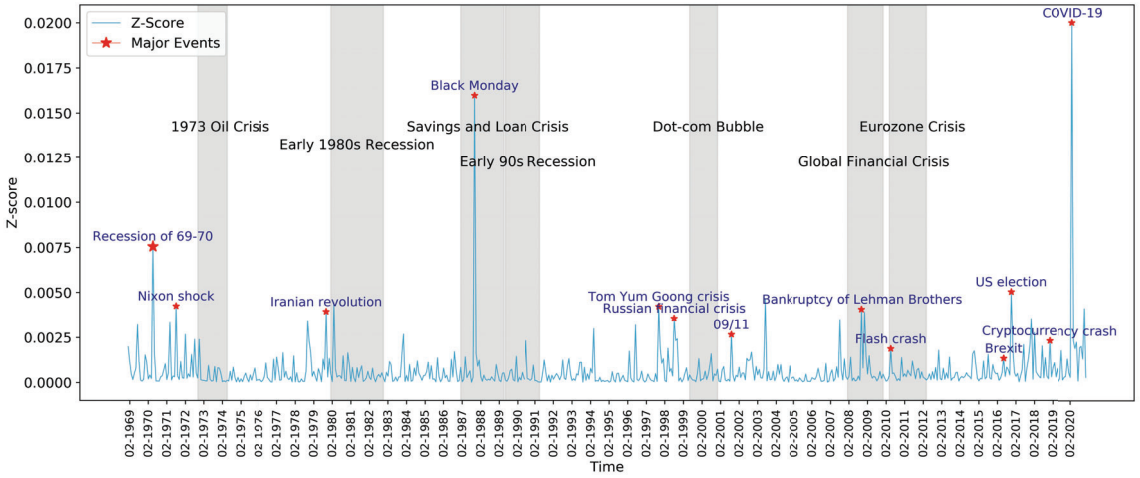


Figure 5.7 Z-score along with major events from Jan-1969 to Dec-2020. Financial crisis periods are highlighted in gray blocks and major events are marked by red star.

up Bretton Woods, 1987 Black Monday, 2001 Dot-com bubble, 2008 financial crisis, 2020 COVID-19, and other major events, which have generated apparent impacts on the market network.

It is important to note that the effects of sudden shocks, such as Black Monday, 09/11, and the 2016 U.S. election, are more prominent in the calculated Z-score than the prolonged recession and economic stagnation. When the market experiences recession for an extended period, the change in the Laplacian spectrum from one month to the next month is low. In such volatile periods, the volatility becomes the "normal" market behavior and this is why the magnitude in network change shrinks.[7]

Next, I examine whether the network indicator (Z-score) is highly correlated with other macro indicators or reveals additional information complementary to existing macro indicators. Table 5.1 reports the correlation matrix of Z-score and other macro indicators and shows that the Z-score has a negative relationship with

[7]This behavior also attributes to the use of daily return data for calculating monthly Z-score. During the experiment, I find that the Z-score changes driven by sudden shocks are less prominent once I switch to use monthly returns to construct monthly network indicator. In this case, changes due to the long-term recession become more prominent and severe. For brevity, I did not report the results of correlation calculation based on monthly return data. Results are available upon request.

Table 5.1 Network Factor and Macroeconomic Indicators Correlation Matrix

	S&P	VIX	RUT	EPU	INDPRO	CPI	UNRATE
VIX	−0.65						
RUT	0.81	−0.59					
EPU	−0.28	0.24	−0.26				
INDPRO	−0.07	0.06	−0.04	−0.06			
CPI	−0.03	0.05	0.01	0.05	0.16		
UNRATE	0.09	−0.08	0.08	0.01	−0.70	−0.17	
Network	−0.23	0.15	−0.23	0.19	0.27	0.04	−0.56

Note: This table reports the correlation matrix of the proposed network factor and other macroeconomic indicators.

Table 5.2 Asset Pricing Factors Correlation Matrix

	MKT	SMB	HML	RMW	CMA	Mom.	Liq.	LN1	LN2	LN3	Int. cap.	IP gr.	Cons. gr.
SMB	0.29												
HML	−0.22	−0.03											
RMW	−0.21	−0.34	0.08										
CMA	−0.37	−0.10	0.68	−0.02									
Mom.	−0.16	−0.06	−0.20	0.11	−0.03								
Liq.	−0.02	−0.01	0.05	0.03	0.03	−0.03							
LN1	−0.09	−0.05	−0.06	0.04	0.03	−0.08	−0.03						
LN2	0.28	0.13	−0.08	0.00	−0.08	−0.14	0.00	0.02					
LN3	0.01	−0.01	0.01	0.03	−0.02	0.00	−0.08	0.01	−0.01				
Int. cap.	0.74	0.11	0.01	−0.16	−0.19	−0.26	−0.01	−0.04	0.18	0.08			
IP gr.	−0.04	−0.01	0.05	−0.03	0.00	0.05	0.00	−0.36	−0.12	0.03	−0.03		
Cons. gr.	0.03	0.14	0.04	−0.07	−0.02	0.00	0.08	−0.19	−0.09	−0.06	0.00	0.57	
Network	−0.22	−0.14	−0.08	−0.03	0.06	0.06	0.02	0.12	−0.13	0.01	−0.12	0.26	0.14

Note: This table reports the bivariate correlation between asset pricing factors including the proposed network factor.

S&P, RUT, and UNRATE and a positive relationship with VIX, INDPRO, and EPU. This is intuitive as Z-scores indicate the magnitude of network changes and can be identified as a measurement of network uncertainty and risk. When market volatility (VIX) and policy uncertainty (EPU) are high, firms tend to adjust more, and consequently, the network structure changes more. However, the magnitude of the positive correlations is only 0.15 and 0.19 for VIX and EPU, respectively. This finding suggests that Z-score reflects some information other than volatility and uncertainty, complementing current market indicators.

5.3.3 Network Factor for Equity Pricing

Here I evaluate the significance of the proposed network factor in relation to existing asset pricing models. To begin with, I first analyze correlation statistics. Table 5.2 reports the bivariate time-series correlation between all studied asset pricing factors, including the network factor. The network factor has a high and negative correlation with the market (-0.22) and a high and positive correlation with industrial production growth (0.26). Except for the CMA, the network factor has a negative correlation coefficient with all other FF5 factors. However, the magnitude is small, for SMB -0.14, for HML -0.08, and for RMW -0.03. Among the other non-traded factors, the network factor is negatively correlated with LN2 and intermediary capital factor, and positively correlated with LN1, industrial production growth and aggregate consummation growth. More importantly, the network factor is not highly correlated with any other existing assets pricing factor. This indicates that network factor captures additional information from the market that is not captured by existing factors.

Although correlation statistics reported in Tables 5.1 and 5.2 indicate additional and complementary information of network factor, it still needs formal statistical tests to validate the significance of network factor in asset pricing. Therefore, I perform a battery of tests to compare the network factor's significance with other existing factors. These include Fama-MacBeth cross-sectional regression [226], Giglio and Xiu three-pass estimator [225], and time-series return predictability [223].

Two-Pass Cross-Sectional Estimator The Fama MacBeth two-pass cross-sectional regression estimates a factor risk premium in two steps [226]. First, it estimates the assets risk exposure β by performing a time series regression of each

asset's excess return onto the factor as:

$$R_t = \alpha + \beta f_t + \epsilon_t, \qquad t = 1, ..., T.$$

where f_t is the factor at time t and R_t is a vector of returns on N test assets at time t. At the second step, it estimates the risk premium by performing a cross-sectional regression of the expected cross-section returns on the estimated β.

$$\widehat{\gamma} = (\widehat{\beta}^\top \widehat{\beta})^{-1} \widehat{\beta}^\top R,$$

A rolling time-series regression can estimate the changing β throughout the sample period. For example, [226] use prior 5-year rolling-regressions to estimate beta for month t. One can also use two-pass cross-regression regression to estimate full-sample β. In this subsection, I use the latter approach for simplicity. A rolling-regression approach with changing β is used for analyzing firms' exposure to network factor in subsection 5.3.4.

Table 5.3 reports the results from two-pass cross-sectional regression for the proposed network factor and other traditional factors. The test includes all 173 portfolio ($n = 173$) over 680 months ($T = 680$). The first column reports the time-series average return for the tradable factors. This is the model-free estimator for a factor risk premia and is only available for tradable factors. For each factor, I estimate the risk premium without any additional control factors (No control), controlling for the market return (w/$R - m$), and controlling for the Fama-French three factors (w/FF3). For robustness, I analyze two versions of the network factor. 'Network' is the proposed network factor with Laplacian signed spectra with $m = 36$ in the context matrix C_t of equation (5.6). 'Network (m=12)' is a alternative network

Table 5.3 Two-Pass Regression: Empirical Results

		No control		w/R_m		w/FF3	
Factors	Avg. Ret.	γ	stderr	γ	stderr	γ	stderr
Market	0.57	0.64^{***}	(0.18)	0.64^{***}	(0.18)	0.58^{***}	(0.17)
SMB	0.23	0.66^{***}	(0.19)	0.15	(0.12)	0.07	(0.08)
HML	0.25	-1.33^{***}	(0.44)	0.48^{***}	(0.13)	0.48^{***}	(0.13)
Momentum	0.69	-2.66^{***}	(0.75)	0.27	(0.29)	1.34^{***}	(0.26)
RMW	0.25	-0.12	(0.14)	0.00	(0.14)	0.27^{**}	(0.11)
CMA	0.26	-0.63^{***}	(0.18)	0.29^{***}	(0.10)	0.29^{***}	(0.09)
Liquidity		-0.02^{**}	(0.01)	-0.00	(0.00)	-0.01	(0.00)
Interm. cap.		1.23^{***}	(0.39)	1.95^{***}	(0.54)	-0.07	(0.49)
IP growth		-2.67^{***}	(0.74)	-0.03	(0.11)	-0.24^{**}	(0.11)
LN PC1		0.58^{***}	(0.17)	0.47^{**}	(0.19)	0.19	(0.17)
LN PC2		0.15	(0.12)	0.11^{***}	(0.04)	0.20^{*}	(0.12)
LN PC3		0.48^{***}	(0.13)	-0.08^{**}	(0.03)	0.05	(0.11)
Cons. growth		1.35^{***}	(0.38)	0.13	(0.17)	-0.03	(0.11)
Network (m=36)		-0.22^{***}	(0.06)	-0.14^{***}	(0.04)	-0.08^{***}	(0.03)
Network (m=12)		-0.24^{***}	(0.07)	-0.15^{***}	(0.04)	-0.08^{***}	(0.03)
Network (+)		-0.14^{***}	(0.39)	-0.11^{***}	(0.36)	-0.06^{***}	(0.31)

Note: This table reports the risk premia estimates for each factor using two-pass cross-sectional regression with no control factor in the model, with the market as control, and with the Fama-French three factors as control, respectively; "Avg. Ret." is the time-series average return of the tradable factors; Network (m=36) is the proposed signed network factor with context window m=36, Network (m=12) is the signed network factor with context window m=12, and Network (+) is the unsigned network factor considering all connection as positive; ***, **, * are significant at the 1%, 5%, and 10% significance level, respectively.

factor with $m = 12$. Finally, to evaluate the contribution of including both positive and negative connections among firms, in Network (+), I consider the network factor constructed based on the unsigned network structure, i.e., considering the absolute value of network edge.

For most factors, the estimated risk premia are closely comparable to the values reported in [225]. For example, in my analysis, the time-series average return for Market factor is 57bp, the risk premia with no-control is 64bp, adding the market induces no changes, and adding SMB and HML gives 58bp. In [225], the reported values are 50bp, 59bp, 59bp, and 49bp, respectively. For SMB, my result for time-

series average is 23bp, risk premium with no-control is 66bp, adding the market gives 15bp, and adding FF3 gives 7bp, compariing with the reported values of 23bp, 63bp, 16bp and 13bp in [225], respectively. However, for a few factors, the difference is visible. For example, according to my analysis, the risk premia estimator for intermediary capital factor is 123bp without controls, 195bp after controlling on the market, and -7bp after controlling for FF3. In comparison, [225] report 73bp, -18bp, and 10bp, respectively.[8]

The network factor is significant in all three two-pass model specifications at 1% level of significance. The network factor risk premium without controls is -24bp. After controlling the market return, the risk premium is -14bp, and the risk premium with controlling for the market, SMB, and HML return is -8bp. The result is robust with a shorter context window (m=12). In this case, the risk premia are -24bp, -15bp, and -8bp, respectively. In comparison to other non-tradable factors, the proposed network factor is much more significant. Controlling for the market, SMB, and HML, I find none of the liquidity, intermediary capital, LN PC1, LN PC3, and consumption growth factors are significant. The industrial production growth factor is significant at 5%, and the LN PC2 factor is significant at 10% significance level. The insignificance of these factors on risk premia according to two-pass cross-sectional regression is also reported in [225].

The improvement in information gain by independently considering both positive and negative edges is evident in the difference between the risk premium of Networks (m=36) and the only positive correlation Networks (+). The risk premia associated with a network factor based only on positive edges are -14bp, -11bp, and -6bp in the three two-pass model specifications. The proposed signed network factor

[8]This difference is expected as the data used in these two works are slightly different. In my analysis, I use 173 portfolios whereas [225] use 202 portfolios. The time period for my analysis is from January 1960 to December 2020, whereas in [225], it is from July 1963 to December 2015.

increases the magnitude of the impact on equity premium by 57%, 27%, and 33%, respectively. In addition, the standard error of the two-pass specification is also significantly higher for only the positive network factor. These two observations signify that, by incorporating both positive and negative networks, we can specify a network factor that is more stable and better identifies the information property of the equity network over time.

Three-Pass Estimator Although the evidence from two-pass cross-sectional regression signifies the importance of network factor in equity pricing, the [226] technique is often criticized for its associated bias. The two-pass cross-sectional regression is affected by the omitted variable bias in the time-series and cross-sectional steps [225]. To avoid the omitted variable bias, [225] recently propose a three-pass method that produces valid estimates even when not all factors in the model are specified or observed. I apply the three-pass estimator to corroborate the two-pass regression results for validating the proposed network factor. The three-pass estimator of [225] overcomes the omitted variable bias in the two-pass regressions and mimicking-portfolio estimator by identifying rotation invariant risk premium of an observed factor. In addition, it is also a powerful tool for identifying measurement error in an observed factor and detecting spurious or useless factor [225]. Therefore, the three-pass estimator is a better choice to evaluate the usefulness of my proposed network factor in the asset pricing factor universe.

As the name suggests, the risk premium for a factor in the three-pass estimator is estimated using three steps. First, perform the principal component analysis (PCA) on the matrix $n^{-1}T^{-1}\bar{R}^{\top}\bar{R}$ and identify the latent factors: $\widehat{V} = T^{1/2}(\xi_1 : \xi_2 : ... : \xi_{\hat{p}})^{\top}$ of the normalized eigenvectors (principal components) corresponding to the largest $\hat{p}$ eigenvalues. Second, obtain the risk premia of the estimated latent factors by performing a cross-sectional ordinary least square (OLS) regression on average

returns. Third, identify the relation between observed factor g_t and the estimated latent factors by regressing the time-series $\bar{G} = (g_1, g_2, \dots g_T)^\top$ onto the principle components of the co-variance matrix $\bar{R}^\top \bar{R}$. This regression operation essentially projects $\bar{G}$ onto the principal components of $\widehat{V}$ because $(\widehat{V}\widehat{V}^\top)^{-1} = \frac{1}{T}\mathbb{I}_{\hat{p}\times\hat{p}}$.[9] This step also removes the measurement error from g_t. In the three-pass model, the risk premium of the observed factor is estimated as:

$$\hat{\gamma}_g = \bar{G}\widehat{V}^\top(\widehat{V}\widehat{V}^\top)^{-1}(\widehat{\beta}^\top\widehat{\beta})^{-1}\widehat{\beta}^\top\bar{r}, \tag{5.8}$$

where $\widehat{\beta} = T^{-1}\bar{R}\widehat{V}^\top$ are the loadings on the latent factor $\widehat{V}$.

In addition to estimating the risk premium of the network factor in light of three-pass regression of [225], I also analyze two significance tests proposed in [225], the time-series R^2 for observable factor R^2_g and Wald test for a weak g. The R^2_g measures the signal-to-noise ratio of the observed factor g and is calculated as follows:

$$R^2_g = \frac{\widehat{\eta}\widehat{V}\widehat{V}^\top\widehat{\eta}^\top}{\bar{G}\bar{G}^\top},$$

where $\widehat{\eta} = \bar{G}\widehat{V}^\top(\widehat{V}\widehat{V}^\top)^{-1} = \frac{1}{T}\bar{G}\widehat{V}^\top$ of the time-series regression of observed factors onto the latent factors. The Wald test evaluates the null hypothesis that an observed factor g is weak. Therefore, the Wald test allows me to examine whether the proposed network factor is weak or strong and validate the necessity of incorporating it in the asset pricing model. [225] showed that the parameters and Wald test statistics of three-pass estimator possesses asymptotic property and when n, $T \longrightarrow \infty$, the value converges. The detailed theorems and mathematical proof can be found in [225].

[9] I retains the term $(\widehat{V}\widehat{V}^\top)^{-1}$ to be consistent with the formula in [225].

Table 5.4 Three-Pass Regression: Empirical Results

Factors	γ	stderr	R_g^2	Wald test p-value
Market	0.58^{***}	(0.17)	99.25	0.00
SMB	0.22^{*}	(0.12)	97.92	0.00
HML	0.20^{**}	(0.10)	63.91	0.00
Momentum	0.60^{***}	(0.22)	71.09	0.00
RMW	0.08	(0.06)	49.01	0.00
CMA	0.08	(0.07)	53.36	0.00
Liquidity	-0.23^{*}	(0.13)	3.90	0.08
Interm. Cap	0.65^{***}	(0.23)	60.52	0.00
IP growth	-0.01	(0.01)	0.97	0.25
LN PC1	0.28^{*}	(0.17)	2.07	0.01
LN PC2	0.12	(0.15)	5.48	0.00
LN PC3	0.06	(0.10)	2.22	0.13
Cons. growth	0.01	(0.01)	2.90	0.00
Network (m=36)	-0.02^{***}	(0.00)	10.40	0.03
Network (m=12)	-0.02^{***}	(0.00)	10.03	0.04
Network (+)	-0.02^{***}	(0.00)	7.03	0.01

Note: This table reports the risk premia estimates for each factor using three pass estimator; the R^2 of the projection of factors onto the latent factors; and the p-value of the test that factor is weak; Network (m=36) is the proposed signed network factor with context window m=36, Network (m=12) is the signed network factor with context window m=12, and Network (+) is the unsigned network factor considering all connection as positive; ***, **, * denote the 1%, 5%, and 10% significance level, respectively.

Table 5.4 reports the results from three-pass regression. Following [225], I use seven principle components as latent factors. Similar to two-pass cross-sectional results, the risk premia estimators from most of the factors are closely comparable to the result reported in [225]. The risk premium of the proposed network factor is -2bp with 1% level of significance. Among all the competing tradable and nontradable factors, the risk premia of market, momentum, and intermediary capital are significant at 1% level of significance. The risk premium of HML is significant at the 5%

significant level, and risk premia of SMB, liquidity, and LN PC1 are significant at the 10% significance level.

The R_g^2 of the network factor is 10.40%. This is higher than other nontradable factors like liquidity, IP growth, all three macro factors, and consumption growth factor. In comparison to tradable factors, the R_g^2 of network factor is very low. [225] also report that the R_g^2 for nontradable factor is much lower and, for some cases, below 1%. They attribute this finding with less measurement error for tradable factors and associated noise of nontradable factors. The Wald test rejects the null hypothesis that the network factor is weak at 5% significance level. The result for the network factor is also robust for a shorter context window. The risk premium of the network factor (m=12) is also -2bp at 1% level of significance, R_g^2 is 10.03%. The null hypothesis of the weak network factor when m=12 is rejected at the 5% significance level.

Finally, the risk premium of only positive network factor according to three-pass regression is the same as the other two versions of network factors, and the null hypothesis of the weak factor is rejected at 1% significance level. This consistent finding corroborates the initial conjecture that the network itself is an important determinant of equity pricing. Even without a perfect specification (unsigned network without considering both positive and negative edges separately), the network factor can explain a significant portion of the variation of the equity return. However, the R_g^2 of network factor with only positive connections is 7.03. This is 3.37bp lower than the proposed signed network factor. This provides evidence of the additional contribution of considering a signed network over the unsigned network in equity analysis.

5.3.4 Cross Sectional Return Predictability

To better understand the implication of Z-score on equity prices, in this section, I conduct cross-sectional analyses in two ways: first sorting firms into portfolios based on the β of network factor and second incorporating the network factor β into the

Fama-MacBeth regressions with market, size, value, momentum βs. The analysis in this section differs from the analysis in Section 5.3.3 in two aspects. First, instead of using portfolio returns to test assets, I use the individual returns of stock here, including all the listed stocks in the NYSE, AMEX, and NASDAQ. Using individual stocks as the test assets allow me to understand how a given firm reacts to changes in the equity market network structure. Second, instead of a single β, I estimate changing βs. Particularly, I compute the sensitivity β for each factor using a 60-month rolling window with a minimum requirement of 15 months and then regress next-month future stock excess returns on those βs at each month. The independent variables are standardized by the cross sectional standard deviations to make results more consistent over time.

I begin the analysis by univariate sorting. At a given month, I sort firms into deciles based on β_Z (β of Z-score) estimated in last month and report the average returns for each portfolio over time. Results are summarized in Table 5.5. With full periods, there seems to be an inverted U shape curve of stock returns. The difference across ten groups is not large and the return difference (1 – 10) is not significant. As the network structure of equity market varies over time, the magnitude of structure change may play a role here. To investigate my argument, I simply divide time periods equally into four groups, i.e., network structure with very small change, small change, large change, and very large change.

Results are interesting. When the market network is stable with very small change, the future returns increase as β_Z increases, almost monotonically. The return difference between two portfolios (1 – 10) is -0.400% at five percent significance. When the market network is volatile with very large change, the relation becomes opposite. The return difference is 0.648% at one percent significance. Differently, the middle two groups seem to have the non-monotonic pattern, an inverted U shape.

Table 5.5 Univariate Portfolio Sorts by Network Beta

		Sub periods			
	Full periods	Very small change	Small change	Large change	Very large change
1	0.591	0.389	0.184	0.297	1.501
2	0.705	0.441	0.446	0.549	1.386
3	0.755	0.503	0.569	0.673	1.279
4	0.685	0.479	0.539	0.620	1.106
5	0.692	0.469	0.632	0.604	1.063
6	0.700	0.524	0.604	0.589	1.085
7	0.695	0.575	0.662	0.545	1.000
8	0.700	0.635	0.670	0.568	0.931
9	0.719	0.755	0.569	0.512	1.045
10	0.559	0.789	0.312	0.288	0.853
1-10	0.032	−0.400	−0.128	0.010	0.648
t-test	(−0.340)	(−2.220)	(−0.720)	(−0.060)	(−2.990)

Note: This table reports the average excess returns over time for each decile portfolio based on β_Z in the previous month. Excess returns are the monthly returns after risk-free rate and in unit of percent. 1 denotes firms with smallest/most negative β_Z whereas 10 denotes firms with largest/most positive β_Z. 1 – 10 stands for the long-short portfolio returns and its t-statistics are in parentheses. Full periods are from 1965 Oct to 2019 Dec. Sub periods are divided equally into four groups based on the level of Z-score, i.e., periods with very small change of market network, with small change, large change, and very large change.

For robustness, I further control for several conventional pricing factors including β_{mkt}, β_{smb}, β_{hml}, and β_{umd}. At each month, firms are independently sorted based on β_Z and one risk beta estimated in the previous month, forming firms into 5 by 5 groups. Then for each group based on a given risk beta, I aggregate to get the average return of five β_Z groups, compute the return difference (denoted as 1 – 5), and report the return difference for five groups with controls. As Table 5.6 shows, the pattern is generally consistent after controlling for multiple risk factors. The return difference 1 – 5 is significantly negative when the market network is stable with very small change and is significantly positive when the market network is volatile with very large change. However, the pattern seems to be stronger for the later one. As the positive spreads are significant at one percent in all cases whereas the

Table 5.6 Portfolio Sorts with Controls for Other Risk Factors

	β_{mkt}		β_{smb}		β_{hml}		β_{umd}	
	1-5	t-test	1-5	t-test	1-5	t-test	1-5	t-test
Panel A. Periods with very small change								
1 = small	−0.165	(−1.940)	−0.398	(−4.610)	−0.529	(−7.591)	−0.337	(−3.770)
2	−0.499	(−6.570)	−0.262	(−3.110)	−0.117	(−1.240)	−0.002	(−0.020)
3	−0.206	(−2.230)	−0.127	(−1.420)	−0.421	(−4.370)	−0.301	(−3.300)
4	−0.490	(−5.530)	−0.341	(−4.380)	−0.158	(−1.750)	−0.579	(−7.580)
5 = large	−0.645	(−7.790)	−0.462	(−5.140)	−0.397	(−3.900)	−0.368	(−4.620)
Panel B. Periods with very large change								
1 = small	0.628	(−6.340)	0.362	(−3.540)	0.365	(−4.350)	0.421	(−4.190)
2	0.501	(−5.400)	0.432	(−4.220)	0.352	(−3.970)	0.469	(−4.950)
3	0.282	(−2.950)	0.448	(−5.660)	0.445	(−4.280)	0.546	(−4.970)
4	0.519	(−5.240)	0.483	(−4.920)	0.546	(−5.010)	0.478	(−4.580)
5 = large	0.443	(−4.890)	0.525	(−5.620)	0.643	(−5.510)	0.460	(−5.700)

Note: This table reports the average return difference between firms with smallest and largest β_Z after controlling for other risk factors including β_{mkt}, β_{smb}, β_{hml}, and β_{umd}. 1 denotes firms with smallest/most negative betas whereas 5 denotes firms with largest/most positive betas. At each month, firms are independently sorted based on β_Z and one risk beta in the previous month, i.e., 5 by 5 groups. Then for each group based on a given risk beta, I aggregate to get the average return of five β_Z groups, compute the return difference, denoted as 1 – 5, and report the return difference for five groups with controls. t-statistics of the return differences are in parentheses. Only periods with very small network change (bottom) and very large network change (top) are included.

significance of negative spreads disappears in few cases. This finding suggests when the interconnection among firms change a lot, firms with fast adaptability (largest β_Z) are likely more attractive to investors and therefore are required with lower future returns. When the market structure is stable, such adaptability lose its benefit and firms moving against market change might offer additional investment opportunities.

Both patterns are reasonable. On the one hand, if a given firm responds to the change of network in the opposite direction, it offers an option as investment hedging against market conditions. Investors appreciate such hedging option and the demand of those stocks increases, which results in price increase and return decrease. If so, firms with low (negative) β_Z are expected to have lower future returns, i.e., the coefficient of β_Z should be positive. On the other hand, if a given firm reacts to the

Table 5.7 Fama-MacBeth Regression on Next Month Excess Return

Variables	1. Full-Sample	2. Low-Z	3. High-Z
β_Z	-0.047^{*}	0.034	-0.130^{***}
	(−1.665)	(1.073)	(−3.088)
β_{MKT}	−0.000	−0.049	0.049
	(−0.005)	(−0.450)	(0.394)
β_{SMB}	0.115	0.142	0.087
	(1.290)	(1.171)	(0.738)
β_{HML}	0.143^{*}	0.227^{*}	0.057
	(1.887)	(1.909)	(0.505)
β_{UMD}	-0.153^{**}	−0.132	-0.176^{**}
	(−2.478)	(−1.490)	(−2.210)
Constant	0.677^{***}	0.556^{***}	0.801^{***}
	(4.281)	(2.743)	(4.116)
Observations	3,355,977	1,722,132	1,633,845
R-squared	0.036	0.036	0.036
Number of groups	651	329	322

Note: This table reports the average value of Fama-MacBeth regressions. The dependent variable is the next-month excess returns. Independent variables are the sensitivity (β) of Z-score (β_Z), Fama-French three factors ($\beta_{MKT}, \beta_{SMB}, \beta_{HML}$) and momentum factor ($\beta_{UMD}$). Model (1) is for full sample period. Model (2) is when Z-score is low (below the median), and model (3) represents values when Z-score is high (above the median). Newey-West adjusted t-statistics are in parentheses. ***, **, * are significant at the 1%, 5%, and 10% significance level, respectively.

change of network in the same direction, it suggests that this firm is able to adjust to market changes quickly, less vulnerable with lower risks especially when macro environment is volatile. Investors therefore treasure such adaptability and require lower compensation. Given this situation, firms with high (positive) β_Z tend to have lower future returns, i.e., the coefficient of β_Z should be negative.

To substantiate this hypothesis, I further perform cross-sectional regression tests with multiple controls, i.e., market, size, value, and momentum. The result is reported

in Table 5.7. In Table 5.7 model (1), the coefficient of β_Z is significantly negative at 10%. In other words, stocks with great adjustment are valued more than hedging option, and their future returns are lower. I then divide the time periods into two groups: below the median Z-score (periods with small or no network changes) and above the median Z-score (periods with large network changes). In model (2), when network changes are minor, the significance of β_Z disappears. Conversely, in model (3) when there are big changes in network structure, the impact of the Z-score on firms' next month returns is more significant both economically and statistically. With an increase in β_Z by one standard deviation, future returns decrease by 13bps per month.

These findings are intuitive. When Z-score is low, the interconnection between firms is stable and firms do not change much. Its impact on equity prices is limited. On the contrary, firms adjust accordingly when Z-score is high, i.e., the network changes significantly due to exogenous shocks or endogenous strategic changes. As a result, the changes in the network affect the underlying performance of a firm and its equity prices. These results are robust across different hyperparameters and not influence by the number of eigenvalues k or the size of sliding window m. The robustness test results are presented in the Appendix C.2.

5.4 Conclusion

This chapter proposes a framework to quantify complex network structures consisting of both positive and negative correlations among firms. I construct a network index Z-score from the k-smallest eigenvalues of a well-designed signed graph Laplacian that treats positive and negative interconnection differently. This signed framework is technically more advanced and suitable for the complex network system and is robust for different parameters.

Empirical results show that Z-score is an ideal representation of the dynamic network structure in the U.S. equity market. It aligns with the major events in

the financial markets and contains valuable information other than market volatility, traditional asset pricing factors, and macroeconomic indicators. More importantly, the comprehensive analysis of factor models reveals the importance of network factor in the equity market, as it contains a significant risk premium for firms' network risk exposure. The network factor reduces model mispricing when used as an additional pricing factor to existing pricing models. The result also suggests that the network factor's impact be not the same across firms and over time. Different events affect the market network differently, and as a result, their influence on the asset returns also differs.

This chapter sheds light on the fruitful future research about incorporating network into asset pricing. My work helps market participants to better understand the network evolution in equity market and how it can affect equity prices. In this work, I use historical return data to construct the time-varying network structure of the equity market without taking some static networks into consideration, such as supply-chain and industry similarity. The importance of these static networks is well documented, and they likely contain useful information in addition to my current framework. Integrating heterogeneous networks to construct a network hierarchy and understanding the aggregated network implications on equity pricing can be a potential area for future research.

CHAPTER 6

ATTENTION BASED DYNAMIC GRAPH LEARNING FRAMEWORK FOR ASSET PRICING

6.1 Introduction

As the machine learning gains great successes in many applications, its application in predicting the price of financial assets has been increasingly interesting to both academic researchers and practitioners. Traditional pricing models mainly focus on firm-specific and market/macro factors, which cannot capture the inter-connection among assets. However, firms are not operating independently and one firm might be affected by the others through multiple networks. As a result, a firm's stock price depends not only on its own characteristics but also on the characteristics of other relevant firms, i.e., the interconnection among firms affects each other's market price. In this paper, we propose a novel two-step graph learning model to capture the dynamic interconnections among firms (connect the "dots") and investigate their contributions to the stock price movement (network-aware prediction).

The idea that the interconnections in networks affect stock prices is intuitive. However, many challenges exist in how to capture the network structure of the equity market as it can be dynamic and complex. Recent efforts in network representation learning and Spatio-temporal modeling show that the network information improves traffic prediction [65, 75, 76, 244] and COVID-19 trend forecasting [79]. These models are largely based on the Graph Convolutional Networks proposed in [72] that focus primarily on a static network with predefined topologies. Unlike social networks or road networks, the equity market's network structure is unknown. Finance studies typically use the pairwise Pearson correlation of firms' historical returns to represent firms' network structure [60, 80, 217]. The Pearson correlation only reveals the linear relationship among entities and might not be sufficient to model the inter-dependency

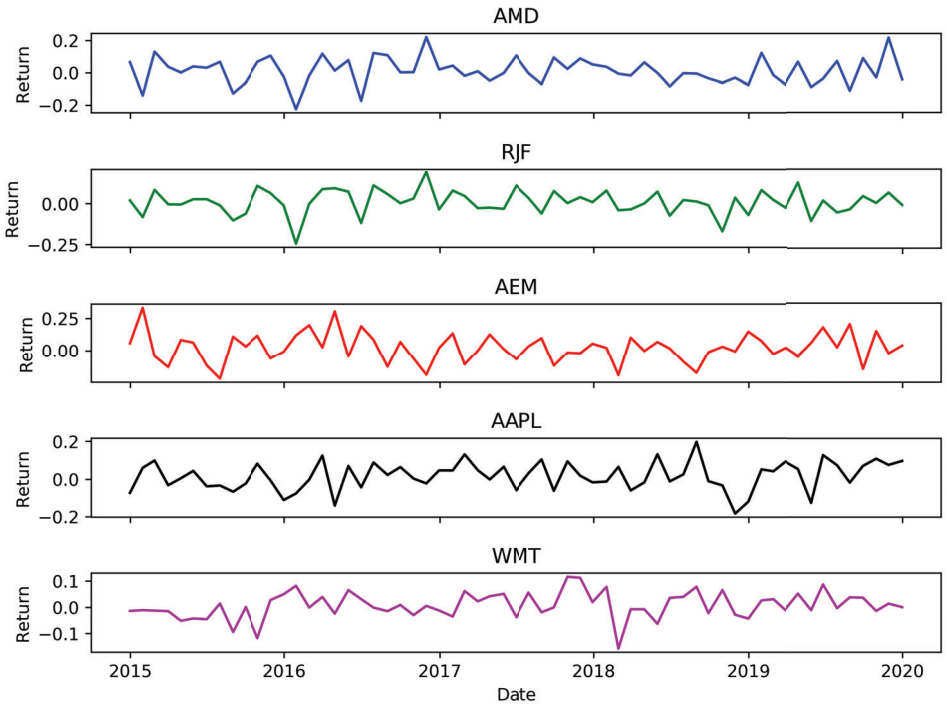


Figure 6.1 Rate of return from five assets over time. The timing of earning returns by AMD (Technology) is similar to that of RJF (Financial Services) and almost opposite to that from AEM (Materials-Mining). AMD and AAPL (Technology) are from the same industry sector, whereas their returns vary significantly. The dynamic nature of the changing relations among firms' return is also visible from the trend lines of AEM and WMT (Consumer Discount Stores). AEM and WMT started with little correlation in the early 2017, began to have strong co-movements between early 2017 and mid-2018, and then diverged into opposite movements from mid-2018 to December 2019.

among stocks. Moreover, it is challenging to integrate the correlation networks in graph structure data as they contain both positive and negative coefficients. Existing approaches typically use absolute values with the assumption that significant correlation represents high similarity regardless of the sign. This assumption violates the core idea that a positive correlation indicates convergence, while a negative correlation indicates divergence. Herskovic [59] use sector similarity as the linkage among firms. However, as Figure 6.1 shows, falling in the same sector or industry group does not necessarily ensure similar returns. Many other latent factors, such as institutional holdings, may affect firms' underlying connections. Moreover, unlike

traditional networks, financial networks are often dynamic. Firms enter or exit the market, and change their business models, capital structures, and supply chains. As a result, relationships among firms continuously evolve (AEM vs. WMT in Figure 6.1). Therefore, directly applying techniques developed for static-known graphs [65, 75, 76, 244] into dynamic equity market network may produce sub-optimal results.

This paper attempts to capture the time-varying networks of the equity market, model non-linear connectivity and dependency relationships, and use them for prediction. The main idea is: in the first step, an attention mechanism is used to learn the dynamic network structure of the US equity market, and in the second step, a recurrent diffusion convolution network is applied to model the spatial and temporal dependency among firms. The first step helps overcome the problem associated with unknown graphs. The second step considers both positive and negative connections and learns the graph embeddings with better predictive power on stock prices. The highlights of this paper are summarized as follows:

- To the best of our knowledge, it is the very first paper to propose the dynamic graph learning framework that tracks and follows the global and local patterns in the equity market over time. Our work enables financial analysis to be network-aware and minimizes the uncertainty associated with the prediction over stand-alone assets.
- We adopt a flexible attention mechanism to learn new networks from scratch or improve upon initial networks. The learned network topologies are non-linear and superior to the commonly used Pearson correlation and can capture the relationships of assets in the complex market environment.
- Unlike previous work, we use graph neural networks to integrate heterogeneous datasets and utilize the fundamentals, historical returns, and the US equity market network structure to improve the asset price prediction.
- The model is superior in prediction accuracy and portfolio performance compared to the conventional asset pricing methods and other off-the-shelf machine learning models.

The rest of the chapter is organized as follows: Section 6.2 details the methodology of the proposed dynamic graph learning model for asset pricing. Section 6.3 describes the experimental settings, data, and analysis results including the

network learning capacity of the model. We provide the results of the ablation study in Section 6.4 and offer the conclusions in Section 6.5.

6.2 Methodology

In this section, we first define the problem of asset pricing prediction and then present the building blocks of the DYnamic Graph learning model for Asset Pricing (DY-GAP).

6.2.1 Problem Definition

The problem related to asset pricing here is defined as how to identify the intrinsic value of assets. One successful investment strategy is to identify undervalued (the current market price of the asset is lower than the intrinsic value) or overvalued (its market price is higher than the intrinsic value) stocks. Then, investors take long positions on (buy) undervalued stocks, or short positions on (sell) overvalued stocks, or both to make profits with the expectation that the market prices will eventually converge to the intrinsic values. The intrinsic values are not observed and need to be estimated by some asset pricing models. Asset pricing models can be grouped into two broader categories based on the input used: (i) time-series models and (ii) factor models.

Historical Return: Time series models are mainly based on historical returns. Researchers find that there exists time-varying pattern of stock returns. Stocks can maintain the performance in the short term named as the momentum effect, i.e., stocks with high (low) returns in the past are likely going to have high (low) returns in the near future. However, in the long term, stock prices could reverse. According to these theories, stock returns of a firm can be estimated with a function of its historical returns.

$$\hat{y}_{t+1} = f_1(y_t, \cdots, y_{t-K}) \tag{6.1}$$

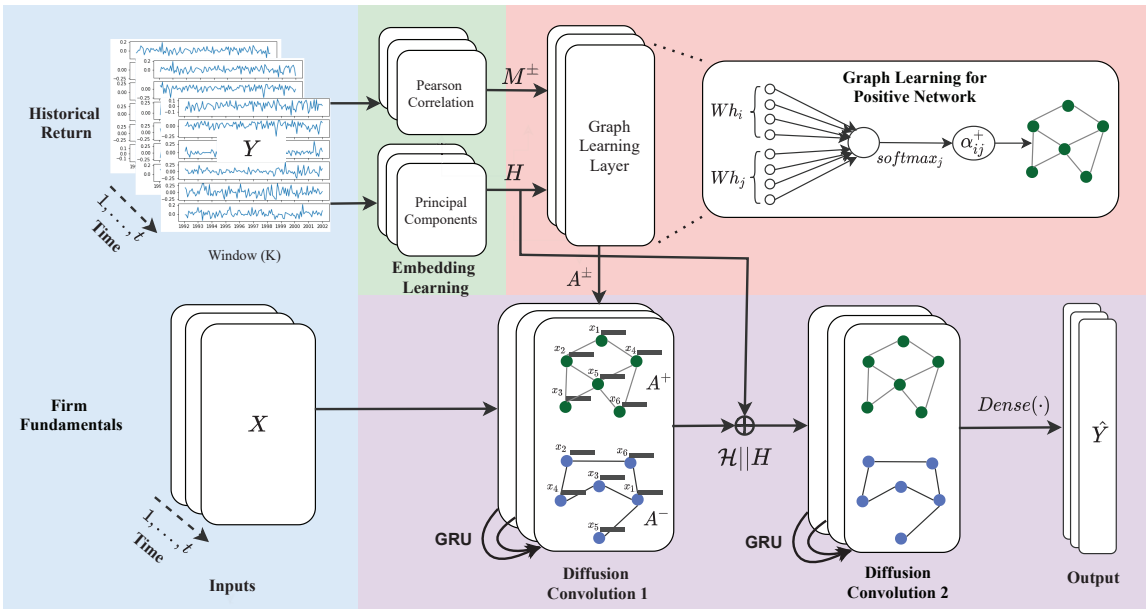


Figure 6.2 The three-stage DY-GAP model architecture. (1) PCA on the historical return learns latent embedding for each firm. (2) Self-attention on the latent embedding learns the network architecture. Two different attention mechanisms are performed to learn both positive (green) and negative (blue) networks. Pearson correlation of historical return ensures masked attention. (3) A diffusion convolution on firms' signals uses the learned network for learning spatial dependency, and GRU recurrent neural network learns temporal dependency. Two diffusion layers are used: the first one with the firm fundamentals and the second one after concatenating latent embedding with the output of first diffusion layer. Solid arrows indicate the flow of information.

here, K is the window size of historical return.

Factor Universe: Factor models are based on the assumptions that asset returns can be expressed as a linear function of a variety of macroeconomic, market, and security-specific factors. These factors include market risk premia, volatility, trading volume, and accounting fundamentals such as firm size, profitability, and liability.

$$\hat{y}_{t+1} = f_2(x_1, \cdots, x_P)$$

Here, P are the total number of factors.

We formulate the asset pricing problem that combines these two types of inputs, the time-series returns and accounting fundamentals, to explain the return variability. First, we define multivariate temporal graphs at time t as $\mathcal{G}_t = (V_t, A_t)$, where V_t is the set of firms (nodes) $|V_t| = N$ and $A_t \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix representing firms' quantitative proximity at time t. The value of A_{ij} indicates the strength of the interdependence. $A_{ij} = 0$ indicates that the firms i and j are independent to each other. The input signal on graph at time t is $X_t = \{x_{ip}\} \in \mathbb{R}^{N \times P}$ and the output is $Y_{t+1} = \{y_i\} \in \mathbb{R}^{N \times 1}$, where N is the number of firms (nodes). Instead of using pre-defined networks, we use neural networks to learn the adjacency matrix A_t at time t from the historical returns $Y_t = \{y_{ik}\} \in \mathbb{R}^{N \times K}$, where K is the rolling window size of historical returns.

Given the observed returns of previous K timestamp $Y_t, \cdots, Y_{t-K}$ and graph signal X_t, the objective of the model is to learn an effective network structure $\mathcal{G}_t$ at each time step and in the meanwhile to predict the next time step $\hat{Y}_{t+1}$ with the integrated model of Graph Neural Networks and Recurrent Neural Networks. We use neural network to implement the forecasting function $f(\cdot)$ with parameters $\Theta = \theta_1, \theta_2, \theta_3$ (θ_1 represents the neuron weights for "connecting the dots", θ_2 contains the parameters for the graph convolutional filter g_{θ_2} defined on $\mathcal{G}_t$, and θ_3 represents the neuron weights for transforming the input node features.):

$$\begin{aligned} \mathcal{G}_t &= Attn(Y_t, \cdots, Y_{t-K}; \theta_1) \\ \hat{Y}_{t+1} &= f(g_{\theta_2} \star X_t; \theta_2, \theta_3) \end{aligned} \tag{6.2}$$

6.2.2 DY-GAP Model Framework

Figure 6.2 presents the architecture of our proposed model. The model consists of three different learning functions: (i) Embedding learning, (ii) Graph learning, and (iii) Spectral and temporal dynamics learning. We follow [245] and apply the

self-attention function to learn dynamic network structures from the historical return data. The return data is often noisy and large. An embedding learning layer is used to clean the noisy data and obtain the firm's condensed representation before performing the attention function. Finally, a spectral-temporal recurrent convolution function is performed on the firms' fundamentals using the learned network.

Embedding Learning Layer Principle Component Analysis (PCA) is widely used in finance for dimension reduction and feature extraction from time series data [246, 247]. At each time period t, we perform Singular Value decomposition (SVD) on historical returns and extract principle components for each firm. For simplicity, we drop the time index for the remaining discussion. Given $Y \in \mathbb{R}^{N \times K}$, we perform SVD on Y as follows:

$$Y = USV^{\top} \tag{6.3}$$

where, U is a unitary matrix and S is the diagonal matrix of singular values corresponding to the Eigenvalues in the correlation matrix and $H = US$ is the principle components. We perform dimension reduction and choose the first L principal components $\sum_{i=1}^{L} U_{:i} S_{:i}^{\top}$ as the embedding matrix of return data, where $L < K$. With N firms, $H = \{h_{il}\} \in \mathbb{R}^{N \times L}$ holds the initial node embeddings (features).

Graph Attention Learning Layer The graph learning layer employs the attention mechanism proposed in [245] to learn the edge coefficient between two nodes. The attention function is performed on each firm's learned embedding H. Following [248], each node in the attention pair is first undertaken a linear transformation with weight matrix $W \in \mathbb{R}^{L' \times L}$ before attended by shared attention mechanism $a : \mathbb{R}^{L'} \times \mathbb{R}^{L'} \to \mathbb{R}$.

The learned attention coefficient is:

$$e_{ij} = a(\mathbf{W}h_{i:}, \mathbf{W}h_{j:}) \tag{6.4}$$

The learned value e_{ij} indicates the importance of firm j's return on that of firm i. We adopt a nonlinear activation function to add nonlinearity to the dependence relation among firms and apply the softmax operation to make the coefficient easily comparable across nodes. The final network edge α_{ij} is defined as follows:

$$\alpha_{ij} = \frac{\exp(\sigma(\mathrm{a}^\top[\mathbf{W}h_{i:}||\mathbf{W}h_{j:}]))}{\sum_{j=1}^{n}\exp(\sigma(\mathrm{a}^\top[\mathbf{W}h_{i:}||\mathbf{W}h_{j:}]))} \tag{6.5}$$

Here $\top$, $||$, and σ represent transpose, concatenation and nonlinear activation operation, respectively. $\mathrm{a} \in \mathbb{R}^{2L'}$ is a weight vector parametrizing the attention mechanism $a(\mathbf{W}h_{i:}, \mathbf{W}h_{j:})$. The softmax operation satisfies $\sum_{j=1}^{n}\alpha_{ij} = 1$ and $\alpha_{ij} >= 0$, and thereby, provides normalization across the learned network.

Studies suggest that the desired objective can be achieved with a simple attention strategy attending to all node pairs while ignoring the structural information [248]. However, as discussed in Section 6.1, the interactions among firms are not uniform and rather complex: some of them are converging and render the market towards the same direction, while the others are diverging and lead to the heterogeneous behaviors in the market. Later, we show in Section 6.3.6 that the distribution of the interaction relationships aligns with the market conditions.

A firm is not necessarily connected to all other firms, and a spurious relationship might do more harm than no connection due to the overfitting problem and excessive computation costs. Therefore, to expedite the graph learning process, we use the masked attention function where only the closely relevant firms of the target firm

are attended. We use the Pearson correlation coefficients of the historical return data to determine a firm's initial connection and potential neighbors. The Pearson's correlation, ρ_{ij}, between firm i and j at time t with a rolling window K is defined as follows:

$$\rho_{ij} = \frac{Cov(r_i, r_j)}{\sqrt{Var(r_i)Var(r_j)}} \tag{6.6}$$

We keep the rolling window size to be the same as the embedding learning framework window for both daily and monthly data. The original return correlation matrix is dense, with almost all firms' returns (positively or negatively) correlated to others. The values close to zero do not provide useful information about firms' similarity but noise. To enhance the signal-to-noise ratio (SNR), we adopt the noise filtering technique proposed in [60]. The signal enhancement algorithm first denoises the empirical Pearson covariance matrix by performing an eigendecomposition and replacing the noisy eigenvalues with their average to preserve the trace of the correlation matrix. The denoised matrix might not be a positive definite symmetrical matrix. Finally, the signal enhancement algorithm applies convex optimization to construct a correlation matrix that is the closest to the denoised matrix.

As a result, the correlation matrix has positive and negative coefficients. A positive correlation between i and j indicates that i and j reassemble to each other, while a negative one means the opposite. A single attention head treats both positive and negative coefficients indiscriminately and violates the correlation property. Therefore, to learn meaningful network representations, we use two separate attention heads, each attending the positive and negative components of the firm correlation matrix separately. We first decompose the new correlation matrix into two mask matrices: M^+, where $M^+_{i,j} = 1$ if $\rho_{i,j} \geq 0$, otherwise 0; and M^- where

$M^-_{i,j} = 1$ if $\rho_{i,j} < 0$, otherwise 0. Then we perform the following masked attentions according to the positive and negative matrices:

$$\alpha^{\pm}_{ij} = \frac{M^{\pm}_{ij}\exp(\sigma(\mathrm{a}^\top[\mathbf{W}h_{i:}||\mathbf{W}h_{j:}]))}{\sum_{j=1}^{n} M^{\pm}_{ij}\exp(\sigma(\mathrm{a}^\top[\mathbf{W}h_{i:}||\mathbf{W}h_{j:}]))} \tag{6.7}$$

The learned attention coefficients α_{ij} are assembled into two new affinity matrices $A^+ = \{\alpha^+_{ij}\} \in \mathbb{R}^{N\times N}$ and $A^- = \{\alpha^-_{ij}\} \in \mathbb{R}^{N\times N}$.

Recurrent Diffusion Convolution Layer We use the recurrent diffusion function to model the spatial relationship (network) and temporal dependency among firm fundamentals based on the learned network structure. We follow the diffusion process in [75] to propagate the firm information to their neighbors.

Graph Diffusion We apply two separate diffusions in two networks $A^{\pm}$ learned from the attention function. The diffusion process consists of a random walk on Graph $\mathcal{G}$ with the state transition matrix (A^+ or A^-) and its random walk normalization $D^{\pm^{-1}}$ where $D^{\pm}$ is the diagonal matrix of the node degree, $D^{\pm}_{ij} = deg(v_i)$ if $i = j$, otherwise 0.[1] Following [75], we define the diffusion convolution on graph filter f_θ and each input channel $X_{:,p} \in \mathbb{R}^N$ $(1 \le p \le P)$ of the graph signal consisting of firm fundamentals as follows:

$$f_\theta \star_{\mathcal{G}} X_{:,p} = \sum_{s=0}^{S-1} \theta_{s,1}(D^{+^{-1}}A^+)^s X_{:,p} \| \sum_{s=0}^{S-1} \theta_{s,2}(D^{-^{-1}}A^-)^s X_{:,p} \tag{6.8}$$

[1]In our case, all row sums in $A^{\pm}$ equal to 1, and degree matrices $D^{\pm}$ become an identity matrix. We still keep $D^{\pm}$ in the subsequent convolutional diffusion equations and make them generalize to any other un-normalized affinity matrix $A^{\pm}$.

where f_θ denotes the graph diffusion filters, $\|$ concatenates the positive and negative diffusions of X, and $\theta \in \mathbb{R}^{S\times 2}$ are the parameters for filter. $D^{+^{-1}}A^+$, $D^{-^{-1}}A^-$ are the (random-walk normalized) transition matrices of the diffusion process learned from the positive and negative correlations, respectively. $(D^{\pm^{-1}}A^\pm)^s X$ is a s-step matrix power iteration on input X (s-step diffusion). The weighted sum parameterized by θ represents the learned features from the up-to-S-hop neighborhood via diffusion. S denotes the maximum number of diffusion steps. In [75], Li et al. show that a sufficiently large number of diffusion will converge to a stationary distribution. The S-step diffusion network becomes a spectral graph convolutional neural network once we replace the attention matrices $A^\pm$ with symmetrical matrices $\hat{A^\pm} = \frac{A^\pm + A^{\pm\top}}{2}$ and then perform symmetrical normalization. Equation (6.8), with these necessary modifications, can be transformed into a generalized form of the polynomial filter defined on graph filter matrix with the learnable parameters θ (Equation 3 in [73]).

It is important to note that, unlike the work in [75], our model is based on the undirected graphs with the positive and negative relationships, performs diffusion on these two different networks, and concatenates two outputs. Instead of transposing the original network, the second term in Equation (6.8) employs the negative network. The two-term diffusion allows us to learn the spatial dependencies from the positively connected neighborhood and negatively connected neighborhood. The diffusion convolution layer takes $X \in \mathbb{R}^{N\times P}$ as the input and generates the tensor output $\mathcal{H} \in \mathbb{R}^{N\times Q\times 2}$, where P is the number of input channels and Q is the number of output channels. For each output channel $q \in 1, \ldots, Q$, the output contains two components: one from the positive diffusion and the other one from the negative diffusion, and is in the form of:

$$\mathcal{H}_{:,q} = \sigma\left(\sum_{p=1}^{P} f_{\Theta_{p,q,:,:}} \star_{\mathcal{G}} X_{:,p}\right) \tag{6.9}$$

here, $\Theta_{p,q,:,:} \in \mathbb{R}^{S\times 2}$. $\{f_{\Theta_{p,q,:,:}}\}$ are the diffusion filters and σ is the activation function, i.e., ReLU.

GRU for Temporal Modeling Asset pricing models suggest that past returns (prices) of a stock have the predictive power for its future returns (prices). It is essential to model the temporal dependency along with the spatial dependence. Gated Recurrent Units (GRU) [67] is proved to be effective for temporal dependency modeling. As suggested in [75], we replace the common linear transformation matrices for all GRU gates with the graph diffusion and augment the diffusion convolution with GRU as follows:

$$\begin{aligned} r^t &= \sigma(\Theta_r \star_{\mathcal{G}} [X^t, \mathcal{H}^{t-1}] + b_r) \\ u^t &= \sigma(\Theta_u \star_{\mathcal{G}} [X^t, \mathcal{H}^{t-1}] + b_u) \\ C^t &= \sigma(\Theta_c \star_{\mathcal{G}} [X^t, (r^t \odot \mathcal{H}^{t-1})] + b_u) \\ \mathcal{H}^t &= u^t \odot \mathcal{H}^{t-1} + (1 - u^t) \odot C^t \end{aligned} \tag{6.10}$$

where $X^t, r^t, u^t, \mathcal{H}^t$, denote the input, reset, update gate and the hidden state at time t, respectively. $\star_{\mathcal{G}}$ denotes the graph convolution diffusion and $\Theta_{r,u,c}$ contains the parameters of the respective filters for each gate in GRU.

The embedding H serves two roles in the proposed framework in Figure 6.2: learning attention coefficients and supplying the stock market information of each firm. To support the second role, we design two recurrent diffusion convolution layers to process the firm signals in Figure 6.2. The first layer performs the diffusion convolution on the firm fundamentals and the hidden state $[X^t, \mathcal{H}^{t-1}]$. The second layer concatenates the output $\mathcal{H}^t$ from the first diffusion layer and the embedding H from the embedding learning module, and applies another recurrent diffusion on $\mathcal{H}^t||H^t$. This concatenation function ensures that the return data is directly incorporated into the recurrent diffusion convolution framework. Consequently, we

avoid the vanishing gradient problem and significantly improve the learning process. The recurrent diffusion using GRU defined in Equation (6.10) is applied in these two layers, as shows in Figure 6.2.

Finally, for prediction, the diffusion network simply uses one layer MLP to regress the prediction output $\hat{Y}_{t+1} \in \mathbb{R}^{N \times P}$ on the hidden state of the second recurrent diffusion convolution layer.

6.3 Experimental Details

6.3.1 Data

For the experiment, we collect two sets of data with different frequencies – monthly and daily. The monthly data include 1098 stocks from Russell 3000 Index. The sample period is from January 01, 1990, to December 31, 2019, and divided into three folds: training (January 1990 to December 2009), validation (January 2010 to December 2013), and test (January 2014 to December 2019). The daily data involves all the stocks in the S&P-500 index. The sample period is from January 01, 2010, to December 30, 2020, and divided into training, validation, and test sub-sample as January 01-2010 to December 31-2015, January 01-2016 to December 31-2017, and January 01-2018 to December 31-2020, respectively. For both monthly and daily data, selected firms are based on the index constituents by October 31, 2020. Few inconsistent firms are removed from the sample. The monthly and daily stock returns are from CRSP, and firms' fundamental variables are from Compustat. These two datasets are available at https://wrds-www.wharton.upenn.edu/. Based on the data availability, fundamental variables included in the daily and monthly analysis are slightly different. Following the asset pricing literature, we calculate 21 monthly features and 24 daily features from firms' fundamentals. The detailed calculation procedure is in Table 6.1.

Table 6.1 Fundamentals Variables

Firm Characteristics	Calculation Procedure
Total assets to market	Total asset / market value of equity
Size	log $(p_t \times share\ outstanding)$
Turnover	Volume / share outstanding
Growth rate of volume	$V_t - V_{t-1}/V_{t-1}$
Growth rate of share outstanding	$SO_t - SO_{t-1}/SO_{t-1}$
Closeness to past year high	$P_{t-1} - max(P_{t-1}, ...p_{t-12})/max(p_{t-1}, ..., p_{t-12})$
Closeness to past year low	$P_{t-1} - min(P_{t-1}, ...p_{t-12})/min(p_{t-1}, ..., p_{t-12})$
Spread	$p^h_{t-1} - p^l_{t-1}$, monthly [daily] high minus low price
Opening and closing spread	$(p^o_{t-1} - p^c_{t-1})$, daily opening minus closing price
Capital gain	Value is 0, if no capital gain is recorded
EPS	Earning per share
Dividend	Dividend paid in cash
Total volatility	Price volatility of last 60 months [last 252 days].
Idiosyncratic volatility	Total volatility - market volatility
Market return	Return on S&P-500 index
CAPM market Beta	Beta on Fama-French market factor
Small minus big beta	Beta on Fama-French size factor
High minus low beta	Beta on Fama-French value factor
1 week momentum	$(p_{t-1} - p_{t-5})/p_{t-5}$ -daily only
2-week momentum	$(p_{t-1} - p_{t-10})/p_{t-10}$ -daily only
1-month momentum	$(p_{t-1} - p_{t-2})/p_{t-2}$ $[(p_{t-1} - p_{t-21})/p_{t-21}]$
2-month momentum	$(p_{t-1} - p_{t-42})/p_{t-42}$ -daily only
3 month momentum	$(p_{t-1} - p_{t-3})/p_{t-3}$ $[(p_{t-1} - p_{t-63})/p_{t-63}]$
6 month momentum	$(p_{t-1} - p_{t-6})/p_{t-6}$ $[(p_{t-1} - p_{t-126})/p_{t-126}]$
12-month momentum	$(p_{t-1} - p_{t-12})/p_{t-12}$ -monthly only

6.3.2 DY-GAP Setting

We implement all machine learning-based models in Python using Pytorch [249] and/or TensorFlow [151]. For monthly data, the window size K for historical returns for embedding layer and for Pearson correlation is 36 months; for daily data, it is 122 days (6 months). In the embedding layer, we use embedding size $L = 10$. For attention, we use eight attention heads, and for diffusion convolution, we use ten

diffusion steps. Other hyperparameters for the attention and diffusion convolution layer are selected based on the validation result. We use early stopping criteria for model training and stop training if validation loss does not decrease in 10 Epoch.

6.3.3 Baselines

We first compare our model with a series of multi-factor pricing models and time-series methods. For multi-factor pricing models, we consider the well-acknowledged Fama-French five-factor model [49], multivariate regression using all fundamentals [250] and recently proposed empirical asset pricing via machine learning (EAP-ML) [47] model. For time-series models, we use the classic ARIMA and several advanced deep neural network-based approaches, including fully connected long short-term memory (FC-LSTM) [251] and the state-of-the-art neural basis expansion analysis for interpretable time series forecasting (N-BEATS) [252]. Finally, as our approach is inspired and combines two models: graph attention network (GAT) [248] and diffusion convolution recurrent neural network (DCRNN) [75], we compare the performance of our model with them in order to ensure the advantage of integration. For most models, we use authors' source codes, if available, with necessary modifications. The best hyper-parameters are chosen based on the validation dataset. For MR, FF-5, and ARIMA, coefficients are determined with the training and validation data, and then the learned coefficients are used to estimate the performance values for test data.

6.3.4 Forecasting Future Returns

The model performance is first evaluated in terms of the prediction accuracy of future returns. There are three matrices to assess the prediction performance, including RMSE, MAE, and MAPE. The small values represent small prediction errors and high accuracy. Table 6.2 presents the prediction performance of each model. The left panel represents results from the monthly data (Russel-3000 Index), and the right panel represents results from the daily data (S&P-500 Index). Reported results are

Table 6.2 Forecasting Results

	Monthly Data			Daily Data		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
MR	0.1211	0.0944	66.4063	0.0310	0.0251	24.5875
FF-5	0.0941	0.0843	58.9432	0.0274	0.0209	20.8055
ARIMA	0.1141	0.0852	62.8968	0.0351	0.0252	32.6051
EAP-MLP	0.0969±.009	0.0726±.007	61.5435±7.3	0.0519±.018	0.0524±.017	42.0046±9.1
FC-LSTM	0.0949±.006	0.0723±.006	56.1883±4.2	0.0254±.003	0.0183±.001	19.6615±0.9
N-BEATS	0.1065±.001	0.0739±.001	61.4771±2.3	0.0241±.008	0.0173±.002	17.3975±1.1
DCRNN	0.0912±.002	0.0727±.001	54.8688±1.2	**0.0230**±.005	0.0178±.002	16.8916±1.0
GAT	0.0953±.006	0.0778±.005	61.7650±5.5	0.0292±.006	0.0195±.005	17.4611±1.6
DY-GAP	**0.0853**±.002	**0.0632**±.001	**52.2400**±1.3	0.0233±.003	**0.0161**±.001	**15.6098**±0.7

Note: This Table reports the return prediction results from our proposed model (DY-GAP) and other benchmarks. MR is multivariate regression with fundamentals, FF-5 is Fama-French five-factor model, ARIMA is time series model on return, EAP-MLP is the machine learning-based asset pricing model, FC-LSTM and N-BEATS are deep learning-based recurrent neural networks for time series prediction, and DCRNN and GAT are two graph convolutional neural networks. A lower value of RMSE, MAE and MAPE indicates better performance.

Table 6.3 Portfolio Performance

	Monthly Data			Daily Data		
	Average (%)	STD (%)	Sharpe Ratio	Average (%)	STD (%)	Sharpe Ratio
MR	1.0046	4.0400	0.8614	0.0120	1.7669	0.1081
FF-5	1.0980	4.4340	0.8578	0.0116	1.7450	0.1055
EAP-MLP	1.1455	5.1880	0.7649	0.0733	2.2850	0.5092
ARIMA	1.1840	4.9540	0.8279	0.0401	1.6876	0.3772
LSTM	1.1790	4.8877	0.8356	0.0322	2.0711	0.2468
N-BEATS	1.1046	4.0400	0.9471	0.0424	1.8450	0.3648
DCRNN	0.9121	5.1650	0.6117	0.0218	1.6104	0.2149
GAT	1.0820	4.7730	0.7853	0.0220	1.7669	0.1980
S&P-500	0.8236	**3.2534**	0.8769	0.0449	**1.4892**	0.4785
DY-GAP	**1.5500**	4.5617	**1.1771**	**0.0854**	1.9500	**0.6955**

Note: This Table reports the performance of long portfolios created based on the prediction of our proposed model (DY-GAP) and other benchmarks. MR is multivariate regression with fundamentals, FF-5 is Fama-French five-factor model, ARIMA is time series model on return, EAP-MLP is the machine learning-based asset pricing model, FC-LSTM and N-BEATS are deep learning-based recurrent neural networks for time series prediction, and DCRNN and GAT are two graph convolutional neural networks.

from the test data sets only. The stochastic nature of machine learning models may lead to different forecasts with different initialization. Therefore, we run each model ten times with different random seeds and report the average performance and the standard deviation.

Our proposed model DY-GAP outperforms other models in all three performance metrics for forecasting with monthly data. In RMSE, DY-GAP outperforms FF-5 and EAP-MLP by 9% and 12%, respectively. The two best performing models among all machine learning-based baseline models are FC-LSTM and DCRNN. Nevertheless, DY-GAP still outperforms these two models by 10%, and 6% in RMSE and 7% and 5% in MAPE, respectively.

The prediction performance of our model (DY-GAP) is consistently better using daily data with only one exception where DCRNN is the best model in terms of RMSE. However, in terms of MAE and MAPE, our model is still the best. In the later section, we show that DCRNN has a high risk that lowers its Sharpe Ratio. The prediction error of DY-GAP in terms of MAPE is smaller than DCRNN by 7%, GAT by 11%, N-BEAT by 10%, and FF-5 by 25%. It is worth noting that the superiority of sophisticated predictive models is also visible in the result of daily data. As the data frequency and size increase, sophisticated models can take advantage of large datasets to minimize their overfitting problem and use their increased learning capacity to reduce bias. As a result, the state-of-the-art N-BEATS outperforms the earlier deep learning model (FC-LSTM) by 5% in terms of RMSE with daily data and becomes the third-best model.

The integration of two powerful methods and nonlinearity explain the superior performance of DY-GAP. The use of both attention and diffusion function enables our model to harness the advantage of both GAT and DCRNN and outperforms the application of these two models individually. DY-GAP has clear superiority over time series models, such as ARIMA, FC-LSTM, and N-BEATS, as DY-GAP

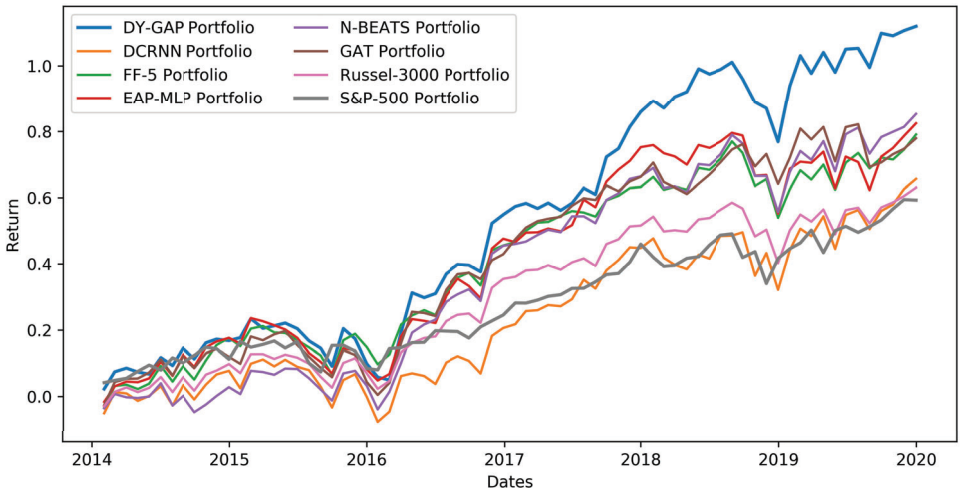


Figure 6.3 The portfolio performance from the monthly data. At the beginning of each month based on each model prediction, we hold the top 10% stocks and at the end of the month, we liquidate the stocks. The thick blue line is the cumulative return of our model in the test period and thick gray line is the S&P-500 index return during the test period. Russel-3000 is the average return of all studied firms.

considers both time series and factor information. In addition, these time series models are applied for individual firms and therefore, information from the market or related firms is ignored in these models. Compared with multi-factor pricing models, such as FF-5 and EAP-MLP, our model considers the spatial connectedness and nonlinear interaction among asset returns and attains a great performance improvement compared to these traditional models.

6.3.5 Portfolio Performance

In this section, we show the economic benefits of our model via the portfolio analysis. For the monthly data, we take a long position on (i.e., buy) the top 10% stocks with the highest predicted returns at the beginning of each month, hold the position until the end of the month, and then liquidate (i.e., sell) all stocks. Figure 6.3 represents the cumulative returns of the portfolios by different models from January 2014 to December 2019. The thick blue line represents the cumulative return of our model in the test period, and the thick gray line is for the S&P-500 index return during the

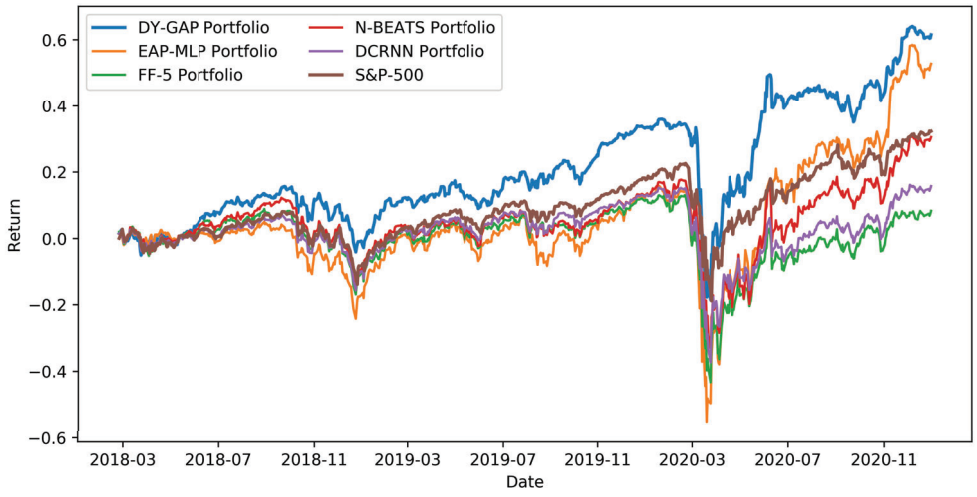


Figure 6.4 The portfolio performance from the daily data. For each trading day, we take the long position at (i.e., buy) the top 10% of highest predicted stocks. The thick blue line is the cumulative return of our model in the test period.

test period. Although in the early years, the performance of the DY-GAP portfolio is indistinguishable from others, our model stands out over time. The cumulative return of our model doubles both the S&P-500 index return and the weighted average return of all stocks in the Russel-3000 index. The finding suggests a profitable investment strategy by identifying the "success" group (i.e., the top 10% stocks with the highest returns) with our model. Among all other baseline models, EAP-MLP and N-BEATS also perform well in the cumulative return.

Table 6.3 reports the monthly average rate of return, monthly standard deviation, and annualized Sharpe ratio of different portfolios. Sharpe ratio is an essential measure of portfolio performance because it provides the return by the unit of risk and exposes which portfolio offers the best returns (rewards) by taking the same amount of risk. The higher the Sharpe ratio indicates the better predictability of a model. Following the formula from Morningstar, Inc, we calculate the annualized (yearly) Sharpe ratio from monthly returns as $SR = \frac{\sum_{m=1}^{M} r_m}{\sigma_{r_m}} \times \sqrt{12}$. Our proposed model earns a high return rate (1.5% monthly) with relatively low risk (3%). Although DCRNN has high predictive power in Section 6.3.4, the portfolio analysis shows that

the risk associated with DCRNN is much higher. As a result, the Sharpe ratio of DCRNN is lower than our model. S&P-500 index portfolio has the lowest risk, but it generates a low rate of return. The Sharpe ratio of the S&P-500 index portfolio is similar to that of MR, FF-5, ARIMA, and LSTM. The Sharpe Ratio of our model, DY-GAP, is the highest with 1.18 and significantly higher than that of the second-best model, N-BEATS, with 0.95.

The performance decline of DCRNN and the improvement in N-BAEATS in portfolio evaluation attribute to their model formulation. During training, the models similar to DCRNN learn global parameters by minimizing errors over all firms, whereas N-BEATS learns individualized parameters for each firm. In portfolio analysis, only the selected (top 10%) firms are included. Other firms may have a better prediction, but do not fall in the leading 10% group, and are excluded from the portfolio. However, the evaluation results suggest that our model is less susceptible to this issue, has a better prediction accuracy, and provides a good portfolio performance.

We also perform a portfolio analysis on the S&P-500 stocks using daily data. Figure 6.4 shows the cumulative return from the portfolio constructed based on the daily prediction. Our model, DY-GAP, maintains superior performance over all other models throughout the test period. Our cumulative return is more than doubled compared with the S&P-500 index return. There is an interesting pattern that the up-down trend in the portfolio performance from all models echos the market return. All models, including the DY-GAP, are affected by the market's downside, particularly for unexpected events like the COVID-19 pandemic. However, the incorporation of network and spatial dependency allows DY-GAP to select the stock groups that generate relatively higher returns in the middle of the pandemic. EAP-MLP is the second-best, especially in the latter part of 2020. This finding suggests that our model is able to identify stocks (firms) with faster recovery during recession periods.

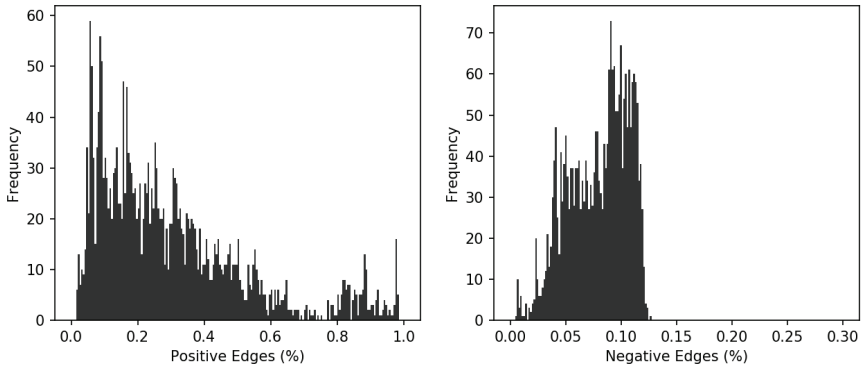


Figure 6.5 Histogram of the learned edges from S&P-500 daily data.

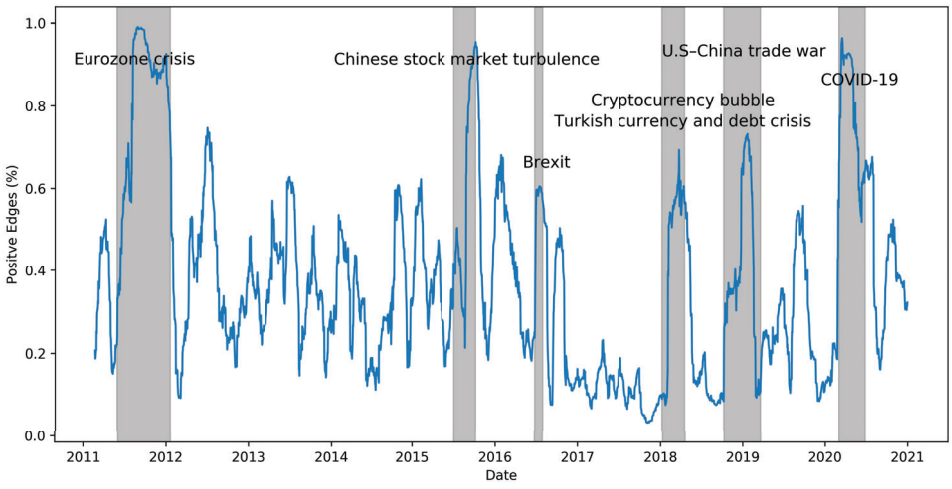
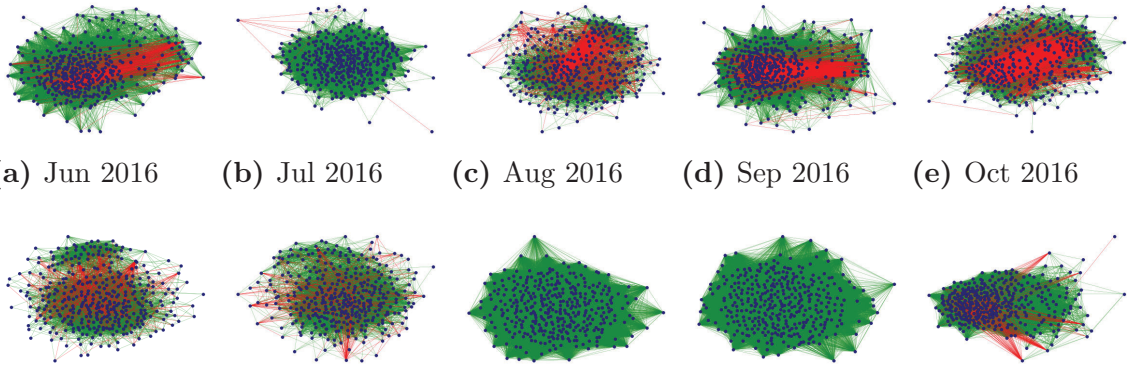


Figure 6.6 The learned positive edges from S&P-500 daily data. There are visible spikes in the degree of positive edges during significant financial, political, and economic events.

6.3.6 Graph Learning

This section evaluates the network learning capacity of our proposed model. The dynamism of the US equity market is evident in the learned networks from the S&P-500 stocks daily data. Figure 6.5 shows the distribution of positive and negative edges during the studied period. Most of the time, the percentage of negative edges is 5%-12%, the positive edges are 15%-30% among all possible connections. The remaining percentages are the insignificant edges that are eliminated via the graph



(a) Jun 2016 **(b)** Jul 2016 **(c)** Aug 2016 **(d)** Sep 2016 **(e)** Oct 2016

(f) Jan 2020 **(g)** Feb 2020 **(h)** Mar 2020 **(i)** Apr 2020 **(j)** May 2020

Figure 6.7 The network structure of S&P-500 stocks at different point of time. Figure 8a-8e are the market network before, during, and after the Brexit, and Figure 8f-8j are for the Covid pandemic. Green represents positive edges and red represents negative edges.

learning process. In the equity market, many firms are positively associated, which is also the case for large firms from the S&P-500 index. However, at some time points, the percentage of positive edges reaches $> 90\%$. In-depth analysis shows that those periods with extremely high positive edges are associated with economic or political events. Figure 6.6 shows that, during the Eurozone crisis, Chinese stock market turbulence, Brexit, and Covid-19, the percentage of positive edges significantly increases. A similar finding is documented by [233]. During a normal period with a stable macro environment, the performance of a firm might be driven more by its own financing, investment, and operation decision, rather than the global market. It is natural to see both positive and negative linkages among firms. By contrast, when there is a major event, most firms respond accordingly in the same direction, i.e., there is a similar upward or downward trend in stock prices, and therefore, positive linkages increase. This can also be associated with contagion effects and systematic risks during recessions and crises.

Besides, the impacts of major events are not identical. We use two events as the example, one is the Brexit in 2016, and another is the Covid pandemic in 2020, and

show how the market network changes before and after those two events in Figure 6.7. Figures 6.7a-6.7e represent the market structure before, during, and after Brexit from June 2016 to October 2016. Figures 6.7f-6.7j are for the Covid-19 pandemic impact on the market network structure from January 2020 to May 2020. During the Brexit, the network structure changed (sub-Figure 6.7b), but all the negative edges did not vanish overnight. The market also recovered quickly and went back to its normal state. However, during the outbreak of Covid-19 pandemic, the almost all edges turned to be positive (sub-Figures 6.7h and 6.7i). At the beginning of May, the market coped with the pandemic's initial setback and started to recover to a certain extend. In sub-Figure 6.7j, it is also noticeable that instead of scattered negative edges through the market, there exist a handful of companies that had negative edges. Further investigation reveals the good performance of this handful of companies, such as Amazon and Walmart, during the peak months of the Covid-19 Pandemic.

6.4 Ablation Study

To gain a better understanding of the contributions of individual components of our model, we perform an ablation study with five alternative versions of our model: (i) the proposed model with a signed network structure, attention mechanism, and diffusion convolution (DY-GAP), (ii) without any network structure (DY-GAT), (iii) considering both positive and negative relationships in a single network (DY-GAP-Unsigned), (iv) replacing diffusion convolution with simple graph convolution (GCN) as suggested in [72] (DY-GCN), and (v) with only a single diffusion layer (DY-GAP-Single Diffusion).

Table 6.4 reports the prediction accuracy from all other alternative models. DY-GAT ignores the constraints on network connectivity and allows all firms to attend to all other firms. The change to the model leads to the performance reduction by 31% (monthly) and 26% (daily) compared to the model abiding by the restrictions.

Table 6.4 Prediction Results from Ablation Study

	Monthly Data		Daily Data	
	RMSE	MAPE (%)	RMSE	MAPE (%)
DY-GAP	0.0853	52.2400	0.0233	15.6098
DY-GAT	0.1121	61.7160	0.0294	19.2406
DY-GAP-Unsigned	0.1072	58.4270	0.0261	17.0153
DY-GCN	0.0936	53.1476	0.0270	20.5101
DY-GAP-Single Diffusion	0.0908	52.6881	0.0240	16.9219

Note: This table reports the return prediction results from alternative versions of our model. DY-GAP is our proposed model, DY-GAT is a model without any network structure, DY-GAP-Unsigned is a model with unsigned networks using absolute values, DY-GCN is a model with simple graph convolution instead of diffusion convolution, and DY-GAP-Single Diffusion is a model with only a single diffusion layer. A lower value of RMSE and MAPE indicates better performance.

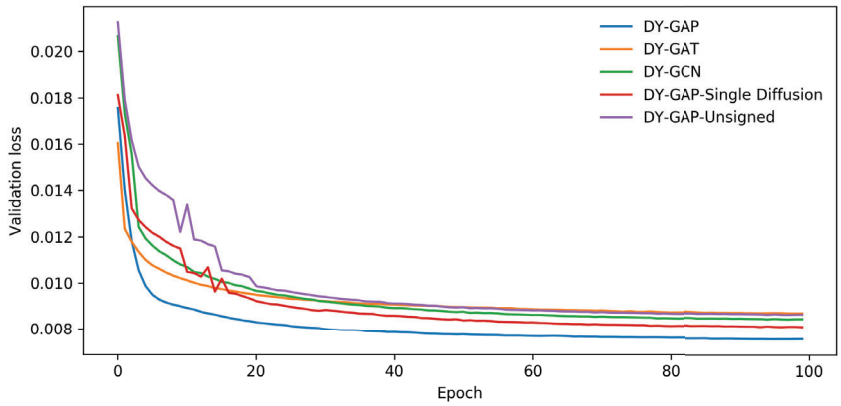


Figure 6.8 The loss curves of our proposed model and some other alternative of network and convolution operation. DY-GAP with both positive and negative network and diffusion convolution achieves the lowest validation error.

In DY-GAP-Unsigned, we provide a binary network structure for masked attention by ignoring the sign. This version does not differentiate between positive and negative connections and provides the same attention to both. As the result of modeling contradictory information identically, the model achieves a sub-optimal performance. In DY-GCN, instead of the diffusion convolutional recurrent neural network, we use the simple graph convolution neural network. Simple convolution only models spatial

dependency among firms, ignoring any temporal dependency. Earlier studies suggest that the historical average excess return is an important predictor for future excess return. Therefore, ignoring the critical temporal relation will result in sub-optimal model performance. The performance drop in DY-GCN is 10% in monthly data and 16% in daily data. In DY-GAP-Single Diffusion, we use only one diffusion layer and predict $\hat{Y}$ from initial $\mathcal{H}$, skipping the second layer of concatenating the return embedding H. In this model, the performance decreases by 6% for monthly data and 3% for daily data. The rationale for this observation is that the concatenation of return embedding allows the model to incorporate historical information directly into the learning process. Although we learn initial graphs from the return embedding, the graph learning process loses some information through masking operation and vanishing gradient. The concatenation function and second diffusion layers help us recover the lost information and thereby improve model performance.

Figure 6.8 shows the validation loss curves among all ablation models from the monthly data. The validation error of all the models converges in approximately 40 epochs. DY-GAP has the lowest validation error among all alternative versions. Both unsigned (DY-GAP-Unsigned) and the single diffusion model (DY-GAP-Single Diffusion) suffer sporadic gradient descents initially but eventually converge. The plausible reason for this slow convergence is that these two models access less information than others. In the unsigned graph, the machine learning model considers both positive and negative edges the same; therefore, learning meaningful representation becomes difficult. For single diffusion, we are only learning spatial-temporal dependency on the features. The diffusion convolution layer has no direct access to the historical return data; as a result, the learning process is slower.

6.5 Conclusion

In this paper, we propose a graph neural network-based approach to asset pricing. This work offers a novel solution to two critical problems in asset pricing: the interrelation among the firms and the evolution of the interrelation. Our model outperforms many traditional asset pricing models and advanced machine learning models in terms of prediction accuracy and portfolio performance. We also conduct analyses on several alternative models in the ablation study and show that DY-GAP with the positive and negative correlated networks and diffusion convolution layer performs the best. This paper confirms that the firm interconnection is bidirectional and relevant to the market analysis. The positive and negative relations must be treated differently because they serve different roles in the market. The incorporation of an effective network representation and the model for spatial and temporal relations enhances the stock price prediction and more profoundly, improves our understanding of the network structure in the financial market. Our model is still sensitive to some market conditions during abnormal events, e.g., COVID-19, and follows market downturns. The future research direction includes improving the model under various market conditions and design mitigation strategies.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH DIRECTION

The success of machine learning algorithms largely depends on proper data representation. In this dissertation, I show how machine learning techniques developed by computer scientists for their specific problems are extended to solve financial problems. I show that representation learning improves missing value imputation and solves an aged problem in finance – analysts' earnings forecast and associated missing value. Experiments suggest that representing financial panel data as a third-order tensor and imputing missing values with advanced machine learning techniques increase the accuracy of firms' earnings prediction. I also show the importance of using domain knowledge for learning proper representation. Financial data comes from multiple sources and in a variety of forms. Properly integrating heterogeneous data from all these sources improve the quality of latent embedding significantly.

With the help of high data products and reliable representation, I explored the capacity of machine learning and network modeling technique in developing asset pricing models. Using a simple representation learning framework, "autoencoder", I proposed a latent asset pricing model that attempts to explain the return difference between assets. Furthermore, I designed an advanced graph neural network-based model to learn the U.S. equity network. Representation learning on the network helps us understand the equity market network over time and learn the associated network embedding. The learned embedding facilitates to building both global and firm-specific asset pricing models. More importantly, the comprehensive analysis of these models reveals the importance of network interaction in asset pricing. The network factor contains a significant risk premium for firms' network risk exposure. Incorporating network information from advanced representations learning framework

reduces model mispricing. The result also suggests that network impact is not the same across firms and over time. Different events affect the market network differently, and as a result, their influence on the asset returns also differs.

The prospective future extension of the research in this dissertation has multi-fold. First, from the representation learning perspective, it might be interesting to examine whether different variables will generate different results and some of these variables are more important than others for missing value imputation. Second, this dissertation shows that combining multiple heterogeneous data sets improve the quality of data imputation and the accuracy of return prediction. It would be interesting to see whether the same techniques also work for other areas of finance, including bond price estimation. Third, in the network learning framework, I use historical return data to construct the time-varying network structure of the equity market. This dissertation did not consider static networks information, such as supply-chain relationships and industry affinity. The importance of these static networks that do not change over a short period is well documented, and they likely contain valuable information in addition to my current framework. Integrating heterogeneous networks in constructing a network hierarchy and understanding the aggregated network implications on equity pricing can be a potential area for future research. Finally, the general framework for the signed Laplacian spectrum has potential for improvement and extension, for example, performing graph cuts, identifying proper equity market clusters, and analyzing other financial markets' network structures where negative connections exist.

APPENDIX A

RESULT SUPPLEMENT AND BENCHMARK METHODS

Appendix A provide the supplemental details from Chapter 2. First, section A.1 discusses all the benchmark methods of missing value imputation that I used to compare the performance of MF and CMF. These include four traditional techniques, zero imputation, mean imputation, random-walk imputation, and multiple imputation (MI), and two machine learning-based techniques, k-NNI and i-SVD. Later Section A.2 provides the additional result from the single sorted analysis on the impact of analysts' size and volatility on models earning prediction capacity.

A.1 Benchmark Methods for MF and CMF

Zero Imputation and Mean Imputation The most straightforward imputation techniques are zero imputation and mean imputation. Zero imputation involves filling the missing values with zero.

$$ZI(X_{tn}) = \begin{cases} 0 & \text{if } X_{tn} \text{ is a missing value} \\ X_{tn} & \text{otherwise} \end{cases}$$

Zero imputation induces a series of noise for many real data sets [112, 113]. As a result, many researchers prefer to use mean imputation. Mean imputation involves filling the missing values with the mean values of the data series. For the analyst EPS forecasting with time-series data, a simple mean over the time dimension is not effective as the firms' financial conditions change over time. A firm EPS in the first quarter of 2021 is the most likely to largely deviate from the reported EPS of the first quarter of 2005. Therefore, the mean over historical data may not be a good

proxy for the future EPS. The standard practice in this area is an augmented mean imputation, called *class mean imputation*, that estimate the missing values with the average of existing values from the same class (all the available analyst at a given time period) as the data instances to be imputed.

$$Mean(X_{tn}) = \begin{cases} \frac{\sum_{n'=0}^{|\Omega|} X_{tn'}}{|\Omega|} & \text{if } X_{tn} \text{ is missing value} \\ X_{tn} & \text{otherwise} \end{cases}$$

Where Ω is a set of the observed instances, and the quarter t is the class label. The missing values are replaced by the average of set Ω with the same class label, i.e., quarter t. Throughout the paper, the mean imputation refers to the class mean imputation.

Random-Walk Imputation Random-walk imputation involves imputing missing values with the last available forecast value of an analyst. However, in the studied data, a large number of analysts did not even have any previous forecast, and in many cases, analysts have very old forecast for a firm, i.e., they follow one firm, stop following, and re-follow after several quarters. Firms' financial conditions change over time, and as a result, an old forecast does not accurately portray the firm's current financial condition. To mitigate these scenarios, I use the firm's last available real EPS because the real EPS at $t-1$ is more reliable than an analyst forecast of $t-k$ period for $k > 1$.

$$\textit{Random-Walk}(X_{tn}) = \begin{cases} X_{t-1,n} & \text{if } X_{tn} \text{ is missing \& } X_{t-1,n} \text{ is available} \\ y_{t-1} & \text{if } X_{tn} \text{ is missing \& } X_{t-1,n} \text{ is not available} \\ X_{tn} & \text{otherwise} \end{cases}$$

Where y is the actual EPS value of the firm in the last reported $(t-1)$ period.

Multiple Imputation Multiple imputations (MI) proposed by [112] involves filling in the missing values multiple times. The multiple imputed results for each missing value account for the uncertainties in the imputation process and yield accurate results. Multiple imputations work for both missing at random (MAR) and missing not at random (MNAR). For the Bayesian theory of MI, modeling the response mechanism and underlying assumption of MAR and MNAR are detailed in [112]. I chose the Multiple Imputation with Chained Equations (MICE) algorithm [253] for baseline. MICE models and impute each variable with missing values as a function of other variables in an iterative procedure. The iteration is run until the model converges. The iterative algorithm of the chained equation process includes the following steps:

1. Perform a simple imputation, e.g., mean imputation, to impute the missing values in each variable with a temporary "place holder".
2. For one variable ("var"), set the "place holder" back to missing.
3. Perform a predictive model, e.g., regression, on the observed values of "var" using other variables in the datasets.
4. Use the fitted model in the previous step to predict the missing values in "var".
5. Repeat step ii-iv for each variable with missing data.
6. Repeat step ii-v until convergence.

In experimental setting, I choose the mean imputation for initialization in step-i; in step-ii, the analysts with missing values are selected for imputation based on an ascending order; in step-iii, I use all other variables (features) to fit the predictive model. For each imputation, I sample from the Gaussian predictive posterior of the fitted estimator. The maximum number of iterations is 10.

***k*-NNI** k-NNI is based on the k-nearest neighbor algorithm. It involves imputing a missing value based on the average value of its k-nearest neighbors. k-NNI was first

proposed by [115] and after that several variations of k-NNI have been developed. These includes the weighted k-NNI [254], the k-NNI with mutual information [255], and the k-NNI with penalized dissimilarity [256]. The rationale behind using k-NNI for data imputation is that an analyst EPS forecast can be approximated by the forecast of his/her closest analysts.

$$kNNI(X_{tn}) = \begin{cases} \frac{\sum_{k=1}^{K} X_{tk}}{|K|}, & \text{if } X_{tn} \text{ is missing value} \\ X_{tn}, & \text{otherwise} \end{cases}$$

The hyper-parameter K defines the number of neighbors: A low K ensures the local effect, and is less generalizable. In contrast, a high K deemphasizes the local effect. To select the K for this study, I perform cross-validation with multiple values (3, 5, 10, 15, 20) and select the best performing K, i.e., $K = 5$.

iSVD Singular value decomposition (SVD) based algorithms gained significant attention in recent years. SVD is the eigendecomposition of a regular matrix to two square matrices via an extension of the polar decomposition. However, SVD only works for a complete matrix. Iterative SVD (iSVD) was proposed as a workaround for this limitation [115]. Similar to principal component analysis, iSVD generates a set of mutually orthogonal vectors that form a base to the original matrix and use the linear combination of vector elements to approximate the missing values [121, 257, 258], as shown in the following equation:

$$iSVD(X_{tn}) = \begin{cases} U_{tt}\Sigma_{tn}V_{nn}^{\top}, & \text{if } X_{tn} \text{ is missing value} \\ X_{tn}, & \text{otherwise} \end{cases}$$

Here, matrix $V^\top$ contains the right singular vector whose values are quantified by the diagonal matrix Σ. The most significant vectors (K) are selected from $V^\top$ by sorting their corresponding singular values. A low-rank assumption means that only a few singular vectors are necessary to explain the entire data. Here the rank K is determined empirically [121, 259]. Finally, the missing value n for period t is estimated by combining the K outer-products of the left and right singular vectors.

The U and V matrix represent the time period and analyst respectively. For determining the coefficient, the nth value of period t and the nth values of the K eigenvector are ignored. To overcome the complete matrix constraint of SVD, initially, all the missing values are substituted with the row average of matrix X, obtaining X'. iSVD uses the expectation-maximization (EM), an iterative algorithm: at the beginning of each iteration, estimating the missing values in X' by taking the average, and creating a new X' until X' converges and the total change in the matrix falls below the empirically determined threshold of 0.01 (For details see [115]).

A.2 Single Sorted Results on the Impact of Analyst Size and Volatility

Table A.1 reports the performance of the models in predicting firms' earnings in terms of average R^2, average MSE, and average MAPE. The top panel of the table represents the grouping based on analysts' size and the bottom one represents the grouping based on volatility. Consistent with the result reported in Figure 2.6a, the performance of all models are sensitive to the number of analysts. For the high-analysts group, the prediction accuracy of all models is satisfactory (MF+SVR $\geq$ 90%). However, the performance drops more than 20% for the low-analysts group. Especially the R^2 value for XGBoost becomes negative for the moderate-analysts and low-analysts groups. In line with the findings of section 2.4.2, MF+XGBoost and MF+SVR outperform all other models in all three firm groups. The performance improvement in R^2 over the benchmark mean prediction is 7.6% for the high-analysts

Table A.1 Impact of Analysts Forecast Volatility on Earning Prediction (Single Sorted)

	High Analysts			Moderate Analysts			Low Analysts		
Method	R^2	MSE	MAPE	R^2	MSE	MAPE	R^2	MSE	MAPE
Mean	0.8511	0.1581	31.2331	0.7833	0.7979	41.0821	0.6683	0.5593	47.6578
Random Walk	0.4118	0.5215	82.7396	0.2425	1.9835	115.6855	0.0490	5.4194	143.9878
XGBoost	0.6995	0.0866	33.0309	-0.9450	1.8086	64.4451	-1.5421	0.6629	74.4921
MI+LASSO	0.8250	0.2091	36.8890	0.5758	0.4364	55.8177	0.3404	0.6251	84.3804
MI+XGBoost	0.8767	0.0728	28.6681	0.7667	0.2835	46.6041	0.3404	0.6251	84.3804
MI+SVR	0.8927	0.0836	27.7566	0.7754	0.2835	45.4611	0.4531	0.5644	72.9610
MF+LASSO	0.7838	0.2663	46.1921	0.6394	0.7552	40.1946	0.6438	0.6072	45.3521
MF+XGBoost	**0.9162**	**0.0290**	**10.2132**	**0.8210**	0.0902	**24.5307**	**0.6920**	0.1429	**31.5714**
MF+SVR	0.9126	0.0554	10.2925	0.8103	**0.0797**	25.1345	0.6691	**0.1246**	32.0147
	High Volatility			Moderate Volatility			Low Volatility		
Mean	0.6547	0.8183	80.4632	0.7611	0.6754	33.6700	**0.8869**	0.0217	**10.0231**
Random Walk	0.0024	2.0909	205.0657	0.1796	5.7024	107.8180	0.5186	0.0963	30.6905
XGBoost	-2.3223	2.1399	112.6231	-0.1491	0.3999	56.8142	0.6838	0.0183	17.2941
MI+LASSO	0.4410	0.8073	101.9210	0.5308	0.4027	48.5348	0.7694	0.0606	26.6313
MI+XGBoost	0.5832	0.5923	81.5188	0.6777	0.2484	40.3472	0.8474	0.0385	21.5675
MI+SVR	0.5903	0.6551	85.9804	0.7052	0.2321	40.5121	0.8257	0.0442	19.6862
MF+LASSO	0.6907	0.6859	76.2421	0.7710	0.6018	30.3554	0.8053	0.1408	13.8207
MF+XGBoost	**0.7447**	**0.1282**	**40.0821**	**0.8135**	0.1145	**23.0365**	0.8710	**0.0194**	10.2132
MF+SVR	0.7393	0.1485	40.1456	0.7954	**0.0891**	23.3821	0.8574	0.0267	10.5214

Note: In the top panel, firms are sorted on descending order based on the average number of analysts following a firm over time; top 33% represents the high analysts' group, bottom 33% represents the low analysts' group, and the remaining 34% represents the moderate analysts' group.
In the bottom panel, firms are sorted on descending order based on the average standard deviation of analysts earning forecast for a firm at a given quarter. Top 33% represents the high volatility group, bottom 33% represents the low volatility group, and the remaining 34% represents the moderate volatility group.
R^2, MSE, and MAPE are calculated using Eqn. 2.14, 2.15, and, 2.16 respectively. Smaller MSE, MAPE and larger R^2 indicate better accuracy. The reported values are from test data set.

group, 4.8% for the moderate-analysts group, and 3.5% for the low-analysts group. The difference in performance improvement also signifies the impact of data sparsity on the MF imputation. For the low-analysts group, too many missing values of the original data affect the imputation performance and eventually impair the prediction accuracy.

The bottom panel of the Table A.1 shows the influence of analysts' volatility in predicting earnings, which matches with the result reported in Figure 2.6b. When analysts are in consensus about a firm's future earnings, i.e., the forecast volatility is low, almost all models predict the earnings in the next quarter at high accuracy. In contrast, when analysts do not conform well with each other on a firm's projected earning, i.e., high volatility, the model prediction accuracy drops significantly. In

the low-volatility group, the simple mean prediction outperforms the sophisticated machine learning method. However, MF based models are less sensitive to analysts' forecast volatility. Considering a 26% drop in the mean prediction and a 40% drop in XGBoost, both MF+XGBoost and MF+SVR only experience a 14% drop from the low-volatility group to high-volatility group, a significant improvement over all models. Further experiments show that MF+XGBoost and MF+SVR demonstrate superior performance over the mean prediction consistently, i.e., 6.8% and 13.3% improvements in R^2 respectively, on both moderate-volatility and high-volatility groups.

APPENDIX B

INDIVIDUAL INDEX RETURN

Appendix B provides the individual indices average factor return on portfolio based on Fama-French and latent factor from Chapter 4. The 18 (2×3×3) portfolio returns reported are developed based on the size, value, and communal difference. Table B.1 reports the results from S&P-500 stocks, Table B.2 reports the results from RUSSELL-3000 stocks, and Table B.3 reports the results from NASDAQ-100 stocks.

Table B.1 Daily Average Return on Factor Portfolio for S&P-500 Stocks

	Big/High	Big/Medium	Big/Low	Small/High	Small/Medium	Small/Low
Communal	0.0240	0.0375	0.0394	0.0357	0.0537	0.0514
Moderate	0.0531	0.0540	0.0548	0.0540	0.0785	0.0834
Non-Communal	0.0824	0.0808	0.1284	0.0808	0.1600	0.1699

Table B.2 Daily Average Return on Factor Portfolio for RUSSELL-3000 Stocks

	Big/High	Big/Medium	Big/Low	Small/High	Small/Medium	Small/Low
Communal	0.0385	0.0415	0.0406	0.0642	0.0466	0.1680
Moderate	0.0599	0.0642	0.0778	0.0944	0.0882	0.1023
Non-Communal	0.1057	0.1049	0.1030	0.1425	0.1301	0.1388

Table B.3 Daily Average Return on Factor Portfolio for NASDAQ-100 Stocks

	Big/High	Big/Medium	Big/Low	Small/High	Small/Medium	Small/Low
Communal	0.0688	0.0768	0.0271	0.1335	0.0998	0.1312
Moderate	0.1069	0.1027	0.0590	0.1493	0.1223	0.1219
Non-Communal	0.1026	0.1216	0.0473	0.1580	0.1321	0.1471

APPENDIX C

EQUITY NETWORK CONSTRUCTION STRATEGY AND ROBUSTNESS TEST

Appendix C first provides the details of network construction strategy in Section C.1 form Chapter 5 and then in Section C.2 provides the robustness test result of Fama-MacBeth Regression on Next Month Excess Return.

C.1 Network Construction

The correlations among firms market return can serve as a good proxy for a firm network [233, 234]. Following the literature, I use the correlation of historical returns as a measurement of firm proximity and calculate the Pearson's correlation $\rho_{ij,t} = \rho(r_{i,t},\ r_{j,t})$ for firms i and j at time t. For each month, I calculate the end-of-month correlation $\rho_{ij,t}$ using daily returns in that month. The original correlation matrix of returns is dense, and almost all firms' returns are (positively or negatively) correlated with each other. The values close to zero do not provide any useful information about firm affinity but noise. To enhance Signal-to-noise ratio (SNR), a threshold on the correlation matrix of all firms' returns is applied to sparsify the equity network graph G_t, i.e., $W_{ij,t} = \rho_{ij,t}$ if $|\rho_{ij,t}| \geq \xi$ otherwise 0. I select $\xi = 0.40$ following [216, 233]. The sparcified similarity matrix also reduces the computational cost of the otherwise complex task of dealing with a nearly complete graph.[1]

C.2 Robustness Test

The construction of the Z-score involves two main hyperparameters: the number of the smallest eigenvalues k to calculate current network state λ and the size of the sliding window m to select the number of previous months network states for building

[1] A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

Table C.1 Fama-MacBeth Regression on Next Month Excess Return: Robustness Test

	$m = 12$		$m = 24$		$m = 36$		$m = 48$		$m = 60$	
Panel A: $k = 10$										
β_Z	-013^{***}	(-3.39)	-017^{***}	(-3.88)	-015^{***}	(-3.53)	-014^{***}	(-3.26)	-014^{***}	(-3.36)
β_{MKT}	0.16	(1.65)	0.16	(1.34)	0.02	(0.16)	0.07	(0.54)	0.08	(0.60)
β_{SMB}	0.35^{***}	(2.72)	0.22^{*}	(1.76)	0.08	(0.65)	0.09	(0.73)	0.16	(1.34)
β_{HML}	-010	(-0.88)	-001	(-0.12)	0.11	(0.96)	0.07	(0.65)	0.04	(0.43)
β_{UMD}	-014^{*}	(-1.72)	-020^{**}	(-2.42)	-016^{**}	(-2.07)	-019^{**}	(-2.32)	-020^{**}	(-2.40)
Constant	0.82^{***}	(4.23)	0.84^{***}	(4.25)	0.75^{***}	(3.68)	0.78^{***}	(4.15)	0.81^{***}	(4.23)
R^2	3.70		3.60		3.61		3.70		3.70	
Panel B: $k = 15$										
β_Z	-013^{***}	(-3.16)	-015^{***}	(-3.51)	-014^{***}	(-3.23)	-015^{***}	(-3.45)	-014^{***}	(-3.43)
β_{MKT}	0.16	(1.62)	0.15	(1.18)	0.02	(0.15)	0.04	(0.28)	0.07	(0.56)
β_{SMB}	0.35^{***}	(2.78)	0.19	(1.58)	0.08	(0.66)	0.09	(0.79)	0.14	(1.16)
β_{HML}	-010	(-0.94)	0.01	(0.01)	0.07	(0.62)	0.05	(0.45)	0.07	(0.63)
β_{UMD}	$-014*$	(-1.70)	-017^{**}	(-2.06)	-019^{**}	(-2.38)	-021^{**}	(-2.52)	-019^{**}	(-2.28)
Constant	0.84^{***}	(4.43)	0.80^{***}	(4.13)	0.77^{***}	(3.96)	0.79^{***}	(-4.19)	0.82^{***}	(4.37)
R^2	3.71		3.63		3.61		3.70		3.72	
Panel C: $k = 20$										
β_Z	-013^{***}	(-3.15)	-015^{***}	(-3.71)	-015^{***}	(-2.99)	-014^{***}	(-3.43)	-014^{***}	(-3.36)
β_{MKT}	0.19^{*}	(1.91)	0.09	(0.84)	0.04	(0.29)	0.01	(0.03)	0.04	(0.35)
β_{SMB}	0.36^{***}	(2.91)	0.18	(1.45)	0.08	(0.68)	0.08	(0.71)	0.13	(1.13)
β_{HML}	-011	(-0.98)	-002	(-0.18)	0.06	(0.55)	0.06	(0.57)	0.05	(0.52)
β_{UMD}	-011	(-1.36)	-015^{*}	(-1.89)	-018^{**}	(-2.19)	-019^{**}	(-2.37)	-0178^{**}	(-2.11)
Constant	0.882	(4.90)	0.77^{***}	(4.06)	0.78^{***}	(3.99)	0.76^{***}	(4.03)	0.81^{***}	(4.30)
R^2	3.63		3.63		3.64		3.70		3.71	

Note: This table reports the average value of Fama-MacBeth regressions with Z-scores calculated using different hyper-parameters. The dependent variable is the next-month future stock returns. Here, k represents the number of lowest eigenvalues used to calculate network Laplacian Spectrum λ_t and m represent the sliding window size for constructing the context matrix C. Newey-West adjusted t-statistics are in parentheses. ***, **, * are significant at the 1%, 5%, and 10% significance level, respectively.

context matrix C. In this section, I perform a sensitivity test with different selections of these two hyperparameters to ensure the robustness of the proposed model. I use three different values of k, i.e., 10, 15, 20 and five different values of m, i,e, 12, 24, 36, 48, and 60. Table C.1 reports the results of the sensitivity analysis.

Same as Section 5.3.4, the Fama-MacBeth regressions control for the market, size, value, and momentum factor. Consistently, firms with high (positive) sensitivity to Z-score have lower future returns, and this negative relation is more significant when Z-score is higher than the historical median. In Table C.1, I only report the

results when Z-score is high, i.e., the market network experiences substantial changes. Panel A in Table C.1 reports the cross-sectional regression results with $k = 10$, Panel B shows the results with $k = 15$, and Panel C presents the results when $k = 20$. For all the models, the coefficients of Beta of Z-score are negative and highly significant. The coefficient ranges between -0.125 (the largest observed value when $k = 15$ and $k = 12$) and -0.167 (the smallest value observed when $k = 10$ and $m = 24$). Particularly, the sensitivity of the size of the coefficient is not systematic to k. However, when the window size m increases, the coefficient slightly increases. The size and significance of the beta coefficients for control variables remain similar to each other for all models, except $k = 20$ and $m = 12$, where all the traditional factors are insignificant.

The sensitivity test results reported in Table C.1 indicate that the network factor is not sensitive to the choice of the number of eigenvalues or the number of months. With a small expected variance, proposed methodology can identify the network factor effectively. The choices of k and m in this paper are based on network theory, Eigengap, and practicality. A value of $k = 20$ is big enough to capture the maximum information from the Laplacian spectrum, yet small enough to filter out noise from the graph Laplacian [222]. The network factor is also not computationally expensive. Using the Laplacian spectrum calculated from correlations of previous thirty-six months (three years) incorporates enough data points to represent normal market behavior. It also contains an economic cycle. A small m, e.g., 6 or 12 months, might introduce too much volatility and can not accurately represent normal market behavior.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 373–374.

[3] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[4] I. Wallach, M. Dzamba, and A. Heifets, "Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery," *arXiv preprint arXiv:1510.02855*, 2015.

[5] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *Proceedings of the 32nd International Conference on Machine Learning*. JMLR.org, 2015, pp. 1766–1774.

[6] C. J. Maddison, A. Huang, I. Sutskever, and D. Silver, "Move evaluation in go using deep convolutional neural networks," *arXiv preprint arXiv:1412.6564*, 2014.

[7] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proceedings of the 19th Conference on Business Informatics (CBI)*, vol. 1. IEEE, 2017, pp. 7–12.

[8] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[9] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2019, pp. 3009–3017.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[13] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[14] Y. Yuan, M. Chao, and Y.-C. Lo, "Automatic skin lesion segmentation using deep fully convolutional networks with jaccard distance," *IEEE Transactions on Medical Imaging*, vol. 36, no. 9, pp. 1876–1886, 2017.

[15] H. Chen, Q. Dou, L. Yu, J. Qin, and P.-A. Heng, "Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images," *NeuroImage*, vol. 170, pp. 446–455, 2018.

[16] S. Shichijo, S. Nomura, K. Aoyama, Y. Nishikawa, M. Miura, T. Shinagawa, H. Takiyama, T. Tanimoto, S. Ishihara, K. Matsuo *et al.*, "Application of convolutional neural networks in the diagnosis of helicobacter pylori infection based on endoscopic images," *EBioMedicine*, vol. 25, pp. 106–111, 2017.

[17] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio." in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 729–734.

[18] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.

[19] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint word-image embeddings," *Machine Learning*, vol. 81, no. 1, pp. 21–35, 2010.

[20] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Proceedings of the Advances in Neural Information Processing Systems*, 2012, pp. 2222–2230.

[21] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the International Conference on Machine learning (ICML)*. Omnipress, 2011, pp. 513–520.

[22] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 151–161.

[23] C. M. Lee and E. C. So, "Uncovering expected returns: Information in analyst coverage proxies," *Journal of Financial Economics*, vol. 124, no. 2, pp. 331–348, 2017.

[24] D. Fried and D. Givoly, "Financial analysts' forecasts of earnings: A better surrogate for market expectations," *Journal of Accounting and Economics*, vol. 4, no. 2, pp. 85–107, 1982.

[25] A. W. Alford and P. G. Berger, "A simultaneous equations analysis of forecast accuracy, analyst following, and trading volume," *Journal of Accounting, Auditing and Finance*, vol. 14, no. 3, pp. 219–240, 1999.

[26] W. Kross, B. Ro, and D. Schroeder, "Earnings expectations: The analysts' information advantage," *Accounting Review*, pp. 461–476, 1990.

[27] T. Sougiannis and T. Yaekura, "The accuracy and bias of equity values inferred from analysts' earnings forecasts," *Journal of Accounting, Auditing and Finance*, vol. 16, no. 4, pp. 331–362, 2001.

[28] P. C. O'brien, "Analysts' forecasts as earnings expectations," *Journal of Accounting and Economics*, vol. 10, no. 1, pp. 53–83, 1988.

[29] L. D. Brown, "Earnings forecasting research: its implications for capital markets research," *International Journal of Forecasting*, vol. 9, no. 3, pp. 295–320, 1993.

[30] R. D. Harris and P. Wang, "Model-based earnings forecasts vs. financial analysts' earnings forecasts," *The British Accounting Review*, vol. 51, no. 4, pp. 424–437, 2019.

[31] R. Jame, R. Johnston, S. Markov, and M. C. Wolfe, "The value of crowdsourced earnings forecasts," *Journal of Accounting Research*, vol. 54, no. 4, pp. 1077–1110, 2016.

[32] D. Bradley, S. Gokkaya, and X. Liu, "Before an analyst becomes an analyst: Does industry experience matter?" *The Journal of Finance*, vol. 72, no. 2, pp. 751–792, 2017.

[33] R. T. Ball and E. Ghysels, "Automated earnings forecasts: Beat analysts or combine and conquer?" *Management Science*, vol. 64, no. 10, pp. 4936–4952, 2018.

[34] P. Goupillaud, A. Grossmann, and J. Morlet, "Cycle-octave and related transforms in seismic signal analysis," *Geoexploration*, vol. 23, no. 1, pp. 85–102, 1984.

[35] S. Huang, H. An, X. Gao, and X. Huang, "Time–frequency featured co-movement between the stock and prices of crude oil and gold," *Physica A: Statistical Mechanics and its Applications*, vol. 444, pp. 985–995, 2016.

[36] M. B. Mikhail, B. R. Walther, and R. H. Willis, "Do security analysts improve their performance with experience?" *Journal of Accounting Research*, vol. 35, pp. 131–157, 1997.

[37] F. Brochet, G. S. Miller, and S. Srinivasan, "Do analysts follow managers who switch companies? an analysis of relationships in the capital markets," *The Accounting Review*, vol. 89, no. 2, pp. 451–482, 2013.

[38] L. Cohen, A. Frazzini, and C. Malloy, "Sell-side school ties," *The Journal of Finance*, vol. 65, no. 4, pp. 1409–1437, 2010.

[39] S. Das, C. B. Levine, and K. Sivaramakrishnan, "Earnings predictability and bias in analysts' earnings forecasts," *Accounting Review*, pp. 277–294, 1998.

[40] S. Chen and D. A. Matsumoto, "Favorable versus unfavorable recommendations: The impact on analyst access to management-provided information," *Journal of Accounting Research*, vol. 44, no. 4, pp. 657–689, 2006.

[41] K. Hou, M. A. Van Dijk, and Y. Zhang, "The implied cost of capital: A new approach," *Journal of Accounting and Economics*, vol. 53, no. 3, pp. 504–526, 2012.

[42] D. Ayres, X. S. Huang, and M. Myring, "Fair value accounting and analyst forecast accuracy," *Advances in Accounting*, vol. 37, pp. 58–70, 2017.

[43] C. R. Aberger, "Recommender: An analysis of collaborative filtering techniques," *Personal and Ubiquitous Computing Journal*, 2014.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[45] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2016, pp. 785–794.

[46] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the Advances in Neural Information Processing Systems*, 1997, pp. 155–161.

[47] S. Gu, B. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223–2273, 2020.

[48] B. T. Kelly, S. Pruitt, and Y. Su, "Characteristics are covariances: A unified model of risk and return," *Journal of Financial Economics*, vol. 134, no. 3, pp. 501–524, 2019.

[49] E. F. Fama and K. R. French, "A five-factor asset pricing model," *Journal of Financial Economics*, vol. 116, no. 1, pp. 1–22, 2015.

[50] Y. Sha and R. Gao, "Which is the best: A comparison of asset pricing factor models in chinese mutual fund industry," *Economic Modelling*, vol. 83, pp. 8–16, 2019.

[51] G. Feng, S. Giglio, and D. Xiu, "Taming the factor zoo: A test of new factors," *The Journal of Finance*, vol. 75, no. 3, pp. 1327–1370, 2020.

[52] W. F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk," *The Journal of Finance*, vol. 19, no. 3, pp. 425–442, 1964.

[53] S. A. Ross, "The arbitrage theory of capital asset pricing," *Journal of Economic Theory*, vol. 13, no. 3, pp. 341 – 360, 1976.

[54] H. M. Walker, "Degrees of freedom." *Journal of Educational Psychology*, vol. 31, no. 4, p. 253, 1940.

[55] J. R. Birge and Y. Zhang, "Risk factors that explain stock returns: A non-linear factor pricing model," *Available at Social Science Research Network (SSRN) 3061712*, 2018.

[56] L. Cohen and A. Frazzini, "Economic links and predictable returns," *The Journal of Finance*, vol. 63, no. 4, pp. 1977–2011, 2008.

[57] V. Muslu, M. Rebello, and Y. Xu, "Sell-side analyst research and stock comovement," *Journal of Accounting Research*, vol. 52, no. 4, pp. 911–954, 2014.

[58] G. Grullon, S. Underwood, and J. P. Weston, "Comovement and investment banking networks," *Journal of Financial Economics*, vol. 113, no. 1, pp. 73–89, 2014.

[59] B. Herskovic, "Networks in production: Asset pricing implications," *The Journal of Finance*, vol. 73, no. 4, pp. 1785–1818, 2018.

[60] L. F. Di Cerbo and S. Taylor, "Graph theoretical representations of equity indices and their centrality measures," *Quantitative Finance*, pp. 1–15, 2020.

[61] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[62] H. Huang, S. Yoo, D. Yu, and H. Qin, "Density-aware clustering based on aggregated heat kernel and its transformation," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 4, pp. 1–35, 2015.

[63] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," *Advances in Neural Information Processing Systems*, vol. 32, pp. 13 354–13 366, 2019.

[64] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 2001–2009.

[65] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.

[66] C. Li, D. Song, and D. Tao, "Multi-task recurrent neural networks and higher-order markov random fields for stock price movement prediction: Multi-task rnn and higer-order mrfs for stock price classification," in *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2019, pp. 1141–1151.

[67] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, 2014.

[68] O. Post, *Spectral analysis on graph-like spaces*. Berlin, Germany: Springer Science and Business Media, 2012, vol. 2039.

[69] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2016, pp. 855–864.

[70] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2014, pp. 701–710.

[71] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[72] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations, ICLR, Toulon, France*, 2017.

[73] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3844–3852, 2016.

[74] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[75] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proceedings of the International Conference on Learning Representations*, 2018.

[76] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "Gstnet: Global spatial-temporal network for traffic flow prediction." in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 2286–2293.

[77] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 890–897.

[78] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3656–3663.

[79] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong *et al.*, "Spectral temporal graph neural network for multivariate time-series forecasting," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[80] M. Tumminello, F. Lillo, and R. N. Mantegna, "Correlation, hierarchies, and networks in financial markets," *Journal of Economic Behavior and Organization*, vol. 75, no. 1, pp. 40–58, 2010.

[81] M. Elliott, B. Golub, and M. O. Jackson, "Financial networks and contagion," *American Economic Review*, vol. 104, no. 10, pp. 3115–53, 2014.

[82] M. K. So, A. M. Chu, and T. W. Chan, "Impacts of the covid-19 pandemic on financial market connectedness," *Finance Research Letters*, p. 101864, 2020.

[83] T. H. Le, H. X. Do, D. K. Nguyen, and A. Sensoy, "Covid-19 pandemic and tail-dependency networks of financial assets," *Finance Research Letters*, p. 101800, 2020.

[84] L. C. Rogers and L. A. Veraart, "Failure and rescue in an interbank network," *Management Science*, vol. 59, no. 4, pp. 882–898, 2013.

[85] B.-H. Hwang and S. Kim, "It pays to have friends," *Journal of Financial Economics*, vol. 93, no. 1, pp. 138–158, 2009.

[86] L. Bruynseels and E. Cardinaels, "The audit committee: Management watchdog or personal friend of the ceo?" *The Accounting Review*, vol. 89, no. 1, pp. 113–145, 2014.

[87] D. Acemoglu, A. Ozdaglar, and A. Tahbaz-Salehi, "Systemic risk and stability in financial networks," *American Economic Review*, vol. 105, no. 2, pp. 564–608, 2015.

[88] P. Glasserman and H. P. Young, "How likely is contagion in financial networks?" *Journal of Banking and Finance*, vol. 50, pp. 383–399, 2015.

[89] P. Glasserman and H. P. Young, "Contagion in financial networks," *Journal of Economic Literature*, vol. 54, no. 3, pp. 779–831, 2016.

[90] M. Billio, M. Caporin, R. C. Panzica, and L. Pelizzon, "The impact of network connectivity on factor exposures, asset pricing and portfolio diversification," Leibniz Institute for Financial Research SAFE, SAFE Working Paper Series 166, 2017.

[91] A. Buraschi and P. Porchia, "Dynamic networks and asset pricing," in *American Finance Association (AFA) 2013 San Diego Meetings Paper*, 2012.

[92] V. M. Carvalho, "From micro to macro via production networks," *Journal of Economic Perspectives*, vol. 28, no. 4, pp. 23–48, 2014.

[93] A. Ozdagli and M. Weber, "Monetary policy through production networks: Evidence from the stock market," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., 2017.

[94] K. Hou, "Industry information diffusion and the lead-lag effect in stock returns," *The Review of Financial Studies*, vol. 20, no. 4, pp. 1113–1138, 2007.

[95] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.

[96] S. Gu, B. Kelly, and D. Xiu, "Autoencoder asset pricing models," *Journal of Econometrics*, vol. 222, no. 1, pp. 429–450, 2021.

[97] A. Uddin and D. Yu, "Latent factor model for asset pricing," *Journal of Behavioral and Experimental Finance*, vol. 27, p. 100353, 2020.

[98] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2017.

[99] P. Yu and X. Yan, "Stock price prediction based on deep neural networks," *Neural Computing and Applications*, vol. 32, no. 6, pp. 1609–1628, 2020.

[100] L. Chen, M. Pelger, and J. Zhu, "Deep learning in asset pricing," *Available at Social Science Research Network (SSRN) 3350138*, 2020.

[101] 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.

[102] 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.

[103] W. Li, R. Bao, K. Harimoto, D. Chen, J. Xu, and Q. Su, "Modeling the stock relation with graph network for overnight stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 7 2020, pp. 4541–4547.

[104] L. D. Brown, R. L. Hagerman, P. A. Griffin, and M. E. Zmijewski, "Security analyst superiority relative to univariate time-series models in forecasting quarterly earnings," *Journal of Accounting and Economics*, vol. 9, no. 1, pp. 61–87, 1987.

[105] M. T. Bradshaw, M. S. Drake, J. N. Myers, and L. A. Myers, "A re-examination of analysts' superiority over time-series forecasts of annual earnings," *Review of Accounting Studies*, vol. 17, no. 4, pp. 944–968, 2012.

[106] P. C. O'Brien, "Forecast accuracy of individual analysts in nine industries," *Journal of Accounting Research*, vol. 28, no. 2, pp. 286–304, 1990.

[107] P. Sinha, L. D. Brown, and S. Das, "A re-examination of financial analysts' differential earnings forecast accuracy," *Contemporary Accounting Research*, vol. 14, no. 1, pp. 1–42, 1997.

[108] M. B. Clement and S. Y. Tse, "Do investors respond to analysts' forecast revisions as if forecast accuracy is all that matters?" *The Accounting Review*, vol. 78, no. 1, pp. 227–249, 2003.

[109] J. C. Easterwood and S. R. Nutt, "Inefficiency in analysts' earnings forecasts: Systematic misreaction or systematic optimism?" *The Journal of Finance*, vol. 54, no. 5, pp. 1777–1797, 1999.

[110] S. Ramnath, S. Rock, and P. Shane, "The financial analyst forecasting literature: A taxonomy with suggestions for further research," *International Journal of Forecasting*, vol. 24, no. 1, pp. 34–75, 2008.

[111] R. T. Ball and E. Ghysels, "Automated earnings forecasts: Beat analysts or combine and conquer?" *Management Science*, vol. 64, no. 10, pp. 4936–4952, 2017.

[112] D. B. Rubin, *Multiple imputation for nonresponse in surveys*. Hoboken, NJ: John Wiley and Sons, 2004, vol. 81.

[113] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[114] C.-H. Cheng, C.-P. Chan, and Y.-J. Sheu, "A novel purity-based k nearest neighbors imputation method and its application in financial distress prediction," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 283–299, 2019.

[115] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[116] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, p. 717, 2009.

[117] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[118] J. W. Graham, "Missing data analysis: Making it work in the real world," *Annual Review of Psychology*, vol. 60, pp. 549–576, 2009.

[119] H. Kang, "The prevention and handling of the missing data," *Korean Journal of Anesthesiology*, vol. 64, no. 5, p. 402, 2013.

[120] E. F. Fama and K. R. French, "Profitability, investment and average returns," *Journal of Financial Economics*, vol. 82, no. 3, pp. 491–518, 2006.

[121] O. Alter, P. O. Brown, and D. Botstein, "Singular value decomposition for genome-wide expression data processing and modeling," *Proceedings of the National Academy of Sciences*, vol. 97, no. 18, pp. 10 101–10 106, 2000.

[122] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2011, pp. 69–77.

[123] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[124] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, "A collaborative filtering approach to mitigate the new user cold start problem," *Knowledge-Based Systems*, vol. 26, pp. 225–238, 2012.

[125] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[126] E. Acar, G. Gürdeniz, M. A. Rasmussen, D. Rago, L. O. Dragsted, and R. Bro, "Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics," in *Proceedings of the 12th International Conference on Data Mining Workshops*. IEEE, 2012, pp. 1–8.

[127] F. M. Almutairi, N. D. Sidiropoulos, and G. Karypis, "Context-aware recommendation-based learning analytics using tensor and coupled matrix factorization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 5, pp. 729–741, 2017.

[128] M. T. Bradshaw, "Analysts' forecasts: what do we know after decades of work?" *Available at Social Science Research Network (SSRN) 1880339*, 2011.

[129] S. P. Kothari, E. So, and R. Verdi, "Analysts' forecasts and asset pricing: A survey," *Annual Review of Financial Economics*, vol. 8, pp. 197–219, 2016.

[130] W. Ma and G. H. Chen, "Missing not at random in matrix completion: The effectiveness of estimating missingness probabilities under a low nuclear norm assumption," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[131] M. B. Clement, "Analyst forecast accuracy: Do ability, resources, and portfolio complexity matter?" *Journal of Accounting and Economics*, vol. 27, no. 3, pp. 285–303, 1999.

[132] F. Provost and T. Fawcett, *Chapter five: Overfitting and Its Avoidance, Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*, 1st ed. O'Reilly Media, Inc., 2013.

[133] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis," 1970.

[134] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[135] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[136] G. Chamberlain and M. Rothschild, "Arbitrage, Factor Structure, and Mean-Variance Analysis on Large Asset Markets," *Econometrica*, vol. 51, no. 5, pp. 1281–1304, 1983.

[137] A. Uddin, X. Tao, C.-C. Chou, and D. Yu, "Are missing values important for earnings forecasts? a machine learning perspective," *Quantitative Finance*, pp. 1–20, 2021.

[138] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[139] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[140] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[141] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[142] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[143] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," *arXiv preprint arXiv:1105.3422*, 2011.

[144] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3642–3649.

[145] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[146] A. Bucci, "Realized volatility forecasting with neural networks," *Journal of Financial Econometrics*, vol. 18, no. 3, pp. 502–531, 2020.

[147] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[148] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[149] I.-K. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954–959, 2000.

[150] H. Liu, Y. Li, M. Tsang, and Y. Liu, "Costco: A neural tensor completion model for sparse tensors," in *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2019, p. 324–334.

[151] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

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

[153] J. H. van Binsbergen, X. Han, and A. Lopez-Lira, "Man vs. machine learning: The term structure of earnings expectations and conditional biases," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., 2020.

[154] M. McNichols and P. C. O'Brien, "Self-selection and analyst coverage," *Journal of Accounting Research*, vol. 35, pp. 167–199, 1997.

[155] H.-w. Lin and M. F. McNichols, "Underwriting relationships, analysts' earnings forecasts and investment recommendations," *Journal of Accounting and Economics*, vol. 25, no. 1, pp. 101–127, 1998.

[156] S. Du and J. Lee, "On the power of over-parametrization in neural networks with quadratic activation," in *Proceedings of the 35th International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2018, pp. 1329–1338.

[157] S. I. Lee and S. J. Yoo, "Threshold-based portfolio: the role of the threshold and its applications," *The Journal of Supercomputing*, pp. 1–18, 2018.

[158] L. Chen, Z. Qiao, M. Wang, C. Wang, R. Du, and H. E. Stanley, "Which artificial intelligence algorithm better predicts the chinese stock market?" *IEEE Access*, vol. 6, pp. 48 625–48 633, 2018.

[159] Q. Song, A. Liu, S. Y. Yang, A. Deane, and K. Datta, "An extreme firm-specific news sentiment asymmetry based trading strategy," in *Proceedings of the Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 898–904.

[160] S. Y. Yang, S. Y. K. Mo, and A. Liu, "Twitter financial community sentiment and its predictive relationship to stock market movement," *Quantitative Finance*, vol. 15, no. 10, pp. 1637–1656, 2015.

[161] M. R. Hassan, B. Nath, and M. Kirley, "A fusion model of hmm, ann and ga for stock market forecasting," *Expert Systems with Applications*, vol. 33, no. 1, pp. 171–180, 2007.

[162] C.-L. Huang and C.-Y. Tsai, "A hybrid sofm-svr with a filter-based feature selection for stock market forecasting," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1529–1539, 2009.

[163] Y. Kara, M. A. Boyacioglu, and Ö. K. Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5311–5319, 2011.

[164] M. Gorenc Novak and D. Velušček, "Prediction of stock price movement based on daily high prices," *Quantitative Finance*, vol. 16, no. 5, pp. 793–826, 2016.

[165] A.-S. Chen, M. T. Leung, and H. Daouk, "Application of neural networks to an emerging financial market: forecasting and trading the taiwan stock index," *Computers and Operations Research*, vol. 30, no. 6, pp. 901–923, 2003.

[166] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, 2017.

[167] Y. Pan, Z. Xiao, X. Wang, and D. Yang, "A multiple support vector machine approach to stock index forecasting with mixed frequency sampling," *Knowledge-Based Systems*, vol. 122, pp. 90–102, 2017.

[168] W. F. Sharpe, "The sharpe ratio," *Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 1994.

[169] J. L. Treynor, "Toward a theory of market value of risky assets," *Unpublished manuscript*, 1961.

[170] J. Lintner, "The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets: A reply," *The Review of Economics and Statistics*, pp. 222–224, 1969.

[171] J. Mossin, "Equilibrium in a capital asset market," *Econometrica: Journal of the econometric society*, pp. 768–783, 1966.

[172] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[173] E. F. Fama and K. R. French, "The cross-section of expected stock returns," *The Journal of Finance*, vol. 47, no. 2, pp. 427–465, 1992.

[174] R. Novy-Marx, "The other side of value: The gross profitability premium," *Journal of Financial Economics*, vol. 108, no. 1, pp. 1–28, 2013.

[175] S. Titman, K. J. Wei, and F. Xie, "Capital investments and stock returns," *Journal of Financial and Quantitative Analysis*, vol. 39, no. 4, pp. 677–700, 2004.

[176] K. Hou, C. Xue, and L. Zhang, "Digesting anomalies: An investment approach," *The Review of Financial Studies*, vol. 28, no. 3, pp. 650–705, 2015.

[177] E. F. Fama and K. R. French, "Common risk factors in the returns on stocks and bonds," *Journal of Financial Economics*, vol. 33, no. 1, pp. 3–56, 1993.

[178] B. Rosenberg and V. Marathe, *The prediction of investment risk: Systematic and residual risk*, 1975.

[179] M. M. Carhart, "On persistence in mutual fund performance," *The Journal of Finance*, vol. 52, no. 1, pp. 57–82, 1997.

[180] R. Roy and S. Shijin, "A six-factor asset pricing model," *Borsa Istanbul Review*, vol. 18, no. 3, pp. 205 – 217, 2018.

[181] L. P. Hansen, "Large sample properties of generalized method of moments estimators," *Econometrica: Journal of the Econometric Society*, pp. 1029–1054, 1982.

[182] J. Y. Campbell and J. H. Cochrane, "By force of habit: A consumption-based explanation of aggregate stock market behavior," *Journal of Political Economy*, vol. 107, no. 2, pp. 205–251, 1999.

[183] R. Bansal and A. Yaron, "Risks for the long run: A potential resolution of asset pricing puzzles," *The Journal of Finance*, vol. 59, no. 4, pp. 1481–1509, 2004.

[184] Z. He and A. Krishnamurthy, "Intermediary asset pricing," *American Economic Review*, vol. 103, no. 2, pp. 732–70, 2013.

[185] M. Levy and G. Kaplanski, "Portfolio selection in a two-regime world," *European Journal of Operational Research*, vol. 242, no. 2, pp. 514–524, 2015.

[186] J. Glova, "Time-varying capm and its applicability in cost of equity determination," *Procedia Economics and Finance*, vol. 32, pp. 60–67, 2015.

[187] T. Bollerslev, R. F. Engle, and J. M. Wooldridge, "A capital asset pricing model with time-varying covariances," *Journal of Political Economy*, vol. 96, no. 1, pp. 116–131, 1988.

[188] C. R. Harvey and A. Siddique, "Autoregressive conditional skewness," *Journal of Financial and Quantitative Analysis*, vol. 34, no. 4, pp. 465–487, 1999.

[189] K.-i. Kamijo and T. Tanigawa, "Stock price pattern recognition-a recurrent neural network approach," in *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 1990, pp. 215–221.

[190] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks," in *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 1990, pp. 1–6.

[191] R. R. Trippi and D. DeSieno, "Trading equity index futures with a neural network," *Journal of Portfolio Management*, vol. 19, pp. 27–27, 1992.

[192] K.-j. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert Systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.

[193] R. Tsaih, Y. Hsu, and C. C. Lai, "Forecasting s&p 500 stock index futures with a hybrid ai system," *Decision Support Systems*, vol. 23, no. 2, pp. 161–174, 1998.

[194] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[195] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers and operations research*, vol. 32, no. 10, pp. 2513–2522, 2005.

[196] X. Zhong and D. Enke, "Forecasting daily stock market return using dimensionality reduction," *Expert Systems with Applications*, vol. 67, pp. 126–139, 2017.

[197] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.

[198] M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru, "Integrating metaheuristics and artificial neural networks for improved stock price prediction," *Expert Systems with Applications*, vol. 44, pp. 320–331, 2016.

[199] D. Enke and S. Thawornwong, "The use of data mining and neural networks for forecasting stock market returns," *Expert Systems with applications*, vol. 29, no. 4, pp. 927–940, 2005.

[200] J.-J. Wang, J.-Z. Wang, Z.-G. Zhang, and S.-P. Guo, "Stock index forecasting based on a hybrid model," *Omega*, vol. 40, no. 6, pp. 758–766, 2012.

[201] W.-C. Chiang, D. Enke, T. Wu, and R. Wang, "An adaptive stock index trading decision support system," *Expert Systems with Applications*, vol. 59, pp. 195–207, 2016.

[202] A. Kondratyev, "Learning curve dynamics with artificial neural networks," *Available at Social Science Research Network (SSRN) 3041232*, 2018.

[203] G. Jeong and H. Y. Kim, "Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning," *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019.

[204] S. Kozak, S. Nagel, and S. Santosh, "Shrinking the cross section," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., 2017.

[205] G. Connor and R. A. Korajczyk, "Performance measurement with the arbitrage pricing theory: A new framework for analysis," *Journal of Financial Economics*, vol. 15, no. 3, pp. 373–394, 1986.

[206] J. H. Stock and M. W. Watson, "Macroeconomic forecasting using diffusion indexes," *Journal of Business and Economic Statistics*, vol. 20, no. 2, pp. 147–162, 2002.

[207] J. Bai and S. Ng, "Determining the number of factors in approximate factor models," *Econometrica*, vol. 70, no. 1, pp. 191–221, 2002.

[208] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989.

[209] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[210] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[211] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[212] A. Cunningham. (2014) Apple breaks revenue, iphone, and ipad records in q1 of 2014. Retrieved on March 27, 2018. [Online]. Available: https://arstechnica.com/gadgets/2014/01/apple-breaks-revenue-iphone-and-ipad-records-in-q1-of-2014/

[213] A. Uddin, X. Tao, and D. Yu, "Attention based dynamic graph learning framework for asset pricing," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 2021, pp. 1844–1853.

[214] S. Huang, Y. Hitti, G. Rabusseau, and R. Rabbany, "Laplacian change point detection for dynamic graphs," in *Proceedings of the 26th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2020, pp. 349–358.

[215] R. N. Mantegna, "Hierarchical structure in financial markets," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 11, no. 1, pp. 193–197, 1999.

[216] K. T. Chi, J. Liu, and F. C. Lau, "A network perspective of the stock market," *Journal of Empirical Finance*, vol. 17, no. 4, pp. 659–667, 2010.

[217] A. Namaki, A. Shirazi, R. Raei, and G. Jafari, "Network analysis of a financial market based on genuine correlation and threshold method," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 21-22, pp. 3835–3841, 2011.

[218] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak, "Spectral analysis of signed graphs for clustering, prediction and visualization," in *Proceedings of the International Conference on Data Mining*. SIAM, 2010, pp. 559–570.

[219] J. Gallier, "Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey," *CoRR*, vol. abs/1601.04692, 2016.

[220] S. Jin and R. Zafarani, "The spectral zoo of networks: Embedding and visualizing networks with spectral moments," in *Proceedings of the 26th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2020, p. 1426–1434.

[221] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[222] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[223] S. Ahmed, Z. Bu, and D. Tsvetanov, "Best of the best: a comparison of factor models," *Journal of Financial and Quantitative Analysis*, vol. 54, no. 4, pp. 1713–1758, 2019.

[224] K. Hou, C. Xue, and L. Zhang, "Replicating anomalies," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2019–2133, 2020.

[225] S. Giglio and D. Xiu, "Asset pricing with omitted factors," *Journal of Political Economy*, vol. 129, no. 7, pp. 000–000, 2021.

[226] E. F. Fama and J. D. MacBeth, "Risk, return, and equilibrium: Empirical tests," *Journal of Political Economy*, vol. 81, no. 3, pp. 607–636, 1973.

[227] S. C. Ludvigson and S. Ng, "A factor analysis of bond risk premia," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., 2009.

[228] C. J. Malloy, T. J. Moskowitz, and A. Vissing-Jørgensen, "Long-run stockholder consumption risk and asset returns," *The Journal of Finance*, vol. 64, no. 6, pp. 2427–2479, 2009.

[229] B. Herskovic, B. Kelly, H. Lustig, and S. Van Nieuwerburgh, "Firm volatility in granular networks," *Journal of Political Economy*, vol. 128, no. 11, pp. 4097–4162, 2020.

[230] A. Nobi, S. E. Maeng, G. G. Ha, and J. W. Lee, "Effects of global financial crisis on network structure in a local stock market," *Physica A: Statistical Mechanics and its Applications*, vol. 407, pp. 135–143, 2014.

[231] S. Rönnqvist and P. Sarlin, "Bank networks from text: interrelations, centrality and determinants," *Quantitative Finance*, vol. 15, no. 10, pp. 1619–1635, 2015.

[232] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[233] R. S. Sandhu, T. T. Georgiou, and A. R. Tannenbaum, "Ricci curvature: An economic indicator for market fragility and systemic risk," *Science Advances*, vol. 2, no. 5, p. e1501495, 2016.

[234] M. Puliga, G. Caldarelli, and S. Battiston, "Credit default swaps networks and systemic risk," *Scientific Reports*, vol. 4, no. 1, pp. 1–8, 2014.

[235] T. Biyikoglu, J. Leydold, and P. F. Stadler, *Laplacian eigenvectors of graphs: Perron-Frobenius and Faber-Krahn type theorems*. New York: Springer, 2007.

[236] D. S. Watkins, "Matrix mathematics: Theory, facts, and formulas," *SIAM Review*, vol. 52, no. 4, p. 790, 2010.

[237] L. Zelnik-manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005.

[238] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[239] B. Mohar, "Some applications of laplace eigenvalues of graphs," in *Graph Symmetry*. New York: Springer, 1997, pp. 225–275.

[240] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Proceedings of the Army Science Conference*, vol. 1, 2010.

[241] T. Idé and H. Kashima, "Eigenspace-based anomaly detection in computer systems," in *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2004, pp. 440–449.

[242] L. Pástor and R. F. Stambaugh, "Liquidity risk and expected stock returns," *Journal of Political Economy*, vol. 111, no. 3, pp. 642–685, 2003.

[243] Z. He, B. Kelly, and A. Manela, "Intermediary asset pricing: New evidence from many asset classes," *Journal of Financial Economics*, vol. 126, no. 1, pp. 1–35, 2017.

[244] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[245] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[246] R. Bikbov and M. Chernov, "No-arbitrage macroeconomic determinants of the yield curve," *Journal of Econometrics*, vol. 159, no. 1, pp. 166–182, 2010.

[247] J. Juneja, "Common factors, principal components analysis, and the term structure of interest rates," *International Review of Financial Analysis*, vol. 24, pp. 48–56, 2012.

[248] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.

[249] A. Paszke and et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[250] T. Adrian, R. K. Crump, and E. Moench, "Regression-based estimation of dynamic asset pricing models," *Journal of Financial Economics*, vol. 118, no. 2, pp. 211–244, 2015.

[251] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3104–3112, 2014.

[252] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *Proceedings of the International Conference on Learning Representations*, 2019.

[253] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of Statistical Software*, pp. 1–68, 2010.

[254] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.

[255] P. J. García-Laencina, J.-L. Sancho-Gómez, A. R. Figueiras-Vidal, and M. Verleysen, "K nearest neighbours with mutual information for simultaneous classification and missing data imputation," *Neurocomputing*, vol. 72, no. 7-9, pp. 1483–1493, 2009.

[256] S. Datta, D. Misra, and S. Das, "A feature weighted penalty based dissimilarity measure for k-nearest neighbor classification with missing features," *Pattern Recognition Letters*, vol. 80, pp. 231–237, 2016.

[257] T. W. Anderson, "An introduction to multivariate statistical analysis," Wiley New York, Tech. Rep., 1962.

[258] C. F. Van Loan and G. H. Golub, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1983.

[259] K. Cho and N. Reyhani, "An iterative algorithm for singular value decomposition on noisy incomplete matrices," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–6.