ABSTRACT

HYBRID DEEP NEURAL NETWORKS FOR MINING HETEROGENEOUS DATA

by
Xiurui Hou

In the era of big data, the rapidly growing flood of data represents an immense opportunity. New computational methods are desired to fully leverage the potential that exists within massive structured and unstructured data. However, decision-makers are often confronted with multiple diverse heterogeneous data sources. The heterogeneity includes different data types, different granularities, and different dimensions, posing a fundamental challenge in many applications. This dissertation focuses on designing hybrid deep neural networks for modeling various kinds of data heterogeneity.

The first part of this dissertation concerns modeling diverse data types, the first kind of data heterogeneity. Specifically, image data and heterogeneous meta data are modeled. Detecting Copy Number Variations (CNVs) in genetic studies is used as a motivating example. A CNN-DNN blended neural network is proposed to authenticate CNV calls made by current state-of-art CNV detection algorithms. It utilizes hybrid deep neural networks to leverage both scatter plot image signal and heterogeneous numerical meta data for improving CNV calling and review efficiency.

The second part of this dissertation deals with data of various frequencies or scales in time series data analysis, the second kind of data heterogeneity. The stock return forecasting problem in the finance field is used as a motivating example. A hybrid framework of Long-Short Term Memory and Deep Neural Network (LSTM-DNN) is developed to enrich the time-series forecasting task with static fundamental information. The application of the proposed framework is not limited to the stock return forecasting problem, but any time-series based prediction tasks.

The third part of this dissertation makes an extension of LSTM-DNN framework to account for both temporal and spatial dependency among variables, common in many applications. For example, it is known that stock prices of relevant firms tend to fluctuate together. Such coherent price changes among relevant stocks are referred to a spatial dependency. In this part, Variational Auto Encoder (VAE) is first utilized to recover the latent graphical dependency structure among variables. Then a hybrid deep neural network of Graph Convolutional Network and Long-Short Term Memory network (GCN-LSTM) is developed to model both the graph structured spatial dependency and temporal dependency of variables at different scales.

Extensive experiments are conducted to demonstrate the effectiveness of the proposed neural networks with application to solve three representative real-world problems. Additionally, the proposed frameworks can also be applied to other areas filled with similar heterogeneous inputs.

HYBRID DEEP NEURAL NETWORKS FOR MINING HETEROGENEOUS DATA

by
Xiurui Hou

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 Computer Science

Department of Computer Science

August 2020

APPROVAL PAGE

HYBRID DEEP NEURAL NETWORKS FOR MINING HETEROGENEOUS DATA

Xiurui Hou

Dr. Zhi Wei, Dissertation Advisor Professor of Computer Science, NJIT	Date
Dr. James M. Calvin, Committee Member Professor of Computer Science, NJIT	Date
Dr. Senjuti Basu Roy, Committee Member Assistant Professor of Computer Science, NJIT	Date
Dr. Antai Wang, Committee Member Associate Professor of Mathematical Science, NJIT	Date
Dr. Wenge Guo, Committee Member Associate Professor of Department of Mathematical Sciences, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Xiurui Hou
Degree:	Doctor of Philosophy
Date:	August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Bachelor of Engineering in Internet of Things,
  Central South University, Changsha, P. R. China, 2015

Major: Computer Science

Presentations and Publications:

Xiurui Hou, Kai Wang, and Zhi Wei, "ST-Trader: A Spatial-Temporal Deep Neural Network for Modeling Stock Market Movement," *IEEE/CAA Journal of Automatica Sinica*, Under Review.

Xiurui Hou, Joseph T. Glessner, Jie Zhang, Munir Khan, Michael March, Hakon Hakonarson, and Zhi Wei, "DeepCNV: A Deep Learning Approach for Authenticating Copy Number Variations," *Nucleic Acids Research*, Under Review.

Fei Tan, Tian Tian, Xiurui Hou, Xiang Yu , Lei Gu, Fernanda Mafra, Brian Gregory, Hakon Hakonarson, and Zhi Wei, "Elucidation of DNA Methylation on N6-Adenine with Deep Learning," *Nature Machine Intelligence*, Under Review.

Xiurui Hou, Kai Wang, Jie Zhang, and Zhi Wei, "An Enriched Time-Series Forecasting Framework for Long-Short Portfolio Strategy," *IEEE Access*, 8(2020), 31992-32002

Chaoran Cheng, Fei Tan, Xiurui Hou, and Zhi Wei, "Success Prediction on Crowdfunding with Multimodal Deep Learning," *IJCAI'19 (Proceedings of the 28th International Joint Conference on Artificial Intelligence)*, Macao, China, August 2019

Fei Tan, Xiurui Hou, Jie Zhang, Zhi Wei, Zhenyu Yan, and Shiquan Weng, "A Novel Risk Assessment Scheme and Practice for Peer-to-Peer Lending," *SIGKDD Workshop Data Science in Fintech (SIGKDD'18)*, London, UK, August 2018

Fei Tan, Xiurui Hou, Jie Zhang, Zhi Wei, and Zhenyu Yan, "A Deep Learning Approach to Competing Risks Representation in Peer-to-Peer Lending," *IEEE Transactions on Neural Networks and Learning Systems*, 30(5), 1565-1574

To My Beloved Parents (Zhaoju Huang & Daigui Xia),
Sister (Linrong Xia), and Kai Wang.

ACKNOWLEDGMENT

First and foremost, I wish to take this opportunity to express my heartfelt appreciation to my advisor Dr. Zhi Wei. It has been his generous help and sustained encouragement that gave me the courage to overcome the difficulties in the road of pursuing my dream and hunting for the scientific truth. His tremendous efforts, invaluable guidance, and infinite patience enable me to bring this dissertation to its culmination. I will always be indebted to Dr. Wei for generously encouraging and supporting me in this critical stage of my life.

Second, I am extremely grateful to Dr. James Calvin, Dr. Senjuti Basu Roy, Dr. Antai Wang and Dr. Wenge Guo for serving on my committee. They have provided me with academic advice inside and outside my research field. This dissertation would not have been possible without their invaluable guidance and generous help. In addition, I wish to thank Dr. Reza Curtmola, Dr. Ali Mili, Dr. David Nassimi, and Dr. George Olsen for supporting and helping me all the time.

I would also like to thank my fellow graduate students and lab mates for their assistance and support. And I appreciate the Teaching Assistant support given by computer science department of NJIT.

I truly appreciate my family for being my emotional anchor. I thank my parents for their endless love, support, and encouragement. Last, but not the least, I am thankful to all who have helped me directly or indirectly for the past five years. It is their support and encouragement that give me the courage and strength to pursue my PhD degree abroad.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

The value of data has been broadly recognized and emphasized nowadays. More and more decisions are made based on the data and analysis rather than solely on experience and intuition. Airlines make arrangement according to the seasonal traffic forecasting; News media, e.g., the Economist and the New York Times, have benefited from the value of Big Data to guide decisions, trim costs and lift scales; Investment companies or hedge funds are seeking trading patterns from historical charts using machine learning algorithms to make profit; UPS analyzes truck delivery times and traffic patterns to optimize routing.

With the fast development of networking, data storage, and the data collection capacity, have increased in a dramatic scale in industry, science and engineering domains, which brings both great opportunities and challenges [74]. As reported, 2.5 quintillion bytes of data are created daily and 90% of the data in the world today are produced within the past few years [74]. A report from McKinsey Global Institute points out that it would need 140,000 to 190,000 more employees with deep analytical expertise and 1.5 million more data-literate managers [46]. At the same time, new computational methods are in demand to process, analyze and understand these datasets.

However, the data collected from the real world is much noisier than the synthetic data used in developing machine learning algorithms due to a couple of reasons, two of which are: 1) there is not a standard protocol about data collection among industries. Different companies value different data for their specific business goals. While the research in academia can only have access to very limited dataset, where they develop algorithms on. So the proposed algorithms based on ideally clean datasets are not transferable to the industry; 2) the inherent heterogeneity of data is universal. For example, since some doctors diagnose diseases from medical imaging pictures, a natural thinking is to apply the image classification algorithms to predict those medical pictures, which is believed to be helpful for doctors. But the meta information about the patients are critical as well, e.g., age, blood pressure, and so on. While the existing image classification algorithms do not accept heterogeneous data inputs, which results insufficient data usage.

The gap between chaotic real-world data and neat academia algorithms needs extra effort to build the bridge. This dissertation focuses on the development of hybrid neural networks for utilizing heterogeneous data input, with application to industry problems (stock market forecasting) and basic science research (bioinformatics).

First, a CNN-DNN model is proposed to accommodate both the medical imaging picture and numerical meta data to make prediction. Through effectively leveraging auxiliary metadata, it can accurately authenticate the CNV calls from other detection algorithms. It achieves comparable performance when compared to human experts' labelling, which demonstrates the promising ability to replace the experts' labor in filtering out the false positive calls.

Second, a LSTM-DNN network is developed for trading long-short portfolio in stock market. Accurate stock return forecasting and clear elucidation of the underlying causal factors are of interest to various parties. As known, the stock price varies along the time, which is typically a time-series forecasting problem. However, the value-investors believes the fundamental information (cross-sectional) about the firms are more noteworthy than price trends. People have to make a trade-off between time-series analysis (so called Technical Analysis (TA)) and fundamental analysis (FA). Such embarrassment does not only exist in financial area, but also in any other areas that are typically time-series tasks but with potential helpful signals in different scales.

Third, a novel VAE-GCN-LSTM pipeline is proposed to extract latent interaction among companies and integrate it into time-series forecasting tasks. There are many reasons that stock prices trend to move together. First, exchange-traded funds (ETFs), such as S&P500 and NASDAQ, track the prices of a basket of stocks. When people trade those funds, all the underlying stocks are traded simultaneously, which causes common fluctuations of those stock prices (see [1]). Second, most professional portfolio managers are specialized in a couple of strategies and stratefy often involves a similar set of stocks. Third, some companies have cooperational relationship, like Apple and Nvidia. If one of them has good or bad news, the effect on the other one could be reflected on the stock price. Thus, taking such interaction into consideration will benefit the stock price prediction tasks. Although the interaction among companies is not difficult to observe, it is not easy to have it cooperated in stock prices forecasting tasks due to three reasons: 1) there are too many fundamental variables to select from (usually the total is more than 1,000). Extra financial expertise is needed to filter out the key variables; 2) although the key variables are selected, the interaction imposed by those variables is not trivial to model due to nonlinearity and chaos; 3) the way that interaction contributes to the final forecasting goal is the biggest obstacle to utilize the fundamental information because such static information has different frequencies and scales from time-varying price variable.

This dissertation is organized in the following manner. Chapter 2 discusses the background and related work of the CNVs detection, financial stock market forecasting, together with the graph convolutional network in other areas. Chapter 3 introduces the proposed hybrid neural network designed for utilizing both image signal and meta data information to authenticating CNV calls from other detection tools. Chapter 4 proposes an enriched time-series forecasting framework with static firm fundamental information. Chapter 5 develops a novel pipeline for discovering potential spatial dependency and integrating spatial dependency and temporal dependency into stock market forecasting tasks.

CHAPTER 2

RELATED WORK

Convolutional Neural Networks (CNNs) [41] are designed for image recognition; AutoRegression is designed for modeling time-series patterns; Bidirectional Encoder Representations from Transformers (BERT) [17] is a technique for Natural Language Processing (NLP). However, the reality is much complex than the academia research scenario. People never make decisions sorely on single data source while multiple data sources means the necessity of aggregation of heterogeneous data types. [32] proposes an adaptive design that fuses a one-dimensional convolutional neural network and a support vector machine to predict water leakage event. It considers both the spatial location of water systems and other oddments of related information to achieve a more comprehensive prediction. But the two separate models may miss the latent interation between the two kinds of signals. To improve the performance on scene recognition and visual domain adaptation, [75] proposes to combine the feature map extracted by CNNs and traditional dictionary-based features, e.g., BoW and spatial pyramid matching. Via such combination, this model can utilize both the local discriminative and structural information. In terms of utilizing image data and non-image data, [26] transforms image data to semantic labels to be fused with non-image data at the semantic level for multi-sensor fusion problem. But transforming image to semantic labels not only needs extra effort and extra expertise but also limits the representation of image signal, which can be improved using CNNs. [48] develops computerized algorithms for high dimensional data and image analysis for predicting disease outcome from multiple modalities including MRI, digital pathology, and protein expression, which quantitatively integrate prognostic information from multiple data sources and modalities. Those hybrid frameworks inspired us to utilize as much as possible information we have to make the prediction for practical problems.

Different data scales commonly exist due to different sampling or different time-intervals data generation. In finance area, stock prices can be accessible by millisecond while the fiscal report of a firm usually comes out quarterly or half-yearly. When making decisions on investment, people would like to gather as much information as possible to draw conclusions. [9] proposes a multi-scale LSTM to process different scale time-series data. But the way to integrate different scale time-series data in this paper is to calculate the average with different window size to make the sequence data to be of the same length, which is not a real modification on the model but a preprocessing strategy. No matter the traditional models or the advanced machine learning methods, there is not a decent framework to accommodate different scale data. In finance area, researchers or investors either conduct AutoRegression on stock prices data to capture the desired time-varying pattern, or aggregate the price data, like calculating moving-average price [18], trying to cooperate the static firm characteristics in prediction tasks. Padding or partition the unequal interval time-series can be found in [67, 47]. Thus, there always is a trade-off between preserving time-series flavor and integrating more information with a different sampling interval.

Research on different assessment data is also extensively conducted because observing events from different dimensions can provide comprehensive information. For example, the package delivery time for a home depends on several factors. The most straightforward is the distance between this home and the post office or the order in the scheduled delivery list, which can be estimated from massive historical delivery time to this home or the neighborhood using time-series analysis (temporal assessment). However, time-series analysis might not be sufficient because the usual pattern can be interrupted by the time-commitment packages, which has a higher priority to be delivered before or at a specific time point (spatial assessment). Thus, a more comprehensive thinking is to utilize both the usual pattern and the package type to be delivered to the target home or its neighborhood to estimate the delivery time. To model video-based emotion, [20] proposes a hybrid network that combines recurrent neural network (RNN) and 3D convolutional network (C3D). Specifically, RNN takes appearance features extracted by convolutional neural network (CNN) over individual video frames as input and encodes motion later, while C3D models appearance and motion of video simultaneously. By assuming that the wind speed possesses time-series seasonality and the wind farms located nearby will experience similar wind speed, [35] introduces a hybrid model of Graph Convolutional Networks (GCNs) and Long-Short Term Memory (LSTM) to capture both the temporal and spatial dependency to forecasting the wind speed for each wind farm.

CHAPTER 3

BACKGROUND

3.1 Copy Number Variations Detection

Copy Number Variation (CNV), a type of structural variations of the human genome with possible association with complex diseases, such as schizophrenia [13] and osteoporosis [76], has received much attention in the past [58, 15, 28]. Detection of CNVs has become routine in genetic studies and in cancer research of disease susceptibility [10]. Using next-generation sequencing (NGS) to profile whole genomes, whole exomes, or targeted regions has been currently widely adopted. NGS has emerged as a technology with the capability to detect both SNVs and structural variants [5, 55] in a single assay. However, CNV analysis via NGS is not trivial. Reliable CNV calls from NGS data demand high sequencing depth and uniformity of coverage across all target sites, which may not be achievable in a cost- and time-effective manner. Consequently, microarray-based detection, as a cost-effective technique, remains a commonly ordered clinical genetic test and is expected to remain as an important CNV detection strategy for years to come [51]. Microarray-based computational calling algorithms, like PennCNV [71] and QuantiSNP [11], have been widely employed for detecting CNVs. Nonetheless, the quantity and quality of CNVs reported by these methods on different platforms exhibit substantial variance. In particular, it is not uncommon to witness unacceptably high false positive rates of CNV calls. Winchester et al. [72] conducted comprehensive experiments to compare a number of algorithms, including Birdsuite 1.5.5 [40], CNAT (Genome Console 3.0.2) [72], GADA (R 0.7-5) [59], PennCNV [71] and QuantiSNP [11]. Although PennCNV demonstrated the best performance in terms of prediction accuracy, only 49% of the detected events could be confirmed [72, 36]. The high false positive rate may be minimized by only reporting the overlapped events from multiple algorithms [58], but at the expense of low sensitivity.

Compared with other methods, PennCNV incorporates multiple sources of information, including total signal intensity and allelic intensity ratio at each SNP marker, the distance between neighboring SNPs, the allele frequency of SNPs, and the pedigree information. It enables kilobase-resolution detection of small-scale CNVs, which are common in the human genome [69, 12, 22]. In contrast, the resolution of CNV detection in previous experimental studies [31, 30, 61, 73] was limited to tens or hundreds of kilobases [71]. Therefore, PennCNV has become a popular tool for CNV calling in clinic analysis. However, to overcome its issue of high false positive rate, it is often recommended to visually examine CNV calls to judge whether they are reliable or not. For example, human experts can visualize and examine LRR (Log R Ratio) and BAF (B Allele Frequency) scatter plot images using the Illumina BeadStudio software or the Affymetrix genotyping console. This additional manual screening step is laborious. Human experts may also be subjective and exhibit certain variance, which is not desired in a clinical setting.

3.2 Asset Pricing and Long-short Portfolio Strategy in Stock Market

Accurate stock return forecasting and clear elucidation of the underlying causal factors are of interest to various parties. However, it is still a notoriously challenging issue owing to the complex, dynamic, and chaotic nature of the stock market. Despite the challenges, many empirical studies have shown that financial markets are predictable to some extent [2, 37, 57].

In the past decades, we have witnessed a dramatic increase in research on stock return prediction. Technical Analysis (TA) and Fundamental Analysis (FA) are two popular methods in the literature of stock return forecasting. TA focuses on modeling time-series price data and predicts future stock prices primarily based on historical price trends. The development of the TA method is mainly driven by data mining and machine learning fields. Various statistical and machine learning methods have been explored and applied, such as Auto-Regression, Neural Networks, Support Vector Machine, and Random Forest. However, the complex and non-stationary property of stock returns imposes a formidable challenge for most time-series modeling approaches. In contrast, FA is a classical topic in finance, especially in asset pricing, and is generally considered for developing long-term investment strategies. FA pays more attention to fundamental factors and variables that are extracted from the firms' annual reports and other announcements. Both TA and FA have their unique advantages on stock return forecasting.

Recently, several attempts were carried out to integrate TA and FA. They propose to first aggregate the time-series sequence into TA indicators (e.g., Moving Averages, Trend Lines, and Relative Strength Index) and then train the predictive models based on both TA indicators and FA variables. It is not difficult to prove that the temporal dependency extracted shrinks into the ad-hoc TA indicators. Integrating both types of analyses while preserving the superiority of the time-series model remains a challenging task due to three main reasons: Firstly, the types of data to deal with are quite different. TA usually deals with time-series data, while FA focuses on annual or quarterly fundamental data. Secondly, TA and FA use different analyzing tools. Classical TA models use moving average and auto-regression, while sorting stocks based on firm characteristics into portfolios is the conventional method in FA. Finally, they differ in their trading strategies. TA attempts to find better trading signals on one or two assets and simply applies a pair trading or buy-and-hold strategy. In contrast, trading strategies developed by FA require a long-short portfolio, which longs the stocks with higher expected returns and shorts the stocks with lower expected returns.

Intuitively, the long-short portfolio strategy can achieve its best performance when we accurately locate a stock's expected return on the entire cross-section of the expected returns, not necessarily its exact expected return. Thus, rather than using raw data, we apply a cross-section-aware preprocessing on features and feed them to our LSTM-DNN framework. Specifically, we standardize all individual stock features within each month by subtracting the cross-section average value of a feature, except for categorical features, and dividing the resulting value by the cross-section standard deviation of the given feature. For example, if the lag 1-month return of a stock is 1% while the average stock return in that month is 2%, and the cross-section standard deviation is 2%, the lag 1-month return of the given stock after cross-section preprocessing is $(1\% - 2\%)/2\% = -0.5$. Via such a preprocessing, the factors encompass richer information in the sense that it contains the relative position information of the firm characteristics across all firms within each month. It is also important to note that all factors we use are available at the time of portfolio construction.

The long-short portfolio strategy works as follows. It ranks stocks based on their individual predicted returns. Then, given their ranks, we buy the top decile[1] and sell short the bottom decile of securities once every rebalancing period. One advantage of the long-short portfolio strategy is that the movement of the whole market has little impact on the performance of this strategy. For example, a hedge fund might sell short \$1 million of Apple stock and buy \$1 million of Microsoft stock. With this position, any event that causes all tech industry stocks to fall will make a profit from the Apple position and bear a matching loss on the Microsoft position. By exploiting a long-short portfolio, namely, invest the stocks with the highest predicted returns and sell short the stocks with the lowest predicted returns, an investor can profit from inferring the relative performance of a stock in the future period.

3.3 Convolutional Neural Networks in Graph-Structure Data

Recently, applying convolutional neural networks (CNNs) to graphs with arbitrary structures has caught people's attention. Two main directions are being explored

[1]Each part represents 1/10 of the sample or population.

in the literature: 1) the definition of spatial convolution is generalized in [50, 53]; 2) generalizing CNNs to 3-dimensional data as a multiplication in graph Fourier domain is discussed in [4, 16, 50] by the way of convolution theorem. Using geodesic polar coordinates, the authors define the convolution operation on meshes. Therefore, this method is suitable for manifolds and cannot be directly applied to graphs with arbitrary structures. The spatial approach proposed in [53] has more potential possibilities in generalizing CNNs to arbitrary graphs. It has three steps: first, select a target node; second, construct the neighborhood of target node; third, normalize the selected sub-graph by ordering the neighbors. The normalized sub-graphs are then fed into 1-dimension Euclidean CNN. Since there is not a natural ordering property in graphs, either temporal or spatial, it has to be imposed by a labeling procedure. The spectral framework to solve this issue is first introduced by [4]. The main disadvantage of this method is of high computational complexity, $\mathcal{O}(n^2)$, which is overcome by the method proposed in [16], which provides strictly localized filters with a linear complexity $\mathcal{O}(|\mathcal{E}|)$. The first order approximation of the proposed spectral filter is adopted by [39] in a semi-supervised node classification task successfully. Thus, we also use the spectral filters introduced in [16] in our framework because of the efficiency and denote the convolution operation as $*_{\mathcal{G}}$.

Applying graph convolution operation in time-series tasks is demonstrated to be helpful in some studies. [77] develop a spatio-temporal GCN model for traffic forecasting. The conventional method for traffic forecasting is to do time-series analysis for each traffic entity, like a specific highway or a city road. The more natural thought is that if two roads are close, they have high probability to experience the same volume of traffic. In their study, the GCN model can perform convolution operation with much faster training speed with fewer parameters than the traditional CNN model that is more suitable for grid structured data, e.g., images. [44] also proposed a spatio-temporal GCN model for human body skeletons based action recognition and the improvement is significant. [60] used GCN in dynamic texture recognition by extracting low-level features. [33] combined LSTM and Convolutional LSTM for capturing both time sequencing features and map sequencing features. There are some attempts to combine time-series forecasting and graph structured convolution operation, like [35] and [34] forecasting wind speed and solar radiation by enriching the time-series with wind farm distances and solar site distances. The wind farms located nearby are supposed to experience much the same wind speed and direction. Wind forecasting tasks benefit from the geographical information via graph convolutional network.

CHAPTER 4

DEEPCNV: A DEEP LEARNING APPROACH FOR AUTHENTICATING COPY NUMBER VARIATIONS

4.1 Introduction

Copy Number Variations (CNVs) are an important class of variations contributing to the pathogenesis of many disease phenotypes. Detecting CNVs from genomic data remains a difficult problem and even state-of-the-art methods such as PennCNV suffer from an unacceptably high false positive rate. A common practice is to have human experts to manually review original CNV calls for filtering false positives before further downstream analysis or experimental validation. Here we propose DeepCNV, a deep learning-based tool, intended to replace human experts when validating CNV calls, focusing on calls made by PennCNV. The sophistication of the deep neural network algorithm is enriched with over 10,000 expert-scored samples split into training and testing sets. Variant confidence, especially for CNVs, is a main roadblock impeding progress of linking CNVs with disease. We show that DeepCNV adds a dramatic shift to the confidence of CNVs with an optimal AUC of 0.909, far exceeding other machine learning methods. The superiority of DeepCNV was also benchmarked and confirmed using an experimental validation dataset. The improvement made by DeepCNV leads to less false positive results and failures to replicate associations that plague the CNV genome-wide association study community.

4.2 Data Description

4.2.1 Human Labeled Dataset

We profiled 341 unique and independent healthy human samples using SNP microarray (Illumina Omni 2.5). PennCNV was applied with default settings to make CNV calls. As a result, we obtained 10,748 CNVs output from PennCNV.

- **Image Data.** Next, we intended to visually examine CNV calls to judge whether they are true positives or not. We utilized the auxiliary visualization program (visualize_cnv.pl) provided by PennCNV to generate image files for the CNV calls automatically. For each CNV call, it produces one LRR (Log R Ratio) scatter plot image and one BAF (B Allele Frequency) scatter plot image. Human experts then labelled them as either false positive or true positive based on eye-checking.

- **Meta Data.** PennCNV also generates some brief summary statistics for quality checking. The summary statistics consist of 13 features, which we called meta data (see Table 4.1). Both DeepCNV and the competing machine learning methods took meta data as input.

- **Sample Information.** Out of the 10,748 CNVs, 6858 of them were labeled as negative (false positive) samples. Generally, the larger the size of CNV, the easier to detect. Among the 10,748 CNVs, 2505 are small (0-5kb); 3941 are medium (5kb-20kb); and 4302 are large (>20kb). For CNVs of these three sizes, we kept the same positive/negative ratio, and randomly split the data into the training (70%), validation (15%) and testing (15%) datasets as shown in Table 4.2.

Table 4.1 Summary of Meta Data

Index	Feature Name	Data Type	Data Range	Description
1	Call Rate	Float	[0.978, 0.999]	Simple SNP genotyping call rate
2	Length	Float	[31, 6455331]	The CNV length
3	Copy Number	Integer	[0, 4]	The number of copy number
4	Number of SNPs	Integer	[3, 5995]	The number of independent SNP markers
5	PennCNV Confidence	Float	[14.64, 13758.3]	The call confidence of PennCNV
6	Number of CNVs in Sample	Integer	[18, 911]	Inflated numbers of false positive CNV calls
7	LRR_mean	Float	[-0.035, 0.024]	The mean of Log R Ratio
8	LRR_SD	Float	[0.10, 0.27]	The standard deviation of Log R Ratio
9	BAF_mean	Float	[0.498, 0.511]	The mean of B Allele Frequency
10	BAF_SD	Float	[0.029, 0.057]	The standard deviation of B Allele Frequency
11	BAF_DRIFT	Float	[0.000, 0.003]	BAF drift
12	WF	Float	[-0.02, 0.031]	Waviness factor
13	GCWF	Float	[-0.006, 0,009]	The GC base content adjusted waviness factor

Table 4.2 Summary of Human Labeled Samples. 'NEG' and 'POS' Represent the False Positive and True Positive Samples from PennCNV, respectively.

CNV size	Training (70%)		Validation (15%)		Testing(15%)		Total
	POS	NEG	POS	NEG	POS	NEG	
0-5kb (Small)	746	1006	160	216	161	216	2505
5kb-20kb (Medium)	1043	1715	224	367	224	368	3941
>20kb (Large)	932	2079	200	445	200	446	4302
Sub Total	2721	4800	584	1028	585	1030	10748
Total	7512		1612		1615		10748

4.2.2 Experimentally Validated Dataset

We experimentally validated 616 CNV calls using qPCR. 520 samples were confirmed to be true positives, while 96 samples turned out to be false positives. qPCR was performed using TaqPath ProAmp Master Mix (ThermoFisher Scientific). Taqman assays targeting the desired regions were identified using the ThermoFisher Scientific website tools and were selected to be compatible with the hTERT reference Taqman assay. 10 ng of genomic DNA was included in each reaction, along with the indicated Taqman assay and the hTERT reference assay in a reaction volume of 10 ml. Each reaction was run in triplicate. For each assay, three controls were run along with subject samples: a no template control (water alone), and commercial sources of male and female genomic DNA (Promega). PCR was performed on a Viia 7 Real-Time PCR system (ThermoFisher Scientific), using cycling conditions recommended for the TaqPath ProAmp master mix for copy number variant detection (standard cycling conditions: 95°C for 10 minutes to activate the enzyme, followed by 40 cycles of 95°C for 15 seconds and 60°C for 1 minute). Data were exported to text file using the QuantStudio Real-Time PCR Software v1.2 (ThermoFisher Scientific) and imported to Copy Caller v2.1 for analysis (ThermoFisher Scientific). Analysis of each Taqman assay was performed in Copy Caller using the commercial male DNA as the calibrator sample. Normal copy number of the commercial female DNA was confirmed as a control, as was failure of amplification in the no template control sample.

4.3 Methods

To fully explore the available information, we constructed a blended deep neural network with two branches, a deep convolutional neural network [43] (CNN) and a deep fully connected neural network (DNN), to simultaneously model the image data and meta data. We present the network architecture in Figure 4.1. CNN is one of the most popular deep learning architectures and has demonstrated the state-of-the-art performance for visual applications [42, 41, 66]. The CNN part of our DeepCNV model is composed of a stack of convolutional layers, where we use filters with a receptive field of 3×3. The convolution stride is fixed to 1 pixel. All convolutional layers are equipped with LeakyReLU activations [27]. Max-pooling is performed over a 2×2 window, with stride 2. Mathematically, each convolutional layer computes

$$G(X)_{f,i,j} = \sum_{c=0}^{C-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} W^{f}_{m,n,c} X_{i+m,j+n,x}, \tag{4.1}$$

, where X is the input image or the output from the previous layer; f is the index of convolutional filter kernel and (i, j) is the index of output position. Filter W^f is the $M \times N \times C$ weight matrix with M and N being the width and length of the filter and C being the input channel dimension. Specifically, (M, N, C) is $(3, 3, 3)$ for all convolutional layers. LeakyReLU [27] represents leaky rectified linear unit, which is defined as follows:

$$LeakyReLU(x) = \begin{cases} x, & x \geqslant 0 \\ \alpha, & x < 0 \end{cases} \tag{4.2}$$

where α is a parameter. If α is 0, LeakyReLU is equivalent to ReLU [52]. The parameter α of LeakyReLU is tuned on training data and set to be 0.001 for all activation layers. In addition, we apply the dropout regularization [64], with a dropout probability 0.4, to prevent overfitting. As shown in Figure 4.1, there are five convolutional layers with number of filters (32, 64, 64, 128, 128). The input to the CNN branch is a fixed-size 900×900 RGB image.

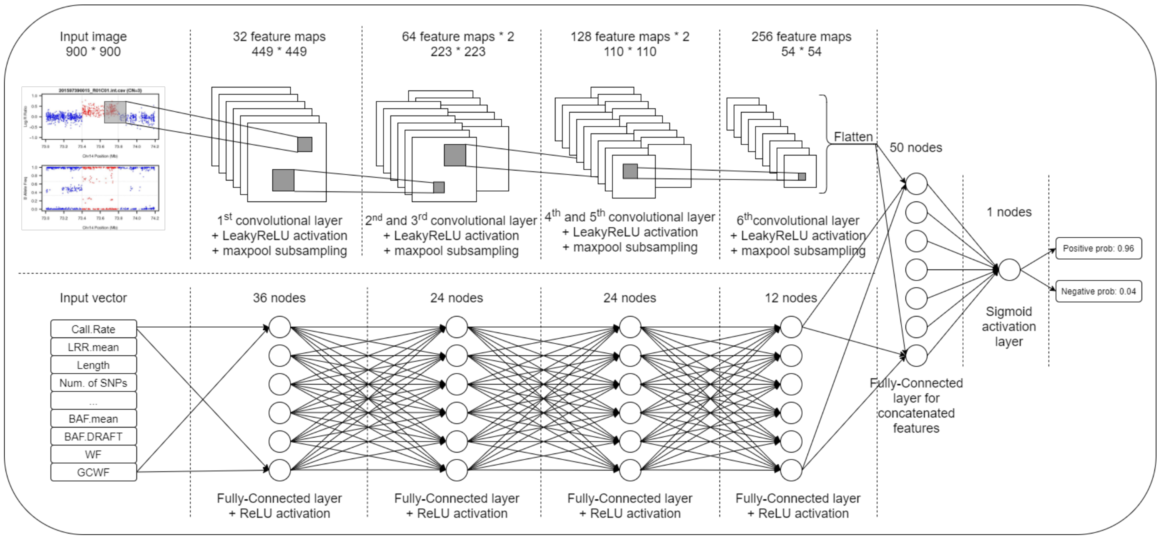
Input image 900 * 900
32 feature maps 449 * 449
64 feature maps * 2 223 * 223
128 feature maps * 2 110 * 110
256 feature maps 54 * 54
Flatten
50 nodes
1 nodes
1st convolutional layer + LeakyReLU activation + maxpool subsampling
2nd and 3rd convolutional layer + LeakyReLU activation + maxpool subsampling
4th and 5th convolutional layer + LeakyReLU activation + maxpool subsampling
6thconvolutional layer + LeakyReLU activation + maxpool subsampling
Positive prob: 0.96
Negative prob: 0.04
Input vector
Call.Rate
LRR.mean
Length
Num. of SNPs
...
BAF.mean
BAF.DRAFT
WF
GCWF
36 nodes
24 nodes
24 nodes
12 nodes
Sigmoid activation layer
Fully-Connected layer for concatenated features
Fully-Connected layer + ReLU activation
Fully-Connected layer + ReLU activation
Fully-Connected layer + ReLU activation
Fully-Connected layer + ReLU activation

Figure 4.1 The architecture of DeepCNV model. The upper part is the Convolutional Neural Network (CNN) for modeling the image data; The lower part is the Deep Fully Connected Neural Network (DNN) for modeling the meta data.

The second branch is a pure fully connected neural network to learn how the meta data contribute to the final decision. This network accepts a numerical input vector with length equal to 13, and then followed by four layers with (36, 24, 24, 12) neurons, respectively. The two branches are flattened, concatenated and fed into a dense layer with 50 neurons, followed by the final sigmoid activation node to generate the score for classifying false positive and true positive samples. The optimizer is RMSprop [68] with learning rate $1e^{-4}$.

We show that how a CNV image looks like in the eye of DeepCNV model when it "annotates" CNV calls. To this end, we employ a technique called Grad-CAM [62] to visualize the regions that are important for prediction. The class-specific gradient information flowing into the final convolutional layer of the CNN has been used to generate a coarse localization map of the important regions of the input image. We show the pipeline in Figure 4.2. In order to obtain the class-discriminative localization map, we first compute the gradient of the score for class c, with respect to feature maps A^n of a convolutional layer. The gradients flowing back are averaged to obtain the neuron importance weight w_n^c:

$$w_n^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^n} \tag{4.3}$$

where A_{ij}^n refers to the activation at location (i, j) of feature map A^n and $Z = \sum_i \sum_j 1$. The weight w_n^c represents the importance of feature map n for a target class c. We perform a weighted combination of forward activation maps and apply an additional ReLU [52] operation, as we are interested in the features that have positive impacts on the class of interest. As shown in Figure 4.2, we resize the importance score of each pixel to [0,1] and highlight the regions that are most valuable to the current prediction.

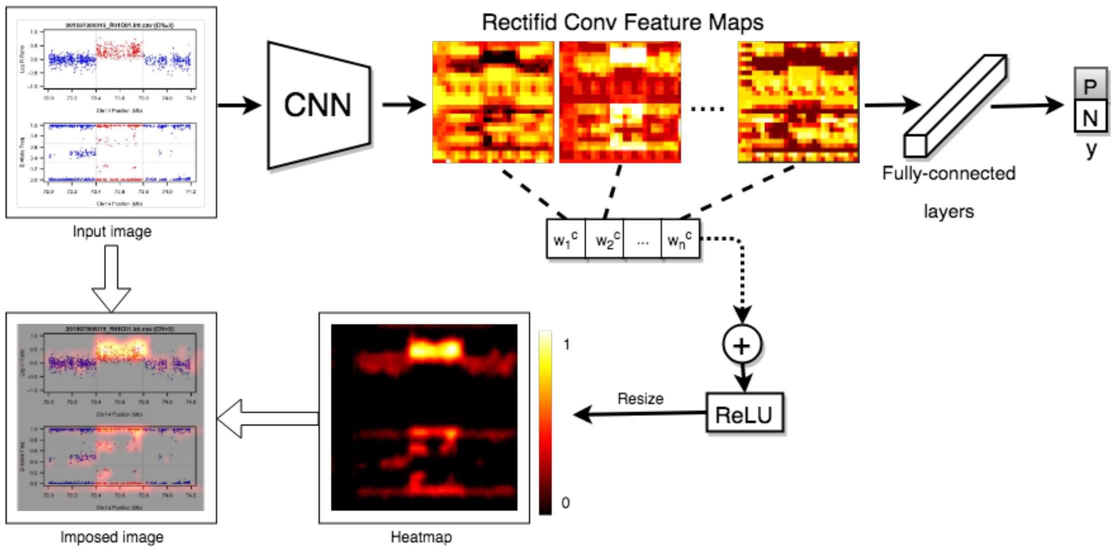

Figure 4.2 The Grad-CAM pipeline for detecting important regions of the image data.

4.4 Experiment

We trained DeepCNV using the training set, tuned its hyperparameters using the validation set, and evaluated it using the test set. DeepCNV consists of two main branches, CNN for image data and DNN for meta data. To demonstrate the contribution of each branch and the utility of the image data and meta data, we also evaluated its two reduced versions, CNN and DNN. The CNN only keeps the CNN branch and takes only image data as input, while DNN employs only the DNN branch with only meta data as input. We compared DeepCNV with DNN, CNN and other popular machine learning methods including Support Vector Machine (SVM) [65], Random Forest (RF) [45] and Logistic Regression (LR). SVM, RF and LR were implemented using scikit-learn library with recommended hyperparameters [56]. Note that SVM, RF, LR and DNN take meta data as input, while the input for CNN is the image data.

4.4.1 Results on Human Labeled Dataset

Figure 4.3 presents prediction performance of these methods in terms of area under the ROC curve (AUC) on the testing dataset. We make a few remarks about the prediction results. Firstly, DeepCNV achieved the best prediction performance across different length of CNVs with an overall AUC of 0.909, followed by RF and then SVM. Second, the smaller the CNV, the harder the prediction. All performances get better along with the CNV size increasing and DeepCNV brings the largest improvement for small CNVs (< 5kb), which are the most challenging cases. Thirdly, for the large CNVs (>20 Kb), the AUC of DeepCNV reaches 0.958, which is promising and in keeping with clinical performance standards.

Consistency between DeepCNV and Human Not only can DeepCNV provide a binary decision (CNV or not), it can also produce a probability of being a CNV. Based on the predicted CNV probability from low to high (0 to 1), we divide

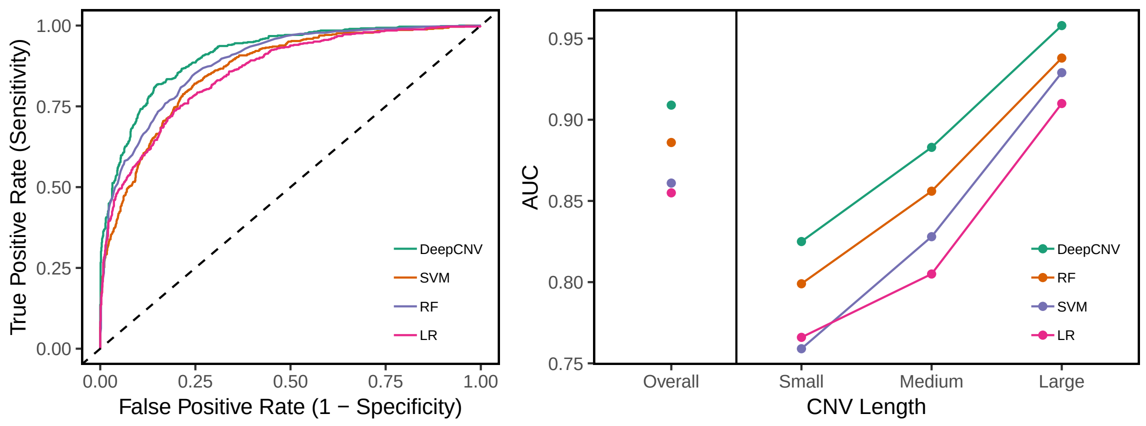

Figure 4.3 Prediction performance on human labeled dataset. The left panel is the ROC curve for each method; The Right panel shows the AUC values for each method on different sizes of CNVs.

the testing samples into ten tiers and summarize the total number of samples in each tier as shown in Figure 4.4. We further stratify the samples based on their copy numbers (CN) estimated by PennCNV. We highlight true positive portions in red. We can see that for all scenarios (CN=0,1 and 3), the higher the predicted probability of DeepCNV, the more likely a CNV call is a true positive. Therefore, the predicted probability can be used for effectively prioritizing candidate calls. For example, a more stringent cutoff may be applied in clinical setting if we want to achieve a high positive predictive value (PPV). It is not supervising that recognizing homozygous deletion (CN=0) is the easiest case: when the probability is higher than 0.5, DeepCNV achieves a 100% consistency with human labeling. When there is one copy deviation, it is interesting to see hemizygous deletion (CN=1) is harder to predict than hemizygous duplication (CN=3). For hemizygous deletion, the predicted probabilities spread over from 0 to 1. Quite a few cases center around probability of 0.5, from which we observe the largest disagreement between DeepCNV and human experts. For hemizygous duplication, in contrast, DeepCNV is more confident with its prediction, with most samples having probability of either less than 0.1 or larger than 0.9, agreeing well with human judgement. For homozygous duplication (CN=4), we have too few samples (less than 10) to draw any conclusion.

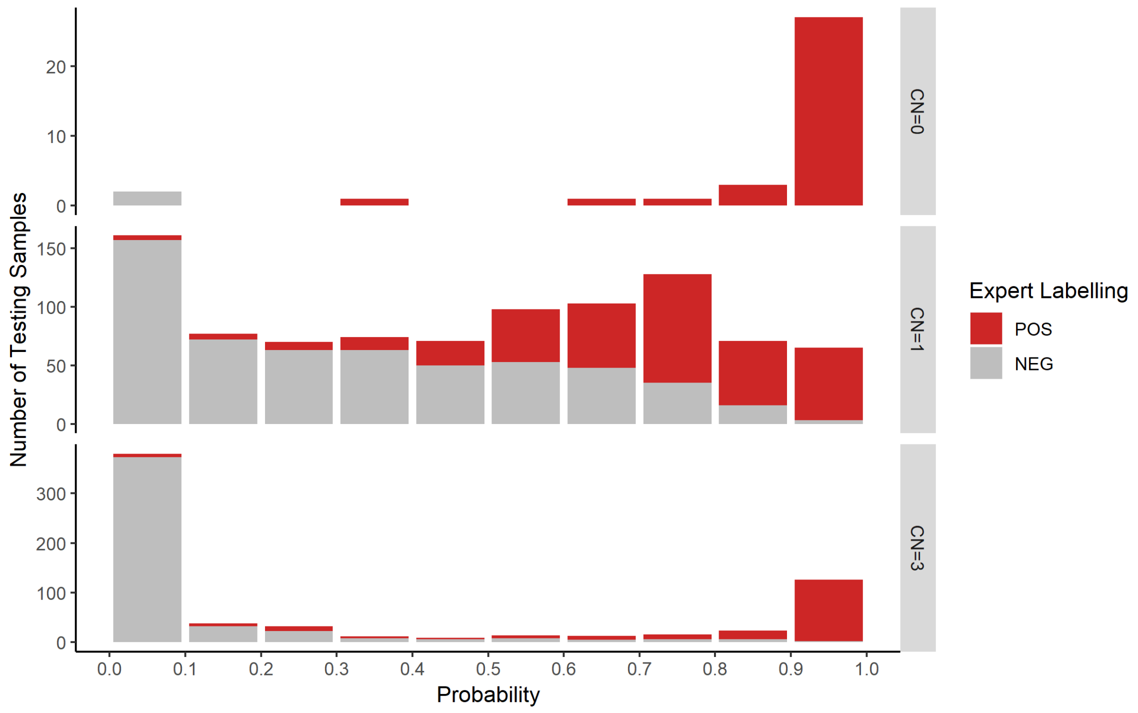

Figure 4.4 Consistency between DeepCNV and human expert labelling in different Copy Number scenarios. The CN (Copy Number) refers to the actual integer copy number estimates calculated by PennCNV, and the normal copy number is 2. For autosome, CN=0 or 1 means there is a deletion and CN$\geqslant$3 means there is a duplication.

4.4.2 Results on Experimentally Validated Dataset

We collected an experimentally validated dataset to benchmark our method in comparison with human experts. DeepCNV model trained on the manually labeled dataset was applied to test on these 616 experimentally validated samples. A human expert was also tested on the same dataset simultaneously. The human expert achieved a positive predictive value (PPV) of 0.924, which was a bit higher than the PPV of 0.893 for DeepCNV. However, DeepCNV obtained a sensitivity of 0.833, much higher than the sensitivity of 0.484 for the human expert. Overall, the DeepCNV yielded a F1 score of 0.862, much larger than the F1 score of 0.635 achieved by the human expert. This testing set has a high true positive rate (520/616=84%). The human expert tends to be stringent and reject a substantial number of cases, based on his prior training and experience. Consequently, during the testing procedure, the human expert may unconsciously reject more cases to meet his prior expectation, which probably explains the inferior performance of human being in this testing experiment.

4.4.3 Ablation Study

Since DeepCNV is a hybrid neural network of CNN and DNN (fully-connect neural network), it is essential to check the contribution of each component. In ablation study, we split DeepCNV into pure CNN (with image data as input) and pure DNN (with meta data as input). We repeat shuffling data, training and testing models 50 times to investigate the performance at a statistical level. Table 4.4.3 presents the mean and standard deviation (in parenthesis) of AUC scores for each component. The degree of freedom is 49 since we repeat 50 times. From Table 4.4.3, DNN with meta data only performs worst but with least standard deviation. It is interesting to observe that DeepCNV achieves significantly higher AUC score than CNN when CNV size is small, which makes sense because it is much harder for human reviewing the small size CNVs and DeepCNV that is of the ability to integrate hybrid data sources can have better performance (The superiority for small CNVs of DeepCNV over CNN is significant at the $P = .05$ level). It suggests that both image data and numerical meta data are useful: each provides some complementary information the other one does not have, while the image data may be more informative than the meta data. The ablation study again demonstrates that it is harder to authenticate small CNVs with visual reviewing and the metadata becomes more indispensable for such situation.

Table 4.3 Mean and Standard Deviation of AUC Scores for each Component with Degree of Freedom = 49

Component	Overall	Small Size	Medium Size	Large Size
CNN (image)	0.905 (0.054)	0.812 (0.047)	0.876 (0.057)	0.958 (0.051)
DNN (meta data)	0.817 (0.003)	0.724 (0.004)	0.789 (0.003)	0.898 (0.003)
DeepCNV (both)	0.909 (0.042)	0.825 (0.032)	0.883 (0.044)	0.958 (0.038)

4.5 Conclusion

Standard approaches for identifying CNVs rely on predictions from the CNV tools (such as PennCNV) followed by expert visual checking. However, manually differentiating the false positives from the true positives becomes less affordable when the sample size is large. DeepCNV is designed to serve the purpose of automating the screening process. Both the CNN and DNN parts of the DeepCNV contribute to the final results. The hierarchical structure of DeepCNV compresses the original image and meta data from intractable large dimension into a feasible and informative representation to increase the distinguishability. Sequencing data can be plotted in the same manner with normalized read depth as LRR and clustered allele depth as BAF. Then the images can be run through DeepCNV similarly. However, a high depth of coverage is required for reliable CNV detection. For WGS, a sequencing depth of 40-50X is recommended while currently most WGS is performed at 15-30X. WES may not have the sequencing depth issue, but its library preparation step introduces bias and noise, which affect the uniformity and consistency of coverage across targets. DeepCNV was applied to recognize false positives output from PennCNV. In principle, it can be coupled with any other CNV callers that are plagued by false positives. We can similarly generate LRR and BAF images and relevant meta data for candidate CNV calls made by other CNV tools, followed by human labelling. DeepCNV then can be trained using the same pipeline. Nevertheless, it may be more desired if the deep learning-based method can be an end-to-end procedure which eliminates the dependence on the PennCNV or other CNV detection programs. In other words, we hope to recognize CNVs directly from the raw data using deep learning. We leave these explorations to future work. Altogether, we can foresee the detection routine would become much more intelligent in the near future.

CHAPTER 5

AN ENRICHED TIME-SERIES FORECASTING FRAMEWORK FOR LONG-SHORT PORTFOLIO STRATEGY

5.1 Introduction

Stock return forecasting typically requires a large number of factors and these factors usually exhibit nonlinear relations with each other. Conventional methods of stock return forecasting mainly fall into two categories: Technical Analysis and Fundamental Analysis. Technical Analysis focuses on time-series data, while Fundamental Analysis explores low-frequency fundamental variables. Although there are substantial works on either time-series analysis or fundamental analysis, few studies have enriched the time-series forecasting with fundamental variables, as the features are characterized by different frequencies, scales and types. In this chapter, we propose a Long Short-Term Memory and Deep Neural Network (LSTM-DNN) hybrid model to integrate the fundamental information into time-series forecasting tasks. We demonstrate how investors can benefit from the superior performance of LSTM-DNN by constructing a long-short portfolio that takes long positions in stocks with the highest forecasting returns and short positions in stocks that are expected to decline.

5.2 Data Overview

The experimental data is obtained from the Center for Research in Security Prices (CRSP). Specifically, the data we downloaded from CRSP includes individual stock returns and prices, S&P 500 index return, industry categories, number of shares outstanding, share code, exchange code, and trading volume. The prices of stocks are recorded at the end of each month and adjusted for stock splits and stock dividends. The stock return of month t is calculated by using the adjusted close price of month t divided by the adjusted close price of month $t-1$ and then minus 1. S&P 500 return data is used to derive market index $past\{6, 24, 60\}ave$ in Table 5.1. Shares outstanding is the number of publicly held shares recorded in thousand. We only study the returns of common shares (with CRSP share code 10 or 11). Daily trading volumes are aggregated to calculate the monthly trading volumes. Standard Industrial Classification Code (SICCD) is used to group companies with similar products or services. In this presentation, we only study stocks listed on the three normal exchanges, NYSE, AMEX, and Nasdaq (with exchange code 1, 2, and 3, respectively).

The firm's book value data is obtained from COMPUSTAT [1] and the one-month Treasury bill rate is obtained from Kenneth French's data library [2].

Book Value Per Share (BKVLPS) in COMPUSTAT is the Common Equity Liquidation Value (CEQL) divided by Common Shares Outstanding (CSHO) in a fiscal year-end.

5.3 Methods

5.3.1 Problem Formulation

We aim to predict the monthly return for each stock based on the historical data. Let subscript t and $t-\tau$ denote the t-th month and lagged τ month, respectively. In addition, we use r and $\hat{r}$ to represent actual stock return and the predicted stock return throughout the paper. We denote the operable stock set at time t as $\boldsymbol{\Omega}_t$. Note that the stocks with missing values in the required features are not included in the set $\boldsymbol{\Omega}_t$. Let r_t^i be the monthly return of the i-th stock at time t, where $i \in \{1, ..., N_t\}$ and $N_t = |\boldsymbol{\Omega}_t|$ represents the number of active stocks at month t. Note that the number of active stocks, N_t, might vary at each time point due to the missing data.

[1]COMPUSTAT is a database held by Standard & Poor's, and we access the COMPUSTAT via Wharton Research Data Service (WRDS).

[2]`http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html` (accessed on 08/01/2019)

For the stock fundamental information, f_t^i represents the latest available annual fiscal report statistics of stock i at time t. Assume that there are M fundamental features in total. We denote by $f_t^{i,j}$, where $j \in \{1, ..., M\}$, the j-th fundamental feature of stock i at time t.

5.3.2 Cross-Section Standardization

We propose to construct the long-short portfolio based on the predicted relative performance of each stock. Instead of using the raw historical stock returns and fundamental information, we normalize the dependent and independent variables across all the active stocks at each month to encode the relative information rather than an absolute term. Let v_t^i be a numerical variable of the i-th stock at time t. We normalize the raw variable v_t^i and make it comparable along the time by removing the impacts of group mean and standard deviation. Specifically, the normalized value $\tilde{v}_t^i$ is calculated as follows:

$$\bar{v}_t = \Sigma_{j=1}^{N_t} v_t^j / N_t \tag{5.1}$$

$$\tilde{v}_t^i = (v_t^i - \bar{v}_t) / \sqrt{\Sigma_{j=1}^{N_t} (v_t^j - \bar{v}_t)^2 / (N_t - 1)} \tag{5.2}$$

Note that the normalization is also consistent with the essence of cross-section stock return prediction, which is to accurately forecast the relative performance of a stock [19].

As shown in Table 5.1, for each stock, we calculate 27 features, including 18 monthly return variables and 9 fundamental variables. Excepting the average market returns and industry category (a classification code to group companies with similar products or services), which are categorical variables, all other variables are cross-section standardized before feeding to various machine learning models. In this paper, we focus on a compact number of features. A larger set of features may improve the overall performance, but at the expense of increased cost to collect the

Table 5.1 Feature Notations and Descriptions

Feature	Description
$\{r^i_{t-1}, ..., r^i_{t-18}\}$	Lagged 1 to lagged 18 monthly return
PRC^i_{t-1}	Lagged 1-month price
VOL^i_{t-1}	Lagged 1-month volume
$SIZE^i_{t-1}$	Lagged 1-month size (price*shares outstanding)
BM^i	The book-to-market ratio in the last fiscal year
BM_pos^i	Indicator of positive book value in the last fiscal year
$SICCD^i$	The industry category
$past6ave$	Average return of the market in past 6 months
$past24ave$	Average return of the market in past 24 months
$past60ave$	Average return of the market in past 60 months

The first feature is for LSTM and the rest are for DNN. Supersript i represents stock i.

required data, especially for inexperienced individual investors. Further works will be exploring more types of features and the impacts of different features.

5.3.3 LSTM-DNN Model

We apply a stacked Long Short-Term Memory Network (LSTM) to model the monthly return time series $\{r^i_{t-1}, ..., r^i_{t-18}\}$. A compact form of the equations for the forward pass of an LSTM unit with a forget gate is represented as follows:

$$f_t = sigmoid\left(W_{f,\tilde{r}}\tilde{r}_t + W_{f,h}h_{t-1} + b_f\right) \tag{5.3}$$

$$i_t = sigmoid(W_{i,\tilde{r}}\tilde{r}_t + W_{i,h}h_{t-1} + b_i) \tag{5.4}$$

$$o_t = sigmoid(W_{o,\tilde{r}}\tilde{r}_t + W_{o,h}h_{t-1} + b_o) \tag{5.5}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{c,\tilde{r}}\tilde{r}_t + W_{c,h}h_{t-1} + b_c) \tag{5.6}$$

$$h_t = o_t \circ tanh(c_t) \tag{5.7}$$

where $\circ$ denotes the element-wise Hadamard product; $\tilde{r}_t \in \mathbb{R}^{\tau}$ is the normalized input vector at timestep t; f_t is the forget gate activation vector; i_t and o_t are input and output gates' activation vectors, respectively, while h_t and c_t represent the hidden state (output vector) and cell state. $W_{*,\tilde{r}} \in \mathbb{R}^{h\times\tau}$, $W_{*,h} \in \mathbb{R}^{h\times h}$ and $b \in \mathbb{R}^h$ are weight matrices and bias vectors. In contrast, the fundamental variables are fed into a fully-connected neural network (FCNN). We concatenate the outputs from both LSTM and FCNN and add additional fully-connected layers to output the final predictions. Mathematically, the output of the hidden neuron in the fully-connected layer is defined as

$$\sigma(\Sigma_{j=1}^{n} w_j x_j + b_j) \tag{5.8}$$

where σ, w_j and b_j represent the activation function, weight matrix and bias, respectively. Following Nair et al. [52], we use rectified linear unit (ReLU) activation function for all hidden neurons. The ReLU activation is defined as follows:

$$\sigma_{ReLU}(\Sigma_{j=1}^{n} w_j x_j + b_j) = max(0, \Sigma_{j=1}^{n} w_j x_j + b_j) \tag{5.9}$$

The loss function we use is the mean squared error (MSE). For illustration, the mean squared error at time t is defined as follows:

$$MSE_t = \Sigma_{i=1}^{N^t} (\tilde{r}_t^i - \hat{\tilde{r}}_t^i)^2 / N^t \tag{5.10}$$

, where $\tilde{r}_t^i$ is the real standardized return of stock i at time t and $\hat{\tilde{r}}_t^i$ is its estimation. Figure 5.1 shows the structure of the proposed LSTM-DNN model together with the number of hidden nodes at each layer. Techniques such as L2 regularization and Dropout [64] are applied to avoid overfitting. Linear activation function is used for the final output layer, while the ReLU activation functions are utilized for other layers. Adam algorithm [38] with learning rate of 0.001, and decay rate of 0.001 is adopted to optimize the model.

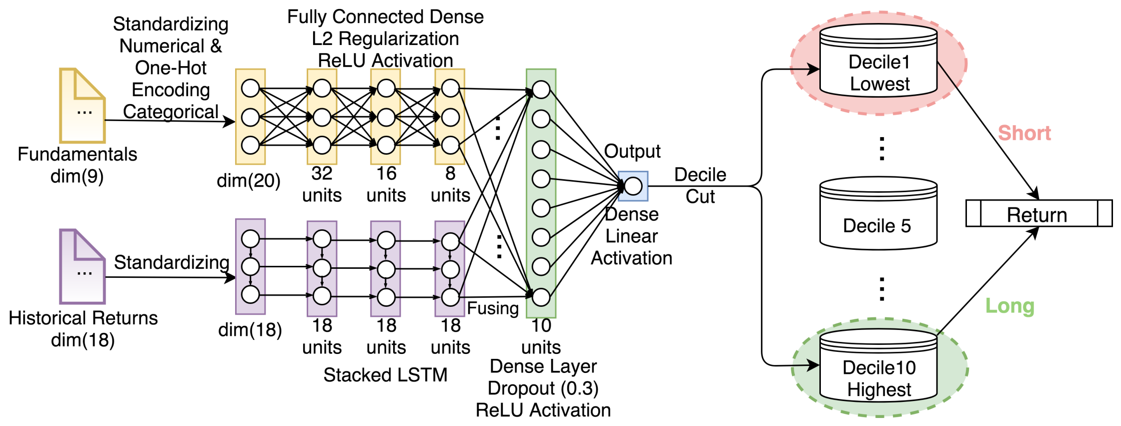

Figure 5.1 LSTM-DNN model and long-short portfolio construction. The yellow part is a fully-connected neural network (DNN) to model the latest fundamental variables. Due to one-hot encoding of categorical fundamental variables, the input dimension of DNN changes from 9 to 20. The purple part is a stacked LSTM network to model the time-series historical data. The green dense layer is the fusing layer, and the blue unit is the final output. With predictions from LSTM-DNN model, we rank the stocks and cut the top and bottom decile to construct the long-short portfolio.

5.4 Experiment

5.4.1 Baseline Methods

We compare our method with following six competing baselines:

- Naive Momentum Strategy (Naive_MM) [6]
- Ordinary Least Square (OLS)
- AutoRegression (AR)
- Support Vector Machine (SVM)
- Random Forest Regression (RF)
- Long Short-Term Memory (LSTM)
- Fully-Connect Deep Neural Networks (DNN)

The conventional momentum strategy (Naive_MM), proposed by Chan et al., ranks stocks by the average of past t-2 to t-12 monthly return, and then longs the stocks in the winner decile (top 10%) and shorts the stocks in the loser decile (bottom 10%) [6]. Fitting an OLS model is another popular technique in finance analysis. During the last decade, SVM and RF have exhibited the state-of-the-art performance in both classification and regression tasks [14, 49, 7, 45, 3]. In addition, we also compare our model with the traditional deep learning-based models: Long Short-Term Memory (LSTM) and Deep Neural Network (DNN). The LSTM takes the historical lagged return time series as inputs, while the DNN uses both lagged returns and fundamental variables. To this end, we conduct experiments on both traditional methods (Naive_MM, OLS, SVM and RF) as well as deep learning-based methods (LSTM and DNN) to comprehensively evaluate the performance among competing methods.

We tune the hyper-parameters for each model based on the validation performance. Specifically, for SVM, we grid search C in {0.1, 1.0, 10}, γ in {0.0001, 0.001, 0.01, 0.1} and ϵ in {0.01, 0.1}. The kernel type is fixed by Radial Basis Function (RBF). For RF, we grid search the *max_features* in {10, 20, 30}, *max_depth* in {3, 5, 7, 9} and *n_estimators* in {50, 100, 200, 500, 1000}. For LSTM, DNN and the proposed LSTM-DNN model, we apply the same method to tune the structure and hyperparameters (Figure 5.1 shows the structure of the proposed LSTM-DNN model).

5.4.2 Evaluation Metrics

Following previous studies, the average expected monthly return (MAR) is used to evaluate the performance of competing models [25, 29, 21, 54, 8]. Mathematically,

$$MAR_t = \overline{\{r_t^i\}}_{i \in \hat{\Omega}_t^{long}} - \overline{\{r_t^j\}}_{j \in \hat{\Omega}_t^{short}} \tag{5.11}$$

In addition, the Sharpe Ratio (SR), which is popular in finance literature, is also considered as the second evaluation metric. Formally,

$$SR = (R_p - R_f)/\sigma_p \tag{5.12}$$

where R_p is the average portfolio return, R_f is the 1-month treasury bill return, and σ_p represents the portfolio return standard deviation.

5.5 Results

Following Gu et al. [25], we calculate one-month-ahead out-of-sample stock return predictions for each method at the end of each month. For each model, we sort the stocks in ascending order based on the predictions. The ordered stocks are equally divided into 10 buckets, denoted as Decile 1 - 10 (See the first column of Table 5.2). Specifically, Decile 1 and 10 represent the stock sets with the lowest and highest investment returns, respectively. The equity long-short strategy takes long positions in stocks that are expected to increase in value (Decile 10) and short positions in stocks that are expected to decrease in value (Decile 1) to earn the spread. The profit of the equity long-short strategy is presented in Table 5.2. To clearly illustrate the results, we calculate the average of actual monthly returns for the stocks in each bucket.

Here we make a few remarks. Firstly, the Decile 1 and Decile 10 obtained based on the predictions from LSTM-DNN have the lowest and highest actual returns, respectively, which demonstrates that, compared with other methods, LSTM-DNN delivers more accurate monthly return predictions. Secondly, LSTM-DNN also provides better results for the middle Deciles (Decile 2-9). As shown in Table 5.2, Decile 2-9 obtained via LSTM-DNN have incremental returns in most cases. With standardized time series data and fundamental variables, LSTM-DNN could generally achieve incremental results from Decile 1 to Decile 10. In contrast, the

Table 5.2 Monthly Actual Return (in percent, %) for Decile 1-10 with Standardized or Unstandardized Inputs

		Standardized								Unstandardized				
Decile	Naive_MM	Lag18			Lag 18 + FDMT					Lag 18 + FDMT				
		OLS	DNN	LD[1]	OLS	DNN	RF	SVM	LD[1]	OLS	DNN	RF	SVM	LD[1]
1 (Low)	1.372	0.488	0.456	0.158	0.118	0.097	1.442	1.438	-0.227	0.522	0.759	1.476	1.331	0.422
2	1.097	1.026	0.936	1.191	0.744	0.879	1.225	1.441	0.698	0.846	0.987	1.302	1.696	0.79
3	1.018	1.183	1.096	1.16	0.917	0.809	1.379	1.336	1.038	0.846	1.194	1.304	1.165	0.983
4	1.054	1.228	1.088	1.225	1.033	1.308	1.131	1.224	1.105	1.088	1.256	1.137	1.3	1.104
5	1.166	1.128	1.302	1.133	1.327	1.045	1.108	1.178	1.297	1.222	1.155	1.303	1.229	1.137
6	1.171	1.107	1.306	1.325	1.129	1.283	1.201	1.172	1.205	1.1	1.313	1.239	1.142	1.225
7	1.264	1.326	1.176	1.291	1.343	1.325	1.055	1.364	1.35	1.318	1.365	1.034	1.264	1.308
8	1.474	1.383	1.396	1.336	1.494	1.4	1.174	1.19	1.524	1.401	1.334	1.274	1.283	1.461
9	1.755	1.561	1.486	1.452	1.762	1.753	1.325	1.219	1.676	1.612	1.371	1.22	1.294	1.643
10 (High)	1.624	2.564	2.77	2.725	3.127	3.095	2.205	1.432	3.329	3.037	2.26	1.732	1.292	2.92
Profit (H-L)	0.252	2.076	2.314	**2.567**	3.009	2.998	0.763	-0.006	**3.556**	**2.515**	1.501	0.256	-0.039	**2.498**

[1] LD: LSTM-DNN.

results obtained via other methods are much worse that some predicted lower Deciles have much larger actual returns than the predicted higher Deciles (For example, Decile 8 and Decile 9 achieved by SVM have much lower actual returns than the Decile 1-2.). Thirdly, the profit of the equity long-short strategy could be maximized by integrating the proposed LSTM-DNN method. With the standardized inputs and the proposed LSTM-DNN model, we could achieve the highest portfolio-level profit of 3.556%.

Table 5.2 also empirically demonstrates the contribution of fundamental variables. When integrated with fundamental features, we have higher returns in Decile 10 and lower returns in Decile 1. The observation is consistent across different methods including OLS, DNN, and LSTM-DNN, strongly indicating that fundamental information could help the model make more accurate predictions and build better portfolios. Interestingly, the predicted 'winner' portfolio (Decile 10) constructed by LSTM-DNN framework achieves 3.329% monthly return, whereas the predicted 'loser' portfolio has a negative monthly return (i.e., -0.227%). These results suggest that the statistical inference of LSTM-DNN contributes to the final profit from both 'long' and 'short' sides, when we apply the equity long-short strategy.

Since we use the batch sliding window to do the evaluation, Table 5.3 presents the monthly average return (MAR) and Sharpe Ratio (SR) for different methods with standardized/unstandardized inputs in different testing periods. LSTM-DNN model with cross-section standardized inputs demonstrates the best performance, measured by MAR and SR, in most batches and yields strongly comparable performance in other batches. By comparing the upper panel and lower panel, we observe that cross-section normalization procedure generally benefits all the models.

Table 5.3 Monthly Average Return (MAR) (in percent, %) and Sharpe Ratio (SR) for Each Method at Different Testing Periods

	Features	Batch	198901-199312		199401-199812		199901-200312		200401-200812		200901-201312		201401-201812	
			MAR	SR	MAR	SR	MAR	SR	MAR	SR	MAR	SR	MAR	SR
S[1]	Lag18	OLS	2.912	0.607	3.235	0.743	3.151	0.281	0.854	0.141	0.889	0.198	-0.009	-0.003
		AR	2.638	0.590	4.231	1.196	3.123	0.376	1.288	0.201	0.568	0.142	0.617	0.118
		DNN	3.157	0.810	3.432	0.853	3.103	0.333	1.271	**0.309**	0.445	0.096	0.447	0.157
		LSTM	2.818	0.589	4.503	**1.366**	3.787	0.468	**1.286**	0.241	0.746	0.155	0.616	0.134
	Lag18 + FDMT	OLS	4.486	1.028	3.471	0.857	4.796	0.494	0.832	0.152	1.727	0.255	1.026	0.297
		AR	4.404	0.997	3.156	0.942	4.756	0.497	0.796	0.148	0.189	0.245	0.576	0.131
		DNN	4.369	0.990	4.332	1.220	4.748	0.545	0.948	0.215	1.121	0.207	0.615	0.171
		LD	**4.549**	**1.254**	**4.967**	1.350	**5.639**	**0.679**	1.156	0.263	1.667	0.332	**1.319**	**0.340**
		RF	0.524	0.103	1.464	0.281	-0.166	-0.028	-0.882	-0.177	0.49	0.076	0.313	0.051
		SVM	-0.787	-0.120	1.73	0.526	-0.385	-0.045	-0.108	-0.024	-0.93	-0.161	-0.047	-0.009
US[2]	Lag18	OLS	2.31	0.549	0.859	0.213	1.373	0.143	0.711	0.128	-0.371	-0.079	-0.106	-0.027
		DNN	1.31	0.392	0.838	0.236	0.845	0.104	0.865	0.233	0.928	0.157	0.226	0.057
		LSTM	2.677	0.643	0.878	0.173	3.798	0.310	1.169	0.226	1.22	0.129	-1.118	-0.183
	Lag18 + FDMT	OLS	3.879	0.867	2.245	0.462	4.214	0.436	1.141	0.183	**2.286**	0.249	0.473	0.142
		DNN	1.749	0.502	1.298	0.347	3.779	0.478	0.365	0.069	1.969	**0.347**	-0.084	-0.021
		LD	4.381	0.894	2.199	0.543	4.368	0.518	0.892	0.234	2.175	0.241	-0.246	-0.047
		RF	0.029	0.006	0.871	0.147	0.615	0.059	-1.193	-0.237	1.159	0.161	-0.572	-0.141
		SVM	-0.568	-0.098	1.57	0.455	-0.639	-0.068	-0.156	-0.034	-0.543	-0.097	-0.297	-0.059

[1] S: Standardized.

[2] US: Unstandardized.

Figure 5.2 visualizes the cumulative profit for each method over the entire out-of-sample test period. For each method, we start trading with 1 dollar from Jan 1st, 1989 and calculate the cumulative value of the portfolio for each month. To avoid a crowded figure, all methods presented in the figure use the standardized inputs. The period of the global financial crisis in 2008 is highlighted in gray in Figure 5.2. As shown in Figure 5.2, LSTM-DNN model could achieve the highest cumulative return from the whole testing period (1989 - 2019). In addition, adding fundamental features makes better predictions, as all the methods with 'Lag18_FDMT' are more profitable than the methods with 'Lag_18'. It should be noted that the investment experiment is conducted under simplified scenarios and external factors (e.g., transaction fee)

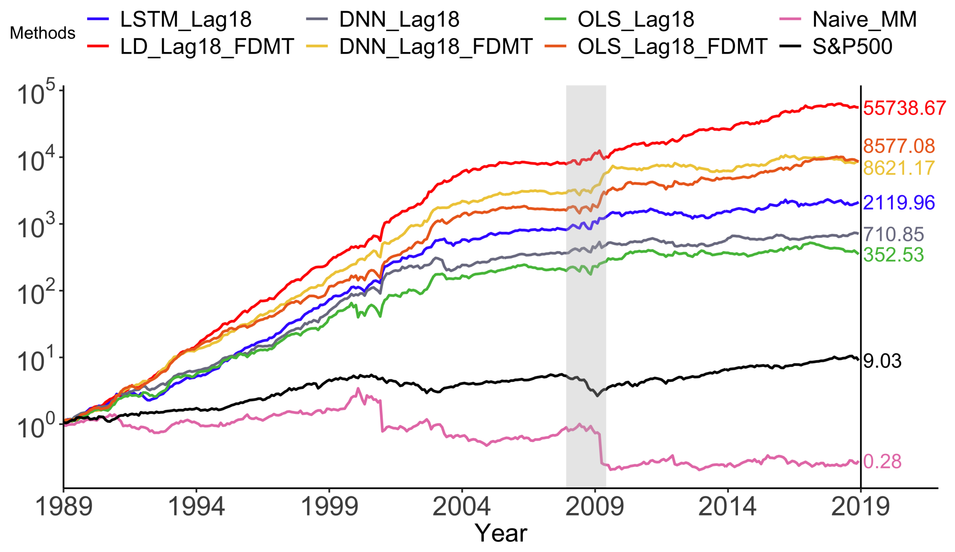

Figure 5.2 Cumulative returns. The x axis is the six testing periods and y axis is the cumulative return. The shaded area represents the 2008 global financial crisis period. LD represents the LSTM-DNN method.

are not considered. In the future, we'd like to build more sophisticated simulation frameworks to provide more accurate analytical results.

For the second aspect, we find that the risk-adjusted LSTM-DNN strategy has similar performance compared with its original returns across the whole testing period. Specifically, we regress the return of the LSTM-DNN model on the Fama-French three factors, and the resulting intercept of the regression is the risk-adjusted return of the strategy. Thus, it is very likely that the LSTM-DNN model obtains better performance by extracting more complex interactions among factors rather than simply leveraging the risk exposures to common risk factors.

5.6 Conclusion

In this paper, we propose a hybrid neural network framework to integrate both time-series data and stable fundamental features for the stock return prediction. A cross-section data normalization method is introduced for enriching the signal of the relative performance of each stock. We conduct various experiments on the CRSP data and show that the proposed method could achieve better performance than existing methods for the long-short portfolio strategy. By comparing linear modeling and the proposed framework, we empirically demonstrate that the nonlinear interactions between the fundamentals and lagged returns have positive impacts on the prediction results. Finally, we conduct comprehensive examinations from multiple perspectives, like the robustness of strategy and the long-short cutting percentage, to demonstrate the gross advantages. In a word, we empirically show that the higher return obtained by our method is not achieved by taking higher risks or picking a favorable threshold.

CHAPTER 6

ST-TRADER: A SPATIAL-TEMPORAL DEEP NEURAL NETWORK FOR MODELING STOCK MARKET MOVEMENT

6.1 Introduction

Forecasting stock market is usually a time-series forecasting problem in the literature. But stocks that are fundamentally connected with each other tend to move together. First, exchange-traded funds (ETFs), such as S&P 500 and NASDAQ, track the prices of a basket of stocks. When people trade those funds, all the underlying stocks are traded simultaneously, which causes common fluctuations of those stock prices (see [1]). Second, most professional portfolio managers are specialized in a couple of strategies and a strategy often involves a similar set of stocks. For example, value investing (see [24]) tilts to firms with high earning-to-price ratio, while momentum strategy focuses on firms with higher returns during the past year. On the one hand, any fundamental shock can affect the prices of a group of stocks together. Considering such common trends is believed to benefit stock movement forecasting tasks. However, such signals are not trivial to model because the connections among stocks are not physically presented and need to be estimated from volatile data.

This chapter proposes a novel pipeline to utilize the fundamentally spatial dependency among firms. First, it uses Variational AutoEncoder to reduce the dimension of stock fundamental information and then cluster stocks into a graph structure (fundamentally clustering). Second, a hybrid model of Graph Convolutional Network and Long-Short Term Memory network (GCN-LSTM) with an adjacency graph matrix (learnt from Variational AutoEncoder) is designed for graph-structured stock market forecasting.

6.2 Methods

6.2.1 Problem Formulation

Consider N stocks and each stock has M fundamental variables. The fundamentals of stock i can be represented by a vector $F_i = \{f_i^{(1)}, f_i^{(2)}, ..., f_i^{(M)}\} \in \mathbb{R}^M$. Moreover, the fundamental matrix for all stocks is $\boldsymbol{F} = \{F_1, F_2, ..., F_N\} \in \mathbb{R}^{N \times M}$. There is not a time subscript since fundamental features are not time-varying. With fundamental matrix, $\boldsymbol{F}$, we construct a graph, $\mathcal{G} = (V, E)$, where V is the set of $|V| = N$ vertices (stocks), E is the set of edges representing the spatial relationship between stocks learned from function $E = \phi(\boldsymbol{F})$, and $E \in \mathbb{R}^{N \times N}$.

For temporal dependency, if the look-back window is T, a time-series vector for stock i can be represented as $X_i = \{x_i^1, x_i^2, ..., x_i^T\} \in \mathbb{R}^T$. Then, we express the data collected for all stocks in a matrix $\boldsymbol{X} = \{X_1, X_2, ..., X_N\} \in \mathbb{R}^{N \times T}$.

Our goal is to forecast the price of each stock in the next time point. With both spatial and temporal features ready, we design a hybrid model of Graph Convolutional Network and Long-Short Term Memory network (GCN-LSTM), denoted as $f_\theta(\cdot)$. We further denote the true values and the predicted values as $y = \{x_1^{T+1}, x_2^{T+1}, ..., x_N^{T+1}\}$ and $\hat{y} = \{\hat{x}_1^{T+1}, \hat{x}_2^{T+1}, ..., \hat{x}_N^{T+1}\}$, respectively. The network parameters, θ, can be estimated as,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (f_\theta(\mathcal{G}, X_i) - y_i)^2 \tag{6.1}$$

6.2.2 Variational Autoencoder

A Variational AutoEncoder (VAE) is a kind of directed probabilistic graphical model whose posterior is approximated by a neural network. In Figure 6.1, we represent the directed graphical model in the area shaded with green color. The generative process starts from the bottle-neck of VAE, z, which is a latent variable. $q_\phi(F'|z)$ represents the data generating process that results in the reconstructed input F'.

Because the marginal likelihood is intractable, the objective function of a VAE is the variational lowerbound of the marginal likelihood of data. And the marginal likelihood is the sum over the marginal likelihood of each individual fundamental variable $\log p_\phi(f^{(1)}, ..., f^{(M)}) = \sum_{i=1}^{M} \log p_\phi(f^{(i)})$. The the marginal likelihood of individual fundamental variables can be written as

$$\log p_\phi(f^{(i)}) = D_{KL}(q_\psi(z|F)||p_\phi(z)) + \mathcal{L}(\phi, \psi; f^{(i)}) \tag{6.2}$$

where $q_\psi(z|F)$ is the approximate posterior and $p_\phi(z)$ is the prior distribution of the latent variable z. The first term on the right hand side of (6.2) means the KL divergence between the approximate posterior $q_\psi(z|F)$ and the prior $p_\phi(z)$. The second term is the objective variational lowerbound on the marginal likelihood of feature i. Since the KL divergence term is always greater than 0, (6.2) can be rewritten as follows.

$$\begin{align}
\log p_\phi(f^{(i)}) &\geq \mathcal{L}(\phi, \psi; f^{(i)}) \tag{6.3}\\
&= E_{q_\psi(z|f^{(i)})}\left[-\log q_\psi(z|F) + \log p_\phi(F|z)\right] \tag{6.4}\\
&= -D_{KL}(q_\psi(z|f^{(i)})||p_\phi(z)) \nonumber\\
&\qquad + E_{q_\psi(z|f^{(i)})}\left[\log p_\phi(F|z)\right] \tag{6.5}
\end{align}$$

where $p_\phi(F|z)$ is the likelihood of the fundamental feature vector F given the latent variable z. The first term of (6.5), the KL divergence, works as a regularization term which pulls the posterior distribution to the prior distribution. The second term of (6.5) is the reconstruction of F through the posterior distribution $q_\psi(z|F)$ and the likelihood $p_\phi(F|z)$. The training goal is to estimate the parameters ψ and ϕ that have minimal loss between reconstruction input and original input. Ideally, the reconstructed input F' and original input F are identical.

Using the bottle-neck feature representation of VAE, we calculate the Euclidean distance between each pair of stocks, $d_{i,j}$ and define the adjacency matrix, A, as following:

$$A_{ij} = \begin{cases} exp(-\frac{d_{ij}^2}{\gamma^2}) & , i \neq j \text{ and } exp(-\frac{d_{ij}^2}{\gamma^2}) \geqslant \epsilon \\ 0 & , \text{otherwise.} \end{cases} \tag{6.6}$$

where $\gamma^2 = 0.1$ and $\epsilon = 0.5$ are thresholds to control the distribution and sparsity of adjacency matrix A. A_{ij} is the learnt distance between stock i and stock j.

6.2.3 GCN-LSTM Model

We propose to incorporate the spatial signal into time-series prediction with graph convolutional network. The global spatial dependency learnt via VAE is represented by the adjacent matrix, A, calculated from the features from low-dimensional latent space. For notational simplicity, we write $*_{\mathcal{G}} x_t$ to mean a convolution operation on x_t with filters (kernel size: $d_h \times d_x$) that are functions of the graph Laplacian L. By replacing the original inputs with convolution-applied inputs, the equations of GCN-LSTM cell are given as follows:

$$f_t = sigmoid\,(W_{f,x} *_{\mathcal{G}} x_t + W_{f,h} *_{\mathcal{G}} h_{t-1} + b_f) \tag{6.7}$$

$$i_t = sigmoid(W_{i,x} *_{\mathcal{G}} x_t + W_{i,h} *_{\mathcal{G}} h_{t-1} + b_i) \tag{6.8}$$

$$o_t = sigmoid(W_{o,x} *_{\mathcal{G}} x_t + W_{o,h} *_{\mathcal{G}} h_{t-1} + b_o) \tag{6.9}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{c,x} *_{\mathcal{G}} x_t + W_{c,h} *_{\mathcal{G}} h_{t-1} + b_c) \tag{6.10}$$

$$h_t = o_t \circ tanh(c_t) \tag{6.11}$$

In our setting, $W_{\cdot,h} \in \mathbb{R}^{K \times d_h \times d_h}$ are the Chebyshev coefficients that defines the support K of the graph convolutional kernels. $W_{\cdot,x} \in \mathbb{R}^{K \times d_h \times d_x}$ determines the number of parameters that is independent to the number of nodes N in the graph. Parameter K determines the number of neighbors used to compute the aggregated states for any target node i so that it also determines the communication overhead in a distributed computing setting. Figure 6.1 visualize the proposed pipeline and the detailed algorithm description is shown in Algorithm 1.

Algorithm 1: Training Process for the ST-Trader

```
Input      : $\boldsymbol{F} \in \mathbb{R}^{N \times M}$. Fundamental feature matrix with $M$ variables
             for $N$ stocks;
             $\boldsymbol{X} \in \mathbb{R}^{N \times T}$. Time-series feature matrix with look-back
             window $T$ for $N$ stocks;
             $\gamma^2 = 0.1$; $\epsilon = 0.5$; minibatch size 256; polynomial order
             $K = 3$; number of epoches $E = 1000$; the hidden space
             dimension $h = 16$; network depth $D = 3$.
Parameters: $\phi$, $\psi$ of VAE; $\theta$ of GCN-LSTM.
$\phi$, $\psi \leftarrow$ Initialize parameters;
repeat
    $g \leftarrow \nabla_{\phi,\psi}\mathcal{L}(\phi, \psi; \boldsymbol{F})$ (Gradients of minibatch estimator of (3))
     $\phi, \psi \leftarrow$ Update parameters using gradients $g$ (s.g. SGD or Adam
     optimizer);
until convergence of parameters $(\phi, \psi)$;
The adjacent matrix $A$ for the edge information in $\mathcal{G} \leftarrow$ Distance of latent
 features derived by the VAE using parameters $\phi, \psi$;
$\tilde{L} \leftarrow$ Calculate rescaled Laplacian using $A$; $\theta \leftarrow$ Initialize parameters;
$e \leftarrow 1$; while $e \geq E$ do
    $\boldsymbol{X}' \leftarrow$ apply convolution operator $*_{\mathcal{G}}$ on $\boldsymbol{X}$ $\hat{y} \leftarrow LSTM(\mathbf{X}')$ Update
     $W_{x\cdot}$, $W_{h\cdot}$, and $b$ in $\theta$ by gradient descent $e \leftarrow e + 1$
```

6.3 Experiment

6.3.1 Data Description

Since we consider the spatial dependency among firms, the number of our training samples is divided by the number of firms. For example, suppose we have 10 firms and 100 observations for each firm. The total number of samples is 1,000 if we take firms independently but the number drops to 100 if we consider the relations among firms. To obtaining enough samples, we use minute-level stock data 87 firms from

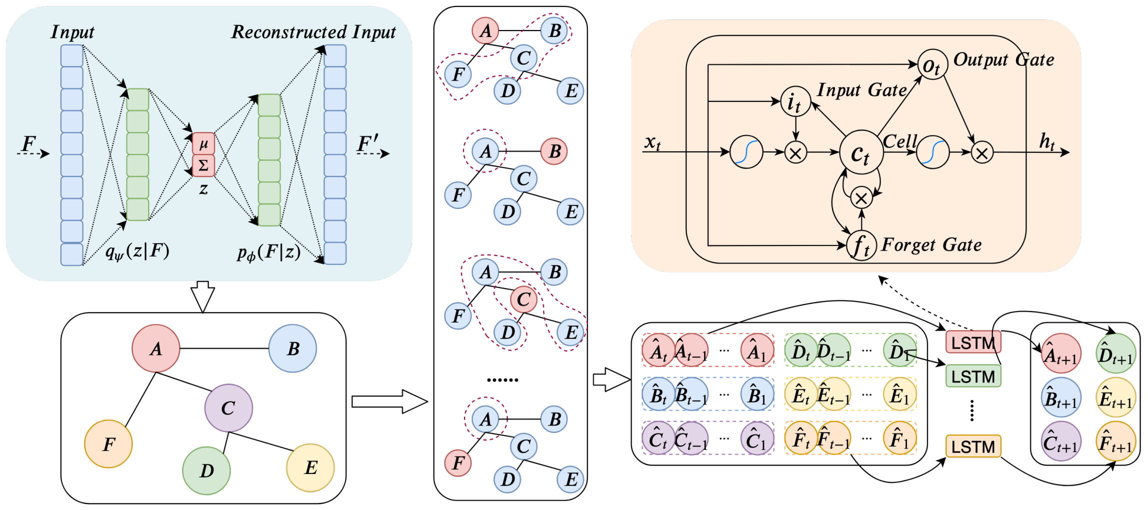

Figure 6.1 Spatial-Temporal modeling using GCN-LSTM framework for unspecified spatial graph structure. The area shaded with green color is VAE, which reduces the dimension of fundamental feature to learn more meaningful distance among stocks. The network below it is the constructed graph based on the learnt distance. The vertical panel to the right of VAE presents the convolution neighbors of each node. The area shaded with yellow background is the network of a LSTM cell. The time-series inputs enriched with fundamental signals by convolution operation are fed into a LSTM network for final predictions.

S&P 100 composite in 2010 due to the availability of data. The number of total minute-observations is 97,890, and we split the whole dataset into batches using the sliding window. We also check the robustness of our results using five-minute-interval stock prices, which guarantees enough samples for at least one epoch training and testing. For five-minute-interval, the sample sizes of training, validation, and testing set are 16384, 2944, 2944 respectively. The used fundamental variables and stock tickers are presented in Table 6.3.1 and Table 6.2.

Table 6.1 Firm Fundamental Variables

Abbreviation	**Full Name**
AT	Assets (Total)
INTAN	Intangible Assets (Total)
BKVLPS	BookValue Per Share
DLTT	Long Term Debt (Total)
DLC	Debt in Current Liabilities (Total)
LT	Liabilities (Total)
RE	Retained Earnings
ICAPT	Invested Capital (Total)
IB	Income Before Extraordinary Iterms
CHE	Cash and Short-Term Investments
PPEGT	Property, Plant and Equipment (Total)
DVT	Dividends (Total)
EBIT	Earnings Before Interest and Taxes
GP	Gross Profit (Loss)
DV	Cash Dividends (Cash Flow)
CAPX	Capital Expenditures
TXPD	Income Taxes Paid
SEC	Stock Exchange Code
SIC	Standard Industry Classification Code

6.3.2 Experimental Settings

For one-minute-level data, the testing period is one month after the training period. For example, if the testing is February 2012, the training data would be January

Table 6.2 Stock Ticker List

Ticker	Ticker	Ticker
AAPL	DUK	NFLX
ABT	EMR	NKE
ACN	EXC	NVDA
ADBE	F	ORCL
AGN	FDX	OXY
AIG	GD	PEP
ALL	GE	PFE
AMGN	GILD	PG
AMZN	GOOG	PM
AXP	GS	QCOM
BA	HD	RTN
BAC	HON	SBUX
BIIB	IBM	SLB
BK	INTC	SO
BMY	JNJ	SPG
BRKB	JPM	T
C	KO	TGT
CAT	LLY	TMO
CL	LMT	TXN
CMCSA	LOW	UNH
COF	MA	UNP
COP	MCD	UPS
COST	MDT	USB
CSCO	MET	UTX
CVS	MMM	V
CVX	MO	VZ
DD	MRK	WFC
DHR	MS	WMT
DIS	MSFT	XOM

2012. The last testing period is Dec 2012. Thus, we have 11 testing period and the number of samples for each month is listed in Table 6.3.

Table 6.3 Number of Samples for Each Month

Month	Jan	Feb	Mar	Apr	May	Jun
# of samples	7,350	7,410	8,970	8,190	7,800	8,680
Month	Jul	Aug	Sep	Oct	Nov	Dec
# of samples	8,190	8,580	8,190	8,190	7,800	8,580

To demonstrate the benefit of incorporating the spatial dependency among stocks on price forecasting, we consider the following baseline methods: (1) **LR**: The classical linear regression model with time-series features as input; (2) **FCNN**: The fully-connected neural network which captures the non-linear relationship between time-series features; (3) **LSTM**: Long-short Term Memory neural network which contributes partially to the proposed method; (4) **ecldn_ST-Trader**: 'ecldn' means calculating the Euclidean Distance between a pair of stocks using the original fundamental variables. The spatial relationship in this setting is expected to be much noisier than using VAE; (5) **idsty_ST-Trader**: 'idsty' means 'industry'. The adjacent matrix for this method is derived from the industry category. $A_{ij} = 1$ means company i and company j are in the same industry category. The proposed model is denoted as **vae_ST-Trader** when compared to those baselines methods because it differs in the method of deriving the adjacent matrix via Variational AutoEncoder. The purpose of studying baseline methods (4) and (5) is to evaluate the ability of VAE of extracting latent features from high dimension feature space. The network structure and the hyperparameters for all methods, are tuned using the validation set and the final performance results are derived on only the testing set.

We apply all methods mentioned above to forecast two targets: the numerical stock price and the binary price movement indicator. Since deep neural networks are not stable when predicting unbounded numerical results, we scale both the training set and testing set using MIN-MAX normalization (see (6.12)) by the maximum and minimum value of the training set.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{6.12}$$

6.3.3 Results

Predicting Stock Price Table 6.4 presents the MAPE and MdAPE for all testing periods. The stock price has been demonstrated to have extensive outliers because MdAPE is usually less than MAPE. The methods enriched with spatial information achieve better prediction results than temporal-only models. The proposed model, vae_ST-Trader, outperforms baselines across the board. Moreover, LSTM is more desirable than LR and FCNN in most batches. Interestingly, idsty_ST-Trader with only industry information is more preferable than ecldn_ST-Trader, which incorporates much more fundamental information. One possibility is that simple Euclidean distance calculated from all fundamental variables brings more noise to the spatial signal because it assigns equal weight to each variable. The contribution of each variable to the final prediction is hard to be quantified by the linear model. Clearly, extracting the latent interaction (spatial distance on the latent features) among firms using VAE benefits the prediction accuracy substantially.

Table 6.4 MAPE(MdAPE) for One-Minute-Interval Prediction

Training Month	201201	201202	201203	201204	201205	201206
Testing Month	201202	201203	201204	201205	201206	201207
LR	0.1579 (0.1679)	0.2056 (0.1534)	0.1684 (0.1392)	0.2313 (0.1877)	0.1672 (0.1282)	0.1641 (0.1439)
FCNN	0.1417 (0.1171)	0.227 (0.1044)	0.1418 (0.1065)	0.2314 (0.1831)	0.155 (0.1139)	0.1503 (0.1139)
LSTM	0.1077 (0.2313)	0.2145 (0.0800)	0.1512 (0.1947)	0.2283 (0.1789)	0.1058 (0.0868)	0.1398 (0.1037)
ecldn_ST-Trader	0.1206 (0.0772)	0.1018 (0.0655)	0.1605 (0.0752)	0.1857 (0.1594)	0.128 (0.0825)	0.1484 (0.1347)
idsty_ST-Trader	0.0904 (0.0699)	0.1142 (0.0714)	**0.1203** (0.0619)	**0.1426** (**0.1319**)	0.1018 (0.0741)	0.1305 (0.0899)
vae_ST-Trader	**0.0789 (0.0686)**	**0.0827 (0.0611)**	0.132 (**0.0583**)	0.1488 (0.1327)	**0.0863 (0.0681)**	**0.1274 (0.0686)**

Training Month	201207	201208	201209	201210	201211
Testing Month	201208	201209	201210	201211	201212
LR	0.1708 (0.1947)	0.1589 (0.1302)	0.1398 (0.1334)	0.1556 (0.1203)	0.2188 (0.1748)
FCNN	0.1364 (0.1302)	0.1478 (0.1389)	0.1478 (0.1268)	0.1481 (0.1143)	0.1995 (0.1810)
LSTM	0.1995 (0.1815)	0.1089 (0.0907)	0.1023 (0.0946)	0.1175 (0.1098)	0.201 (0.1369)
ecldn_ST-Trader	0.1478 (0.1018)	0.0856 (0.0739)	0.1119 (0.0980)	0.0856 (0.0866)	0.1802 (0.1368)
idsty_ST-Trader	0.0975 (0.0933)	0.0917 (0.0741)	0.0926 (0.0744)	0.0863 (0.0749)	0.1419 (0.1084)
vae_ST-Trader	**0.0781 (0.0816)**	**0.0902 (0.0728)**	**0.0882 (0.0606)**	**0.0787 (0.0627)**	**0.1358 (0.0780)**

Bold values indicate the best results

Prediction Stock Price Movement Figure 6.2 presents the Accuracy and AUC scores of different methods on binary movement prediction chronologically. According to the Efficient-market hypothesis [1] that asset prices reflect all available information, there is not much space for algorithms to forecast the future stock prices, which has been demonstrated by the poor accuracy in our study (Figure 6.2) and in much many literature. Although many investors apply value-investing strategies, which tie the market price of a stock to its underlying fundamental value, many other investors keep adopting technical analysis. They make trading decisions based on reading charts of the historical price trends. Therefore, there is still room for the methods that do not take the influence of the technical-analysis trader on stock price into account, to improve their predictive power. The proposed model is enriched by the extracted relationship among the firms so that it is better able to capture the trend signal compared to baseline methods. This advantage is reflected by both Accuracy and AUC scores.

For the extreme short-term price movements in the market, including the flash crash on May 6, 2020.[2] The Dow Jones Industrial Average fell more than 1,000 points in 10 minutes around 2:30 p.m. EST on May 6, 2010, which was the biggest drop in history at that point. Trillions of dollars in equity was wiped out, but the market recovered to the pre-crisis level by the end of the day. Analyzing the causes of such events are beyond the scope of this paper. However, it has a profound effect on the stability of our algorithm since these extreme events can hardly be thought of as shocks from fundamental information. Incorporating fundamental connections in the prediction task may lower the prediction accuracy under this particular scenario. By avoiding such events, we may expect that vae_ST-Trader has a better performance in predicting the longer time-interval stock price, e.g., day-to-day or week-to-week.

[1]https://en.wikipedia.org/wiki/Efficient-market_hypothesis

[2]https://en.wikipedia.org/wiki/2010_flash_crash.

However, we are not able to do experiments on that due to the lack of sufficient number of observations.

The number of neighbors to communicate with We compare performance of the proposed model on different granularity and different parameter K in Table 6.5. We denote one-minute-interval and five-minute-interval price forecasting as $\Delta t = 1$ and $\Delta t = 5$. The superscript 1 and 5 highlight the best scores for $\Delta t = 1$ and $\Delta t = 5$. The results for $\Delta t = 1$ are aggregated across all testing months. There are two results worth mentioning. First, the outcomes of $\Delta t = 5$ are better than $\Delta t = 1$ for all evaluation metrics. This is as expected because the price movement of five-minute-interval is much less noisy than one-minute-interval due to a couple of reasons: (1) Rare events like the flash crash can recover soon so that five-minute-level data can almost screen out such events; (2) The recorded stock price is bounced back and forth between the bid and ask quote and the price fluctuation in five-minute-level is less likely to be affected by the bid-ask bounce. If we have enough number of price observations on a longer interval, the predictive power of the proposed model can be expected to improve.

Second, for the communication overhead parameter K, which is the number of nodes any target node i should exchange signals with in order to derive its local states, we use different K to explore how the communication affects the performance in different granularity. One-minute-interval achieves best performance when $K = 3$ and five-minute-interval achieves best performance when $K = 5$, which means a finer granularity prefers a lighter communication to its neighbors. We give one possible explanation. For one-minute-interval, where the fluctuation of price is more random, and hence the dependency is less reliable, the communication with more neighbors brings more noise in forecasting. While along with the time interval increasing, the price trend becomes more stable and the common fluctuation is more promising so that the infusion of neighborhood signals can be more informative. Thus, the number

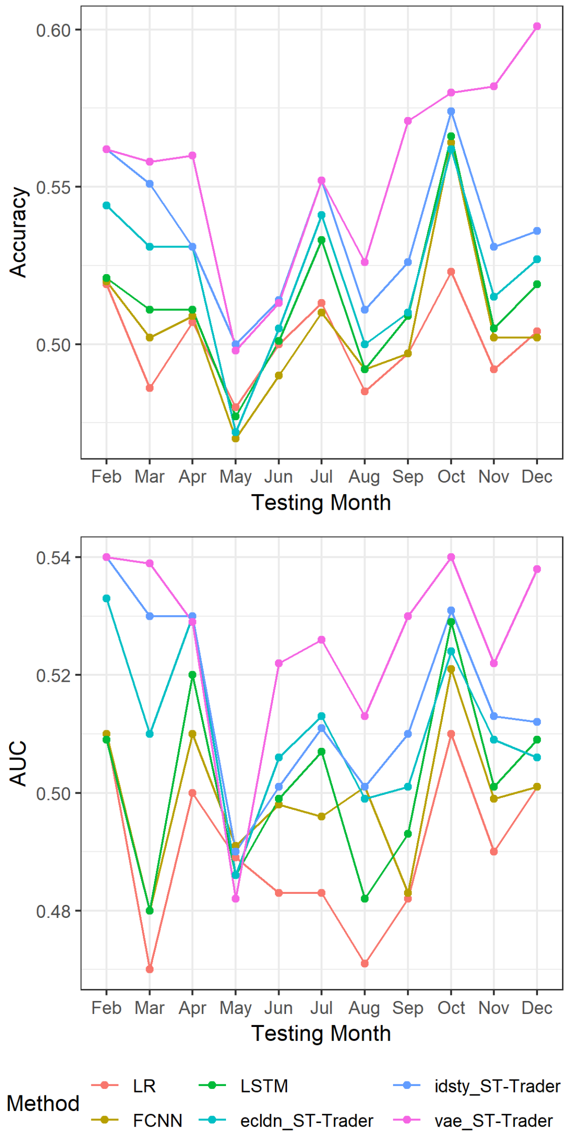

Figure 6.2 ACC and AUC comparison for different methods across different testing months.

Table 6.5 Binary Price Movement Prediction with Different Parameter K for One-Minute-Interval and Five-Minute-Interval

		ACC	AUC	Precision	Recall	F1 Score
$K = 3$	$\Delta t = 1$	0.567^1	0.523^1	0.469	0.619^1	0.534^1
	$\Delta t = 5$	0.577	0.556	0.474	0.411	0.440
$K = 5$	$\Delta t = 1$	0.544	0.511	0.471^1	0.551	0.508
	$\Delta t = 5$	0.587^5	0.568^5	0.492^5	0.665^5	0.565^5
$K = 7$	$\Delta t = 1$	0.552	0.518	0.471^1	0.560	0.512
	$\Delta t = 5$	0.563	0.519	0.475	0.592	0.527

of neighbors for supporting the center node is a key hyperparameter to tune for a specific time-series forecasting task.

In this chapter, we propose a spatial-temporal neural network framework GCN-LSTM, to utilize the spatial dependency or the latent interaction among firms in forecasting the stock price movement. The stock market has never been treated as a graph since there is not an inborn geographical location for stock entities. However, there is strong evidence that the interactions among firms affect the stock price movement. Experimental results show that our model outperforms other state-of-the-art methods on the real-world minute-level stock price data. Fundamental features represented in a spatial structure contribute to the forecasting accuracy improvement. For future directions, we plan to investigate how the combination of fundamental variables and fiscal reports, which can be seen as a dynamic cross-section assessment of a company, contributes to predicting the stock market trend. More practical time-series applications with potential spatial dependency should be explored under the proposed modeling framework. The advanced approaches [23, 70] to the fine tuning of hyper-parameters of the proposed framework should be explored.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This dissertation focuses on the development of hybrid neural networks for modeling heterogeneous input data for mining stock market and biological data. The main contributions of this dissertation are as listed below:

First, a CNN-DNN model is developed for authenticating CNV calls from other detection tools. By leveraging the auxiliary meta information, it can accurately identify the false positive calls and achieve comparable accuracy as human experts' labelling. With the superior performance on both human labelling dataset and experimentally validated dataset than the state-of-the-arts, we can foresee the detection routine would become much more intelligent in the near future.

Second, a LSTM-DNN model is proposed to integrate static fundamental information of firms into time-series forecasting tasks. By comparing linear modeling the proposed framework, we empirically demonstrate that discovering the nonlinear interactions between the fundamentals benefits the long-short portfolio strategy significantly. And we conduct comprehensive examinations from multiple perspectives, like the robustness of strategy and the long-short cutting percentage, to demonstrate the gross advantages, which means the higher return obtained by the proposed method is not achieved by taking higher risks or picking a favorable threshold.

Finally, a spatial-temporal framework VAE-GCN-LSTM, to utilize the latent interaction among firms is proposed in forecasting the stock price movement. The superiority of VAE to extract the distance among firms is demonstrated by comparing other possible distance, like Euclidean Distance and Industry Category Clustering. And the whole pipeline outperforms other state-of-the-art methods on different time granularity.

Future work lies in the following directions:

First, for the CNVs authentication task, it may be more desired if the deep learning-based method can be an end-to-end procedure which eliminates the dependence on the other detection tools. We hope to recognize CNVs directly from the raw data using deep learning. Another potential direction is to use Active Learning [63], which can select the boundary cases through a customized query function for labelling and retraining iteratively until the model achieves a fair performance, which can lower the human labeling cost much because it avoids labeling for obvious cases.

Second, for the enriched time-series forecasting in stock market, more fundamental features and longer historical time-series data can be explored. It is noted that the intra-day momentum may perform differently from the monthly momentum, and the long-short equity strategies can be different in many ways, in which the nonlinear relations among factors are not trivial to capture.

Third, for the spatial-temporal modeling in stock market, I plan to investigate how the combination of fundamental variables and fiscal reports, which can be seen as a dynamic cross-sectional assessment of a company, contributes to predicting the stock market trend. And more practical time-series applications with potential spatial dependency should be explored under the proposed modeling framework.

REFERENCES

[1] Itzhak Ben-David, Francesco Franzoni, and Rabih Moussawi. Do ETFs increase volatility? *The Journal of Finance*, 73(6):2471–2535, 2018.

[2] Tim Bollerslev, James Marrone, Lai Xu, and Hao Zhou. Stock return predictability and variance risk premia: statistical inference and international evidence. *Journal of Financial and Quantitative Analysis*, 49(3):633–661, 2014.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *ArXiv Preprint arXiv:1312.6203*, 2013.

[5] Andrew R Carson, Lars Feuk, Mansoor Mohammed, and Stephen W Scherer. Strategies for the detection of copy number and other structural variants in the human genome. *Human Genomics*, 2(6):403, 2006.

[6] Louis KC Chan, Narasimhan Jegadeesh, and Josef Lakonishok. Momentum strategies. *The Journal of Finance*, 51(5):1681–1713, 1996.

[7] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[8] Yingjun Chen and Yongtao Hao. A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, 80:340–355, 2017.

[9] Min Cheng, Qian Xu, Jianming Lv, Wenyin Liu, Qing Li, and Jianping Wang. Ms-lstm: A multi-scale lstm model for bgp anomaly detection. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2016.

[10] EK Cho, J Tchinda, JL Freeman, Y-J Chung, WW Cai, and C Lee. Array-based comparative genomic hybridization and copy number variation in cancer research. *Cytogenetic and Genome Research*, 115(3-4):262–272, 2006.

[11] Stefano Colella, Christopher Yau, Jennifer M Taylor, Ghazala Mirza, Helen Butler, Penny Clouston, Anne S Bassett, Anneke Seller, Christopher C Holmes, and Jiannis Ragoussis. Quantisnp: an objective bayes hidden-markov model to detect and accurately map copy number variation using snp genotyping data. *Nucleic Acids Research*, 35(6):2013–2025, 2007.

[12] Donald F Conrad and Matthew E Hurles. The population genetics of structural variation. *Nature Genetics*, 39(7):S30–S36, 2007.

[13] International Schizophrenia Consortium et al. Rare chromosomal deletions and duplications increase risk of schizophrenia. *Nature*, 455(7210):237, 2008.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[15] Christina Curtis, Andy G Lynch, Mark J Dunning, Inmaculada Spiteri, John C Marioni, James Hadfield, Suet-Feung Chin, James D Brenton, Simon Tavaré, and Carlos Caldas. The pitfalls of platform comparison: Dna copy number array technologies assessed. *BMC Genomics*, 10(1):588, 2009.

[16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Robert D Edwards, WHC Bassetti, and John Magee. *Technical analysis of stock trends.* CRC press, 2007.

[19] Eugene F Fama and Kenneth R French. The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465, 1992.

[20] Yin Fan, Xiangju Lu, Dian Li, and Yuanliu Liu. Video-based emotion recognition using cnn-rnn and c3d hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 445–450, 2016.

[21] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[22] Jennifer L Freeman, George H Perry, Lars Feuk, Richard Redon, Steven A McCarroll, David M Altshuler, Hiroyuki Aburatani, Keith W Jones, Chris Tyler-Smith, Matthew E Hurles, et al. Copy number variation: new insights in genome diversity. *Genome Research*, 16(8):949–961, 2006.

[23] Shangce Gao, Mengchu Zhou, Yirui Wang, Jiujun Cheng, Hanaki Yachi, and Jiahai Wang. Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 30(2):601–614, 2018.

[24] Bruce CN Greenwald, Judd Kahn, Paul D Sonkin, and Michael Van Biema. *Value investing: From graham to buffett and beyond.* John Wiley & Sons, 2004.

[25] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020.

[26] David L Hall. Perspectives on the fusion of image and non-image data. In *32nd Applied Imagery Pattern Recognition Workshop, 2003. Proceedings.*, pages 217–220. IEEE, 2003.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[28] Susan D Hester, Laura Reid, Norma Nowak, Wendell D Jones, Joel S Parker, Kevin Knudtson, William Ward, Jay Tiesman, and Nancy D Denslow. Comparison of comparative genomic hybridization technologies across microarray platforms. *Journal of Biomolecular Techniques*, 20(2):135, 2009.

[29] Tsung-Jung Hsieh, Hsiao-Fen Hsiao, and Wei-Chang Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2510–2525, 2011.

[30] A John Iafrate, Lars Feuk, Miguel N Rivera, Marc L Listewnik, Patricia K Donahoe, Ying Qi, Stephen W Scherer, and Charles Lee. Detection of large-scale variation in the human genome. *Nature Genetics*, 36(9):949–951, 2004.

[31] Adrian S Ishkanian, Chad A Malloff, Spencer K Watson, J deLeeuw Ronald, Bryan Chi, Bradley P Coe, Antoine Snijders, Donna G Albertson, Daniel Pinkel, Marco A Marra, et al. A tiling resolution dna microarray with complete coverage of the human genome. *Nature Genetics*, 36(3):299–303, 2004.

[32] Jiheon Kang, Youn-Jong Park, Jaeho Lee, Soo-Hyun Wang, and Doo-Seop Eom. Novel leakage detection by ensemble cnn-svm and graph-based localization in water distribution systems. *IEEE Transactions on Industrial Electronics*, 65(5):4279–4289, 2017.

[33] Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Michael Chen. Short-term forecasting of passenger demand under on-demand ride services: A spatio-

temporal deep learning approach. *Transportation Research Part C: Emerging Technologies*, 85:591–608, 2017.

[34] Mahdi Khodayar, Saeed Mohammadi, Mohammad E Khodayar, Jianhui Wang, and Guangxi Liu. Convolutional graph autoencoder: a generative deep neural network for probabilistic spatio-temporal solar irradiance forecasting. *IEEE Transactions on Sustainable Energy*, 2019.

[35] Mahdi Khodayar and Jianhui Wang. Spatio-temporal graph deep neural network for short-term wind speed forecasting. *IEEE Transactions on Sustainable Energy*, 10(2):670–681, 2018.

[36] Jeffrey M Kidd, Gregory M Cooper, William F Donahue, Hillary S Hayden, Nick Sampas, Tina Graves, Nancy Hansen, Brian Teague, Can Alkan, Francesca Antonacci, et al. Mapping and sequencing of structural variation from eight human genomes. *Nature*, 453(7191):56–64, 2008.

[37] Jae H Kim, Abul Shamsuddin, and Kian-Ping Lim. Stock return predictability and the adaptive markets hypothesis: Evidence from century-long us data. *Journal of Empirical Finance*, 18(5):868–879, 2011.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv Preprint arXiv:1412.6980*, 2014.

[39] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv Preprint arXiv:1609.02907*, 2016.

[40] Joshua M Korn, Finny G Kuruvilla, Steven A McCarroll, Alec Wysoker, James Nemesh, Simon Cawley, Earl Hubbell, Jim Veitch, Patrick J Collins, Katayoon Darvishi, et al. Integrated genotype calling and association analysis of snps, common copy number polymorphisms and rare cnvs. *Nature Genetics*, 40(10):1253, 2008.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[42] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, 3361(10):1995, 1995.

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[44] Chaolong Li, Zhen Cui, Wenming Zheng, Chunyan Xu, and Jian Yang. Spatio-temporal graph convolution for skeleton based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[45] Andy Liaw, Matthew Wiener, et al. Classification and regression by random-forest. *R News*, 2(3):18–22, 2002.

[46] Steve Lohr. The age of big data. *New York Times, NY, NY*, page 11, 2012.

[47] Anthony W Lynch and Jessica A Wachter. Using samples of unequal length in generalized method of moments estimation. *Journal of Financial and Quantitative Analysis*, 48(1):277–307, 2013.

[48] Anant Madabhushi, Ajay Basavanhally, Scott Doyle, Shannon Agner, and George Lee. Computer-aided prognosis: predicting patient and disease outcome via multi-modal image analysis. In *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1415–1418. IEEE, 2010.

[49] Mario Martin. On-line support vector machine regression. In *European Conference on Machine Learning*, pages 282–294. Springer, 2002.

[50] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 37–45, 2015.

[51] David T Miller, Margaret P Adam, Swaroop Aradhya, Leslie G Biesecker, Arthur R Brothman, Nigel P Carter, Deanna M Church, John A Crolla, Evan E Eichler, Charles J Epstein, et al. Consensus statement: chromosomal microarray is a first-tier clinical diagnostic test for individuals with developmental disabilities or congenital anomalies. *The American Journal of Human Genetics*, 86(5):749–764, 2010.

[52] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[53] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[54] Dennis Olson and Charles Mossman. Neural network forecasts of canadian stock returns using accounting ratios. *International Journal of Forecasting*, 19(3):453–465, 2003.

[55] Andy W Pang, Jeffrey R MacDonald, Dalila Pinto, John Wei, Muhammad A Rafiq, Donald F Conrad, Hansoo Park, Matthew E Hurles, Charles Lee, J Craig Venter, et al. Towards a comprehensive structural variation map of an individual human genome. *Genome Biology*, 11(5):R52, 2010.

[56] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron

Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] Dinh Hoang Bach Phan, Susan Sunila Sharma, and Paresh Kumar Narayan. Stock return forecasting: some new evidence. *International Review of Financial Analysis*, 40:38–51, 2015.

[58] Dalila Pinto, Katayoon Darvishi, Xinghua Shi, Diana Rajan, Diane Rigler, Tom Fitzgerald, Anath C Lionel, Bhooma Thiruvahindrapuram, Jeffrey R MacDonald, Ryan Mills, et al. Comprehensive assessment of array-based platforms and calling algorithms for detection of copy number variants. *Nature Biotechnology*, 29(6):512, 2011.

[59] Roger Pique-Regi, Jordi Monso-Varona, Antonio Ortega, Robert C Seeger, Timothy J Triche, and Shahab Asgharzadeh. Sparse representation and bayesian detection of genome copy number alterations from microarray data. *Bioinformatics*, 24(3):309–318, 2008.

[60] Adin Ramirez Rivera and Oksam Chae. Spatiotemporal directional number transitional graph for dynamic texture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2146–2152, 2015.

[61] Stephen W Scherer, Charles Lee, Ewan Birney, David M Altshuler, Evan E Eichler, Nigel P Carter, Matthew E Hurles, and Lars Feuk. Challenges and standards in integrating surveys of structural variation. *Nature Genetics*, 39(7s):S7–S15, 2007.

[62] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[63] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[65] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[66] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[67] Ilaria Amerise-Agostino Tarsitano and IL Amerise. Adjusting time series of possible unequal lengths. In *Sis. Proceedings of the Xlvi Scientific Meeting*, pages 1–4, 2012.

[68] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[69] Eray Tuzun, Andrew J Sharp, Jeffrey A Bailey, Rajinder Kaul, V Anne Morrison, Lisa M Pertz, Eric Haugen, Hillary Hayden, Donna Albertson, Daniel Pinkel, et al. Fine-scale structural variation of the human genome. *Nature Genetics*, 37(7):727–732, 2005.

[70] Jiajun Wang and Tufan Kumbasar. Parameter optimization of interval type-2 fuzzy neural networks based on pso and bbbc methods. *IEEE/CAA Journal of Automatica Sinica*, 6(1):247–257, 2019.

[71] Kai Wang, Mingyao Li, Dexter Hadley, Rui Liu, Joseph Glessner, Struan FA Grant, Hakon Hakonarson, and Maja Bucan. Penncnv: an integrated hidden markov model designed for high-resolution copy number variation detection in whole-genome snp genotyping data. *Genome Research*, 17(11):1665–1674, 2007.

[72] Laura Winchester, Christopher Yau, and Jiannis Ragoussis. Comparing cnv detection methods for snp arrays. *Briefings in Functional Genomics and Proteomics*, 8(5):353–366, 2009.

[73] Kendy K Wong, Ronald J deLeeuw, Nirpjit S Dosanjh, Lindsey R Kimm, Ze Cheng, Douglas E Horsman, Calum MacAulay, Raymond T Ng, Carolyn J Brown, Evan E Eichler, et al. A comprehensive analysis of common copy-number variations in the human genome. *The American Journal of Human Genetics*, 80(1):91–104, 2007.

[74] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, 2013.

[75] Guo-Sen Xie, Xu-Yao Zhang, Shuicheng Yan, and Cheng-Lin Liu. Hybrid cnn and dictionary-based models for scene recognition and domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1263–1274, 2015.

[76] Tie-Lin Yang, Xiang-Ding Chen, Yan Guo, Shu-Feng Lei, Jin-Tang Wang, Qi Zhou, Feng Pan, Yuan Chen, Zhi-Xin Zhang, Shan-Shan Dong, et al. Genome-wide copy-number-variation study identified a susceptibility gene, ugt2b17, for osteoporosis. *The American Journal of Human Genetics*, 83(6):663–674, 2008.

[77] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *ArXiv Preprint arXiv:1709.04875*, 2017.