Deep Learning Networks for Stock Market Analysis and Prediction: Methodology, Data Representations, and Case Studies

Eunsuk Chong[a], Chulwoo Han[b], and Frank C. Park[*, a]

[a]*Robotics Laboratory, Seoul National University, Seoul 08826, Korea*
[b]*Durham University Business School, Mill Hill Lane, Durham DH1 3LB, UK*

Abstract

We offer a systematic analysis of the use of deep learning networks for stock market analysis and prediction. Its ability to extract features from a large set of raw data without relying on prior knowledge of predictors makes deep learning potentially attractive for stock market prediction at high frequencies. Deep learning algorithms vary considerably in the choice of network structure, activation function, and other model parameters, and their performance is known to depend heavily on the method of data representation. Our study attempts to provides a comprehensive and objective assessment of both the advantages and drawbacks of deep learning algorithms for stock market analysis and prediction. Using high-frequency intraday stock returns as input data, we examine the effects of three unsupervised feature extraction methods—principal component analysis, autoencoder, and the restricted Boltzmann machine—on the network's overall ability to predict future market behavior. Empirical results suggest that deep neural networks can extract additional information from the residuals of the autoregressive model and improve prediction performance; the same cannot be said when the autoregressive model is applied to the residuals of the network. Covariance estimation is also noticeably improved when the predictive network is applied to covariance-based market structure analysis. Our study offers practical insights and potentially useful directions for further investigation into how deep learning networks can be effectively used for stock market analysis and prediction.

KEYWORDS: Stock market prediction; deep learning; multilayer neural network; covariance estimation

1 Introduction

Research on the predictability of stock markets has a long history in financial economics (*e.g.,* Bradley, 1950; Granger & Morgenstern, 1970; Bondt & Thaler, 1985; J. Y. Campbell & Hamao, 1992; Ang & Bekaert, 2007; J. Y. Campbell & Thompson, 2008; Bacchetta, Mertens, & Van Wincoop, 2009; S. D. Campbell, 2012). While opinions differ on the efficiency of markets, many widely accepted empirical studies show that financial markets are to some extent predictable (Ferreira & Santa-Clara, 2011; J. H. Kim, Shamsuddin, & Lim, 2011; Bollerslev, Marrone, Xu, & Zhou, 2014; Phan, Sharma, &

[*]Corresponding author. Tel.: +82 2 8807133.
Email addresses: bear3498@snu.ac.kr (E. Chong), chulwoo.han@durham.ac.uk (C. Han), fcp@snu.ac.kr (F. C. Park).

Narayan, 2015). Among methods for stock return prediction, econometric or statistical methods based on the analysis of past market movements have been the most widely adopted (Agrawal, Chourasia, & Mittra, 2013). These approaches employ various linear and nonlinear methods to predict stock returns, *e.g.,* autoregressive models and artificial neural networks (ANN) (Bogullu, Dagli, & Enke, 2002; A.-S. Chen, Leung, & Daouk, 2003; Thawornwong & Enke, 2004; Armano, Marchesi, & Murru, 2005; Cao, Leggio, & Schniederjans, 2005; Zhu, Wang, Xu, & Li, 2008; Atsalakis & Valavanis, 2009; Tsai & Hsiao, 2010; Guresen, Kayakutlu, & Daim, 2011a; Khashei & Bijari, 2011; Kara, Boyacioglu, & Baykan, 2011; Wang, Wang, Zhang, & Guo, 2011; Yeh, Huang, & Lee, 2011; Kazem, Sharifi, Hussain, Saberi, & Hussain, 2013; Ticknor, 2013; Adebiyi, Adewumi, & Ayo, 2014; Enke & Mehdiyev, 2014; Monfared & Enke, 2014; Rather, Agarwal, & Sastry, 2015; Y. Kim & Enke, 2016a, 2016b). While there is uniform agreement that stock returns behave nonlinearly, many empirical studies show that for the most part nonlinear models do not necessarily outperform linear models: *e.g.*, C. Lee, Sehwan, and Jongdae (2007), K. J. Lee, Chi, Yoo, and Jin (2008), Agrawal et al. (2013), and Adebiyi et al. (2014) propose linear models that outperform or perform as well as nonlinear models, whereas Thawornwong and Enke (2004), Cao et al. (2005), Enke and Mehdiyev (2013), and Rather et al. (2015) find nonliner models outperfrom linear models. Table I provides a summary of recent works relevant to our research. For more exhaustive and detailed reviews, we refer the reader to Atsalakis and Valavanis (2009), Guresen, Kayakutlu, and Daim (2011b), and Cavalcante, Brasileiro, Souza, Nobrega, and Oliveira (2016).

Recently there has been a resurgence of interest in artificial neural networks, in large part to its spectacular successes in image classification, natural language processing, and various time-series problems (Hinton & Salakhutdinov, 2006; H. Lee, Pham, Largman, & Ng, 2009; CireşAn, Meier, Masci, & Schmidhuber, 2012). Underlying this progress is the development of a feature learning framework, known as deep learning (LeCun, Bengio, & Hinton, 2015), whose basic structure is best described as a multi-layer neural network, and whose success can be attributed to a combination of increased computational power, availability of large datasets, and more sophisticated algorithms (Hinton, Osindero, & Teh, 2006; Bengio, Lamblin, Popovici, Larochelle, et al., 2007; Salakhutdinov & Hinton, 2009; Deng & Yu, 2014; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

There has been growing interest in whether deep learning can be effectively applied to problems in finance, but the literature (at least in the public domain) still remains limited.[1] With the increasing availability of high-frequency trading data and the less-than-satisfactory performance of existing models, comprehensive studies that objectively examine the suitability of deep learning to stock market prediction and analysis seem opportune. The ability to extract abstract features from data, and to identify hidden nonlinear relationships without relying on econometric assumptions and human expertise, makes deep learning particularly attractive as an alternative to existing models and approaches.

ANNs require a careful selection of the input variables and network parameters such as the learning rate, number of hidden layers, and number of nodes in each layer in order to achieve satisfactory results (Hussain, Knowles, Lisboa, & El-Deredy, 2008). It is also important to reduce dimensionality to improve learning efficiency. On the other hand, deep learning automatically extracts features from data and requires minimal human intervention during feature selection. Therefore, our approach does not require expertise on predictors such as macroeconomic variables and enables us to use a large set of raw-level data as input. Ignoring the factors that are known to predict the returns, our approach may not be able to

[1]There exist a few studies that apply deep learning to identification of the relationship between past news events and stock market movements (Yoshihara, Fujikawa, Seki, & Uehara, 2014; Ding, Zhang, Liu, & Duan, 2015), but, to our knowledge, there is no study that apply deep learning to extract information from the stock return time series.

outperform existing models based on carefully chosen predictors. However, considering the fast growth of deep learning algorithms, we believe our research will serve as a milestone for the future research in this direction. (Chourmouziadis & Chatzoglou, 2016) also predict that deep learning will play a key role in financial time series forecasting.

We conjecture that, due to correlation, past stock returns affect not only its own future returns but also the future returns of other stocks, and use 380 dimensional lagged stock returns (38 stocks and 10 lagged returns) as input data letting deep learning extract features. This large input dataset makes deep learning a particularly suitable choice for our research.

We test our model on high-frequency data from the Korean stock market. Previous studies have predominantly focused on return prediction at a low frequency, and high frequency return prediction studies haven been rare. However, market microstructure noise can cause temporal market inefficiency and we expect that many profit opportunities can be found at a high frequency. Also, high-frequency trading allows more trading opportunities and makes it possible to achieve statistical arbitrage. Another advantage of using high-frequency data is that we can get a large dataset, which is essential to overcome data-snooping and over-fitting problems inevitable in neural network or any other non-linear models. Whilst it should be possible to train the network within 5 minutes given a reasonable size of training set, we believe daily training should be enough.

Put together, the aim of the paper is to assess the potential of deep feature learning as a tool for stock return prediction, and more broadly for financial market prediction. Viewed as multi-layer neural networks, deep learning algorithms vary considerably in the choice of network structure, activation function, and other model parameters; their performance is also known to depend heavily on the method of data representation. In this study, we offer a systematic and comprehensive analysis of the use of deep learning. In particular, we use stock returns as input data to a standard deep neural network, and examine the effects of three unsupervised feature extraction methods—principal component analysis, autoencoder, and the restricted Boltzmann machine—on the network's overall ability to predict future market behavior. The network's performance is compared against a standard autoregressive model and ANN model for high-frequency intraday data taken from the Korean KOSPI stock market.

The empirical analysis suggests that the predictive performance of deep learning networks is mixed and depends on a wide range of both environmental and user-determined factors. On the other hand, deep feature learning is found to be particularly effective when used to complement an autoregressive model, considerably improving stock return prediction. Moreover, applying this prediction model to covariance-based market structure analysis is also shown to improve covariance estimation effectively. Beyond these findings, the main contributions of our study are to demonstrate how deep feature learning-based stock return prediction models can be constructed and evaluated, and to shed further light on future research directions.

The remainder of the paper is organized as follows. Section 2 describes the framework for stock return prediction adopted in our research, with a brief review of deep neural networks and data representation methods. Section 3 describes the sample data construction process using stock returns from the KOSPI stock market. A simple experiment to show evidence of predictability is also demonstrated in this section. Section 4 proposes several data representations that are used as inputs to the deep neural network, and assesses their predictive power via a stock market trend prediction test. Section 5 is devoted to construction and evaluation of the deep neural network; here we compare our model with a standard autoregressive model, and also test a hybrid model that merges the two models. In Section 6, we apply the results of return predictions to covariance structure analysis, and show that incorporating return pre-

Table I: A summery of recent studies on stock market prediction.

Authors (Year)	**Data Type (Num. of input features ×lagged times)**	**Target Output**	**Num. of Samples (Training: Validation: Test)**	**Sampling Period (Frequency)**	**Method**	**Performance Measure**
Enke and Mehdiyev (2013)	US S&P 500 index (20×1)	stock price	361	Jan-1980 to Jan-2010 (daily)	feature selection +fuzzy clustering +fuzzy NN	RMSE
Niaki and Hoseinzade (2013)	Korea KOSPI200 index (27×1)	market direction (up or down)	3,650 (8:1:1)	1-Mar-1994 to 30-Jun-2008 (daily)	feature selection +ANN	statistical tests
Cervelló-Royo et al. (2015)	US Dow Jones index (1×10)	market trend (bull/bear-flag)	91,307	22-May-2000 to 29-Nov-2013 (15-min)	template matching	trading simulation
Patel et al. (2015)	India CNX and BSE indices (10×1)	stock price	2,393*	Jan-2003 to Dec-2012 (daily)	SVR+ {ANN, RF, SVR}	MAPE, MAE, rRMSE, MSE
T.-l. Chen and Chen (2016)	Taiwan TAIEX[a] and US NASDAQ[b] indices (27×20)	market trend (bull-flag)	3,818[a,*] 3,412[b,*] (7:0:1)	7-Jen-1989 to 24-Mar-2004 (daily)	dimension reduction +template matching	trading simulation
Chiang et al. (2016)	World 22 stock market indices ({3∼5}×1)	trading signal (stock price)	756 (2:0:1)	Jan-2008 to Dec-2010 (daily)	particle swarm optimization +ANN	trading simulation
Chourmouziadis and Chatzoglou (2016)	Greece ASE general index (8×1)	portfolio composition (cash:stock)	3,907*	15-Nov-1996 to 5-Jun-2012 (daily)	fuzzy system	trading simulation
Qiu et al. (2016)	Japan Nikkei 225 index (71×1)	stock return	237 (7:0:3)	Nov-1993 to Jul-2013 (monthly)	ANN +{genetic algorithm, simulated annealing}	MSE
Arévalo et al. (2016)	US Apple stock (3×{2∼15}+2)	stock price	19,109 (17:0:3)	2-Sep-2008 to 7-Nov-2008 (1-minute)	deep NN	MSE, directional accuracy
Zhong and Enke (2017)	US SPDR S&P 500 ETF (SPY) (60×1)	market direction (up or down)	2,518 (14:3:3)	1-Jun-2003 to 31-May-2013 (daily)	dimension reduction +ANN	trading simulation, statistical tests
Our study	Korea KOSPI 38 stock returns (38×10)	stock return	73,041 (3:1:1)	4-Jan-2010 to 30-Dec-2014 (5-minute)	data representation+ deep NN	NMSE, RMSE, MAE, MI

NN: neural network, SVR: support vector regression, RF: random forest, rRMSE: relative RMSE, NMSE: normalized MSE, MI: mutual information.

* In some studies the number of samples is not explicitly provided. We have calculated the number of samples based on each country's business days.

dictions improves estimation of correlations between stocks. Concluding remarks with suggestions for the future research are given in Section 7.

2 Deep Feature Learning for Stock Return Prediction

2.1 The Framework

For each stock, we seek a predictor function f in order to predict the stock return at time $t+1$, r_{t+1}, given the features (**representation**) u_t extracted from the information available at time t. We assume that r_{t+1} can be decomposed into two parts: the predicted output $\hat{r}_{t+1} = f(u_t)$, and the unpredictable part γ, which we regard as Gaussian noise:

$$r_{t+1} = \hat{r}_{t+1} + \gamma, \quad \gamma \sim \mathcal{N}(0, \beta), \tag{1}$$

where $\mathcal{N}(0, \beta)$ denotes a normal distribution with zero mean and variance β. The representation u_t can be either a linear or a nonlinear transformation of the raw level information R_t. Denoting the transformation function by ϕ, we have

$$u_t = \phi(R_t), \tag{2}$$

and

$$\hat{r}_{t+1} = f \circ \phi(R_t). \tag{3}$$

For our research, we define the raw level information as the past returns of the stocks in our sample. If there are M stocks in the sample and g lagged returns are chosen, R_t will have the form

$$R_t = [r_{1,t}, \cdots r_{1,t-g+1}, \cdots, r_{M,t}, \cdots, r_{M,t-g+1}]^T \in \mathbb{R}^{Mg}, \tag{4}$$

where $r_{i,t}$ denotes the return on stock i at time t. In the remainder of this section, we illustrate the construction of the predictor function f using a deep neural network, and the transformation function ϕ using different data representation methods.

2.2 Deep Neural Network

A Neural network specifies the nonlinear relationship between two variables h_l and h_{l+1} through a network function, which typically has the form

$$h_{l+1} = \delta(W h_l + b), \tag{5}$$

where δ is called an **activation** function, and the matrix W and vector b are model parameters. The variables h_l and h_{l+1} are said to form a layer; when there is only one layer between the variables, their relationship is called a single-layer neural network. Multi-layer neural networks augmented with advanced learning methods are generally referred to as deep neural networks (DNN). A DNN for the predictor function, $y = f(u)$, can be constructed by serially stacking the network functions as follows:

$$\begin{aligned} h_1 &= \delta_1(W_1 u + b_1) \\ h_2 &= \delta_2(W_2 h_1 + b_2) \\ &\vdots \\ y &= \delta_L(W_L h_{L-1} + b_L), \end{aligned}$$

where L is the number of layers.

Given a dataset $\{u^n, \tau^n\}_{n=1}^N$ of inputs and targets, and an error function $\mathcal{E}(y^n, \tau^n)$ that measures the difference between the output $y^n = f(u^n)$ and the target τ^n, the model parameters for the entire network, $\theta = \{W_1, \cdots, W_L, b_1, \cdots, b_L\}$, can be chosen so as to minimize the sum of the errors:

$$\min_{\theta} \left[J = \sum_{n=1}^{N} \mathcal{E}(y^n, \tau^n) \right]. \tag{6}$$

Given an appropriate choice of $\mathcal{E}(\cdot)$, its gradient can be obtained analytically through error backpropagation (Bishop, 2006). In this case, the minimization problem in (6) can be solved by the usual gradient descent method. A typical choice for the objective function that we adopt in this paper has the form:

$$J = \frac{1}{N} \sum_{n=1}^{N} \|y^n - \tau^n\|^2 + \lambda \cdot \sum_{l=1}^{L} \|W_l\|_2, \tag{7}$$

where $\|\cdot\|$ and $\|\cdot\|_2$ respectively denote the Euclidean norm and the matrix L_2 norm. The second term is a "regularizer" added to avoid overfitting, while λ is a user-defined coefficient.

2.3 Data Representation Methods

Transforming (representing) raw data before inputing it into a machine learning task often enhances the performance of the task. It is generally accepted that the performance of a machine learning algorithm depends heavily on the choice of the data representation method (Bengio, Courville, & Vincent, 2013; Längkvist, Karlsson, & Loutfi, 2014). There are various data representation methods such as zero-to-one scaling, standardization, log-scaling, and principal component analysis (PCA) (Atsalakis & Valavanis, 2009). More recently, nonlinear methods became popular: Zhong and Enke (2017) compare PCA and two of its nonlinear variants; fuzzy robust PCA and kernel-based PCA. In this paper, we consider three unsupervised data representation methods: PCA, the autoencoder (AE), and the restricted Boltzmann machine (RBM). PCA involves a linear transformation while the other two are nonlinear transformations. These representation methods are widely used in, *e.g.*, image data classification (Coates, Lee, & Ng, 2010).

For raw data x and a transformation function ϕ, $u = \phi(x)$ is called a representation of x, while each element of u is called a feature. In some cases we can also define a reverse map, $\psi : u \to x$, and retrieve x from u; in this case the retrieved value, $x_{rec} = \psi(u)$, is called a reconstruction of x. The functions ϕ and ψ can be learned from x and x_{rec} by minimizing the difference (reconstruction error) between them. We now briefly describe each method below.

(i) Principal Component Analysis, PCA

In PCA, the representation $u \in \mathbb{R}^d$ is generated from the raw data $x \in \mathbb{R}^D$ via a linear transformation $u = \phi(x) = Wx + b$, where $W \in \mathbb{R}^{d \times D}$, $WW^T = I$, and $b \in \mathbb{R}^d$. The rows of W are the eigenvectors of the first d largest eigenvalues, and b is usually set to $-WE[x]$ so that $E[u] = 0$. Reconstruction of the original data from the representation is given by $x_{rec} = \psi(u) = W^T(u - b)$.

(ii) Autoencoder, AE

AE is a neural network model characterized by a structure in which the model parameters are calibrated by minimizing the reconstruction error. Let $h_l = \delta_l(W_l h_{l-1} + b_l)$ be the network function of the lth layer with input h_{l-1} and output h_l. Although δ_l can differ across layers, a sigmoid function, $\delta(z) = 1/(1 + \exp(-z))$, is typically used for all layers, which we also adopt in our research.[2] Regarding h_l as a function of the input, the representation of x can be written as $u = \phi(x) = h_L \circ \cdots \circ h_1(x)$ for an L-layer AE ($h_0 = x$). Then the reconstruction of the data can be similarly defined: $x_{rec} = \psi(u) = h_{2L} \circ \cdots \circ h_{L+1}(u)$, and the model can be calibrated by minimizing the reconstruction error over a training dataset, $\{x^n\}_{n=1}^N$. We adopt the following learning criterion:

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} ||x^n - \psi \circ \phi(x^n)||^2, \tag{8}$$

where $\theta = \{W_i, b_i\}$, $i = 1, ..., 2L$. W_{L+i} is often set as the transpose of W_{L+1-i}, in which case only $W_i,\ i = 1, \cdots, L$, need to be estimated. In this paper, we consider a single-layer AE and estimate both W_1 and W_2.

(iii) Restricted Boltzmann Machine, RBM

RBM (Hinton, 2002) has the same network structure as a single-layer autoencoder, but it uses a different learning method. RBM treats the input and output variables, x and u, random, and defines an **energy function**, $\mathbb{E}(x, u)$, from which the joint probability density function of x and u is determined from the formula

$$p(x, u) = \frac{\exp(-\mathbb{E}(x, u))}{Z}, \tag{9}$$

where $Z = \sum_{x,u} \exp(-\mathbb{E}(x, u))$ is the partition function. In most cases, u is assumed to be a d-dimensional binary variable, *i.e.,* $u \in \{0, 1\}^d$, and x is assumed to be either binary or real-valued. When x is a real-valued variable, the energy function has the following form (Cho, Ilin, & Raiko, 2011):

$$\mathbb{E}(x, u) = \frac{1}{2}(x - b)^T \Sigma^{-1}(x - b) - c^T u - u^T W \Sigma^{-1/2} x\,, \tag{10}$$

where Σ, W, b, c are model parameters. We set Σ to be the identity matrix; this makes learning simpler with little performance sacrifice (Taylor, Hinton, & Roweis, 2006). From Equations (9) and (10), the conditional distributions are obtained as follows:

$$p(u_j = 1|x) = \delta(c_j + W_{(j,:)} x), \quad j = 1, \cdots, d, \tag{11}$$

$$p(x_i|u) = \mathcal{N}(b_i + u_i^T W_{(:,i)}, 1), \quad i = 1, \cdots, D, \tag{12}$$

where $\delta(\cdot)$ is the sigmoid function, and $W_{(j,:)}$ and $W_{(:,i)}$ are the jth row and the ith column of W, respectively. This type of RBM is denoted the Gaussian-Bernoulli RBM. The input data is then represented and reconstructed in a probabilistic way using the conditional distributions. Given an input dataset $\{x^n\}_{n=1}^N$, maximum log-likelihood learning is formulated as the following optimization:

$$\max_{\theta} \left[L = \sum_{n=1}^{N} \log p(x^n; \theta) \right], \tag{13}$$

[2] $\exp(z)$ is applied to each element of z.

where $\theta = \{W, b, c\}$ are the model parameters, and u is marginalized out (*i.e.,* integrated out via expectation). This problem can be solved via standard gradient descent. However, due to the computationally intractable partition function Z, an analytic formula for the gradient is usually unavailable. The model parameters are instead estimated using a learning method called the contrastive divergence (CD) method (Carreira-Perpinan & Hinton, 2005); we refer the reader to Hinton (2002) for details on learning with RBM.

3 Data Specification

We construct a deep neural network using stock returns from the KOSPI market, the major stock market in South Korea. We first choose the fifty largest stocks in terms of market capitalization at the beginning of the sample period, and keep only the stocks which have a price record over the entire sample period. This leaves 38 stocks in the sample, which are listed in Table II. The stock prices are collected every five minutes during the trading hours of the sample period (04-Jan-2010 to 30-Dec-2014), and five-minute logarithmic returns are calculated using the formula $r_t = \ln(S_t/S_{t-\Delta t})$, where S_t is the stock price at time t, and Δt is five minutes. We only consider intraday prediction, *i.e.,* the first ten five-minute returns (*i.e.,* lagged returns with $g = 10$) each day are used only to construct the raw level input R_t, and not included in the target data. The sample contains a total of 1,239 trading days and 73,041 five-minute returns (excluding the first ten returns each day) for each stock.

The training set consists of the first 80% of the sample (from 04-Jan-2010 to 24-Dec-2013) which contains 58,421 (N_1) stock returns, while the remaining 20% (from 26-Dec-2013 to 30-Dec-2014) with 14,620 (N_2) returns is used as the test set:

$$\text{Training set: } \{R_t^n, r_{i,t+1}^n\}_{n=1}^{N_1}, \quad \text{Test set: } \{R_t^n, r_{i,t+1}^n\}_{n=1}^{N_2}, \quad i = 1, \cdots, M.$$

To avoid over-fitting during training, the last 20% of the training set is further separated as a validation set.

All stock returns are normalized using the training set mean and standard deviation, *i.e.*, for the mean μ_i and the standard deviation σ_i of $r_{i,t}$ over the training set, the normalized return is defined as $(r_{i,t} - \mu_i)/\sigma_i$. Henceforth, for notational convenience we will use $r_{i,t}$ to denote the normalized return.

At each time t, we use ten lagged returns of the stocks in the sample to construct the raw level input:

$$R_t = [r_{1,t}, \cdots, r_{1,t-9}, \cdots, r_{38,t}, \cdots, r_{38,t-9}]^T.$$

3.1 Evidence of Predictability in the Korean Stock Market

As a motivating example, we carry out a simple experiment to see whether past returns have predictable power for future returns. We first divide the returns of each stock into two groups according to the mean or variance of ten lagged returns: If the mean of the lagged returns, $M(10)$, is greater than some threshold η, the return is assigned to one group; otherwise, it is assigned to the other group. Similarly, by comparing the variance of the lagged returns, $V(10)$, with a threshold ϵ, the returns are divided into two groups.

Table II: The sample stocks. They are chosen from the Korean stock market by the size.

Stock ID	Listed Name	Abbreviation	Stock ID	Listed Name	Abbreviation
1	KiaMtr	Kia Mtr	20	SKTelecom	SKTelecom
2	SK hynix	SK hynix	21	LOTTE SHOPPING	LT Shop
3	HyundaiEng&Const	HD Const	22	IBK	IBK
4	SAMSUNG C&T	SS C&T	23	SAMSUNG CARD	SS CARD
5	LG Corp.	LG Corp	24	KT	KT
6	HYUNDAI STEEL	HD STEEL	25	KT&G	KT&G
7	HyundaiMtr	HD Mtr	26	DHICO	DHICO
8	POSCO	POSCO	27	LG Display	LG Disp
9	SamsungElec	SS Elec	28	SKC&C	SKC&C
10	SAMSUNG SDI CO.,LTD.	SS SDI	29	Kangwonland	KW land
11	SamsungElecMech	SS ELMech	30	NCsoft	NCsoft
12	HHI	HHI	31	DSME	DSME
13	OCI	OCI	32	ShinhanGroup	Shinhan
14	KorZinc	KorZinc	33	LGELECTRONICS	LG Elec
15	SamsungHvyInd	SS HvyInd	34	HYUNDAIGLOVIS	HD Glovis
16	S-Oil	S-Oil	35	HANAFINANCIALGR	HANA Fin
17	LOTTE CHEMICAL Corp	LT Chemi	36	AMOREPACIFIC	AMORE
18	Mobis	Mobis	37	SK Innovation	SK Innov
19	KEPCO	KEPCO	38	KBFinancialGroup	KB Fin

The classification is carried out in the training set, and the thresholds η and ϵ are chosen for each stock so that the two groups have the same size. These thresholds are then applied to the test set to classify the returns, and the mean and variance of each group are calculated.

The results are shown in Table III and Figure 1. From the first graph in the figure, it can be clearly seen that the average return of the $M(10) > \eta$ group is lower than that of the other group, for all stocks except AMORE (36). The difference between the two groups is very large for some stocks, *e.g.*, SAMSUNG SDI (10), HHI (12), and IBK (22). More specifically, the mean difference between two groups is significant at 99% confidence level for all stocks except AMORE, which is significant at 95% level. Similar results are obtained for the variance-based classification. The variance difference is significant for all stocks at 99% confidence level. The difference between the two groups may not be large enough for many stocks to make a profit after transaction costs are accounted for. Nevertheless, this example suggests that past returns do have predictive power to a certain degree, which can be further exploited by deep feature learning.

Table III: Mean and variance of stock returns in each group in the test set. The second and the third columns are mean returns of each group and the fourth column is the p-value of t-test on the mean difference. The last three columns are variances of the returns and the p-value of F-test on the variance difference. The details of the grouping method can be found in Section 3.1.

Stock ID	M(10)$> \eta$	M(10)$< \eta$	**p-val**	V(10)$> \epsilon$	V(10)$< \epsilon$	**p-val**
1	-0.079	0.095	0.000	1.069	0.657	0.000
2	-0.023	0.035	0.000	0.915	0.385	0.000
3	-0.049	0.053	0.000	0.956	0.560	0.000
4	-0.038	0.053	0.000	1.113	0.556	0.000
5	-0.073	0.076	0.000	1.071	0.613	0.000
6	-0.024	0.034	0.000	0.627	0.351	0.000
7	-0.124	0.124	0.000	0.935	0.564	0.000
8	-0.099	0.114	0.000	1.117	0.803	0.000
9	-0.041	0.009	0.000	1.018	0.467	0.000
10	-0.154	0.188	0.000	1.141	0.771	0.000
11	-0.048	0.035	0.000	0.953	0.359	0.000
12	-0.178	0.187	0.000	1.548	0.838	0.000
13	-0.093	0.106	0.000	1.005	0.626	0.000
14	-0.007	0.041	0.000	0.693	0.426	0.000
15	-0.065	0.045	0.000	1.111	0.536	0.000
16	-0.044	0.025	0.000	0.729	0.274	0.000
17	-0.068	0.081	0.000	0.997	0.640	0.000
18	-0.068	0.071	0.000	1.026	0.546	0.000
19	-0.029	0.020	0.003	1.529	0.643	0.000
20	-0.092	0.055	0.000	0.990	0.655	0.000
21	-0.091	0.038	0.000	0.878	0.560	0.000
22	-0.164	0.224	0.000	1.054	0.829	0.000
23	-0.042	0.048	0.000	1.204	0.524	0.000
24	-0.124	0.082	0.000	1.176	0.835	0.000
25	-0.067	0.070	0.000	1.279	0.541	0.000
26	-0.023	0.039	0.000	1.168	0.562	0.000
27	-0.026	0.063	0.000	0.846	0.528	0.000
28	-0.080	0.090	0.000	1.066	0.744	0.000
29	-0.065	0.038	0.000	1.067	0.601	0.000
30	-0.095	0.107	0.000	1.137	0.729	0.000
31	-0.037	0.021	0.000	1.209	0.527	0.000
32	-0.086	0.073	0.000	1.068	0.486	0.000
33	-0.048	0.047	0.000	0.607	0.355	0.000
34	-0.009	0.031	0.002	0.946	0.547	0.000
35	-0.067	0.028	0.000	0.731	0.457	0.000
36	0.020	-0.018	0.033	1.594	0.575	0.000
37	-0.113	0.160	0.000	1.028	0.542	0.000
38	-0.069	0.056	0.000	0.942	0.521	0.000

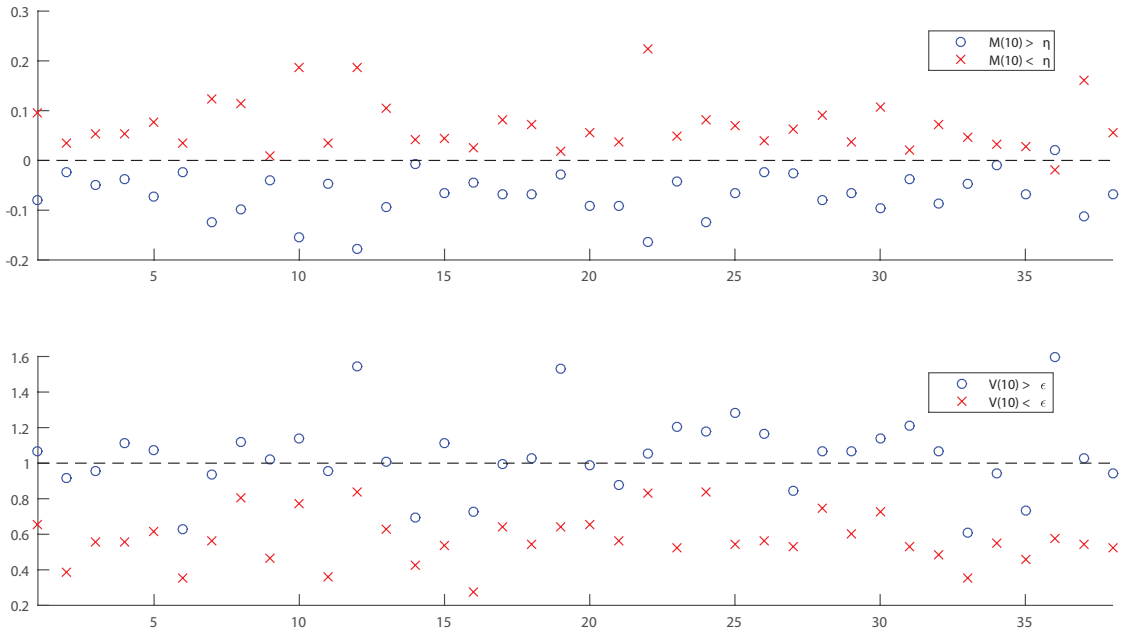


Figure 1: Mean and variance of stock returns in each group in the test set. The upper graph displays the mean returns of the stocks in each group defined by the mean of the past returns. The lower graph displays the variance of the returns in each group defined by the variance of the past returns. x-axis represents the stock ID. The details of the grouping method can be found in Section 3.1.

4 Market Representation and Trend Prediction

In this section, we compare the representation methods described in Section 2.3, and analyze their effects on stock return prediction. Each representation method receives the raw level market movements $R_t \in \mathbb{R}^{380}$ as the input, and produces the representation output (features), $u_t = \phi(R_t)$. Several sizes of u_t are considered: 200 and 380 for PCA, and 400 and 800 for both RBM and AE. These representations are respectively denoted by PCA200, PCA380, RBM400, RBM800, AE400, and AE800 throughout the paper. For comparison, the raw level data R_t, denote by RawData, is also considered as a representation. Therefore, there are total 7 representations.

4.1 Stock Market Trend Prediction via Logistic Regression

Before training DNN using the features obtained from the representation methods, we perform a test to assess the predictive power of the features. The test is designed to check whether the features can predict the up/down movement of the future return. This up/down prediction problem is essentially a two-class classification problem, which can be solved by, among others, the logistic regression method. Let $y \in \{-1, 1\}$ denote the class label with $y = 1$ (-1) representing up (down) movement of r_{t+1}. The

probability of y conditional on the features u_t is defined as

$$P(y|u_t; w, b) = \frac{1}{1 + \exp(-(w^T u_t + b)y)}, \tag{14}$$

where w and b are model parameters. We label the returns r_{t+1} in the training set according to the rule: $y = 1$, if $r_{t+1} > \epsilon$, and $y = -1$, if $r_{t+1} < -\epsilon$ for some ϵ (we set ϵ to 0.02). The regression parameters are then estimated from the input-target pairs in the training set, $\{u_t^n, y^n\}_{n=1}^{N_1}$, by the maximum likelihood estimation:

$$\max_{\{w,b\}} \left[L = -\sum_{n=1}^{N_1} \log\left(1 + \exp(-(w^T u_t^n + b)y^n)\right) \right]. \tag{15}$$

For each representation, we compute the classification accuracy defined as the percentage of correct prediction of ups and downs, and compare with a reference accuracy. The reference accuracy is defined as the percentage of the occurrences of ups or downs whichever is bigger in the test set. For example, if there are 5,000 ups and 4,000 downs, the reference accuracy becomes 5,000/(5,000+4,000). This can be interpreted as having a prior knowledge about which one has a higher probability and betting on that side every time.

The classification results of each representation are reported in Figure 2 through Figure 5, and the representations are compared with one another in Table IV. In each figure, the classification accuracies both from the training set and test set are compared with the reference accuracy obtained from the test set. The results show that the classification accuracy ranges around 55%-70% depending on the representations and the stocks, and is higher than the reference accuracy for most stocks except a few cases where the reference accuracy is very high, *e.g.*, LOTTE SHOPPING (21). This is an encouraging result considering the fact that the reference accuracy assumes a prior knowledge about the probability. The prediction accuracy is rather consistent across stocks and does not deteriorate much in the test set.

The best performing representation varies across stocks. Nonetheless, the untransformed RawData representation appears to perform best on average, followed closely by PCA380. This suggests that the past returns of the stocks are related to the future return dominantly in a linear fashion. On the other extreme, AE400 performs worst, followed by PCA200. These two representations still outperform the reference accuracy. Within each representation method, higher classification accuracy is associated with low reconstruction error, which comes from higher dimensional representation. However, no particular relationship between reconstruction error and classification accuracy can be found across representation methods.

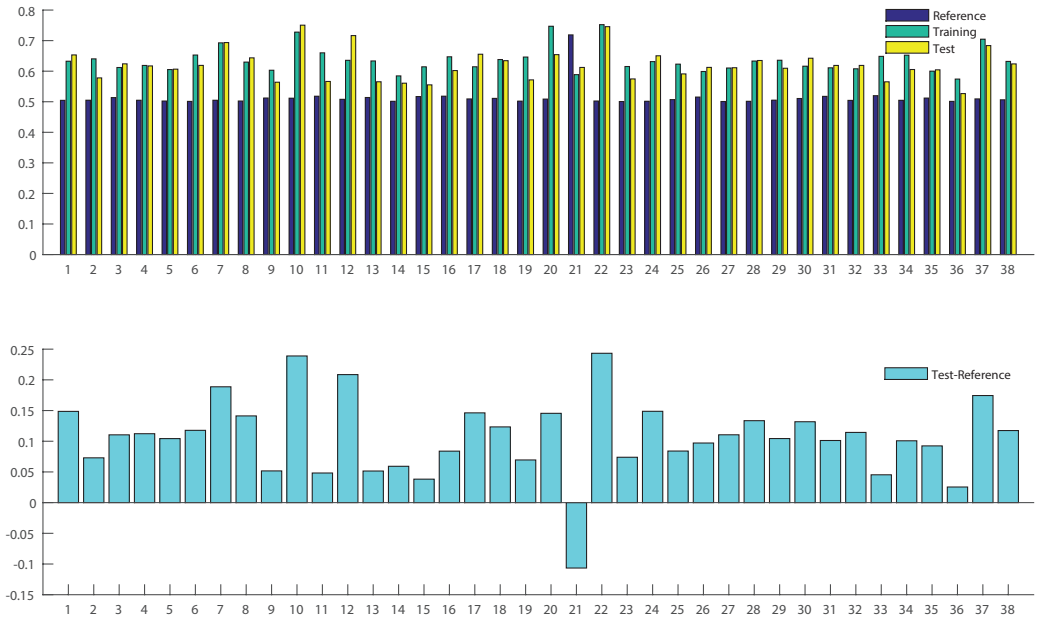
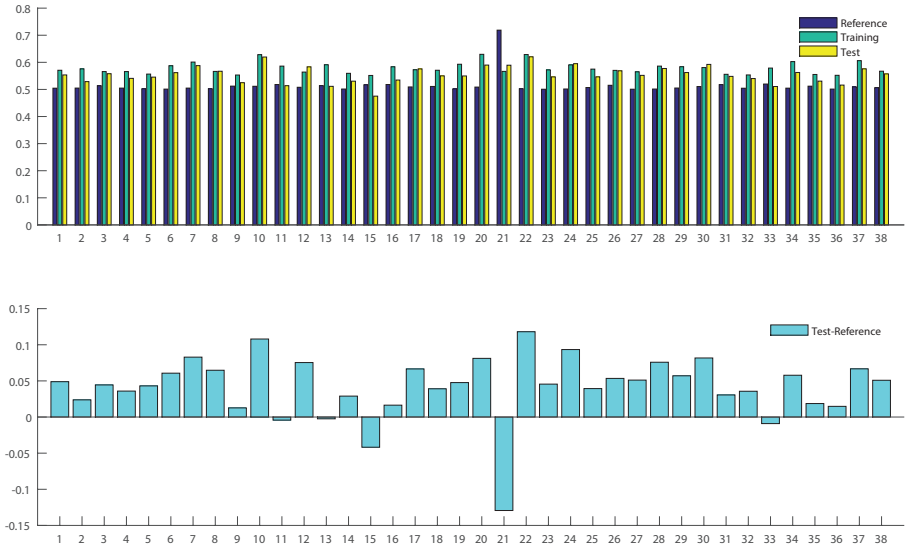


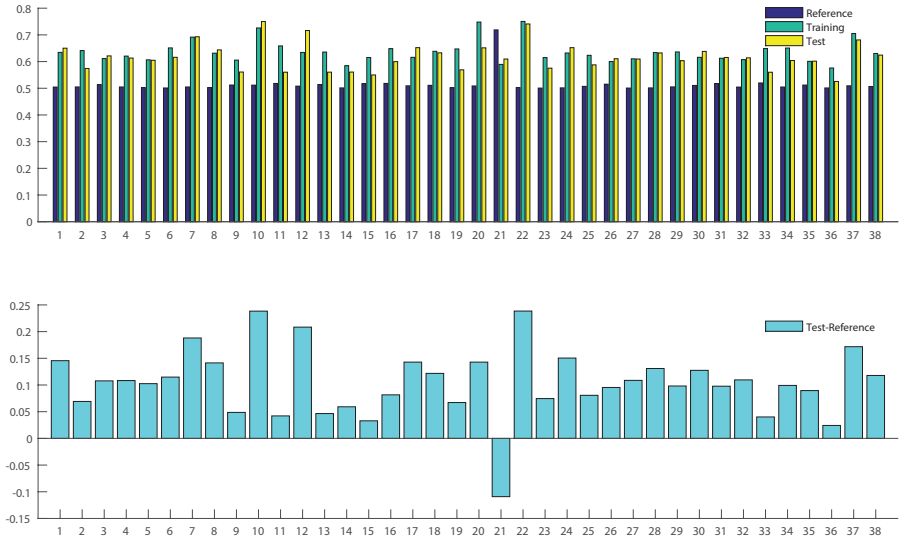
Figure 2: Up/down prediction accuracy of RawData. Upper graph: prediction accuracies in each dataset, and the reference accuracies. Lower graph: The difference between the test set accuracy and the reference accuracy. x-axis represents the stock ID.

Table IV: Summary of the up/down prediction accuracy test. The figures in the last four columns are the average values over the 38 stocks.

Representation	**Number of Features**	**Reconstruction Error:** $\frac{1}{N}\sum_{n=1}^{N}\|x^n - x^n_{rec}\|^2$		**Up/Down Prediction Accuracy**	
		Training Set	**Test Set**	**Training Set**	**Test Set**
RawData	380	0	0	0.6361	0.6201
PCA200	200	109.70	103.17	0.5780	0.5551
PCA380	380	0	0	0.6364	0.6175
RBM400	400	100.32	77.06	0.6079	0.5843
RBM800	800	70.20	60.65	0.6169	0.5928
AE400	400	1.44	1.35	0.5712	0.5341
AE800	800	0.55	0.67	0.6060	0.5692
			Reference	0.5113	0.5134

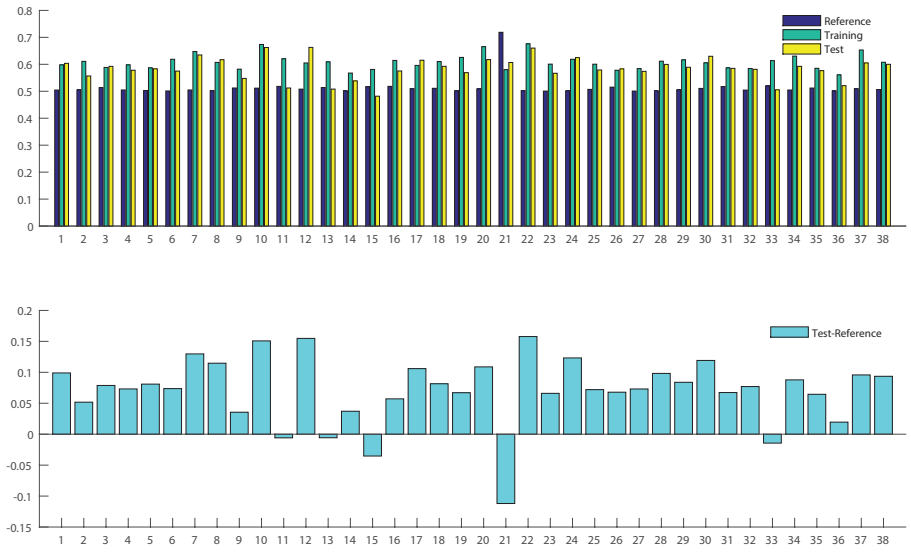


(a) PCA200

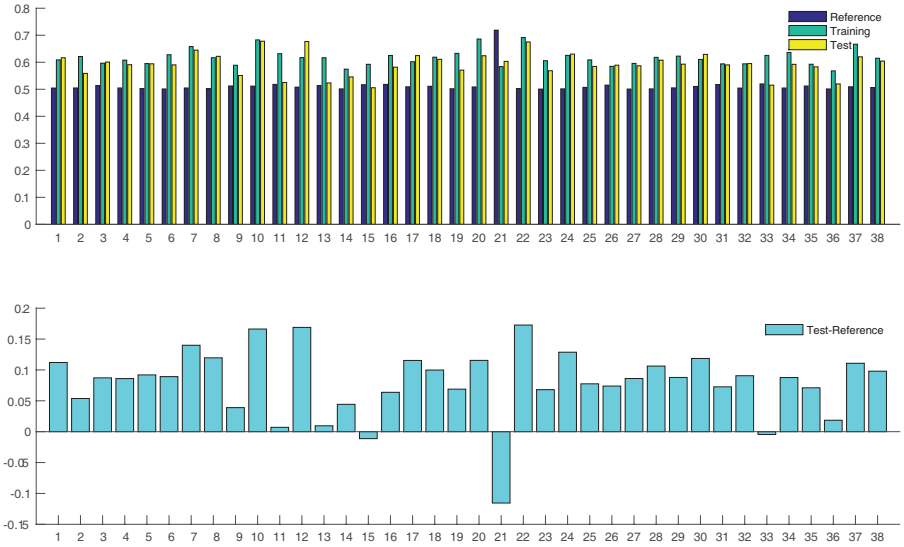


(b) PCA380

Figure 3: Up/down prediction accuracy of PCA200 and PCA380. Upper graph: prediction accuracies in each dataset, and the reference accuracies. Lower graph: The difference between the test set accuracy and the reference accuracy. x-axis represents the stock ID.

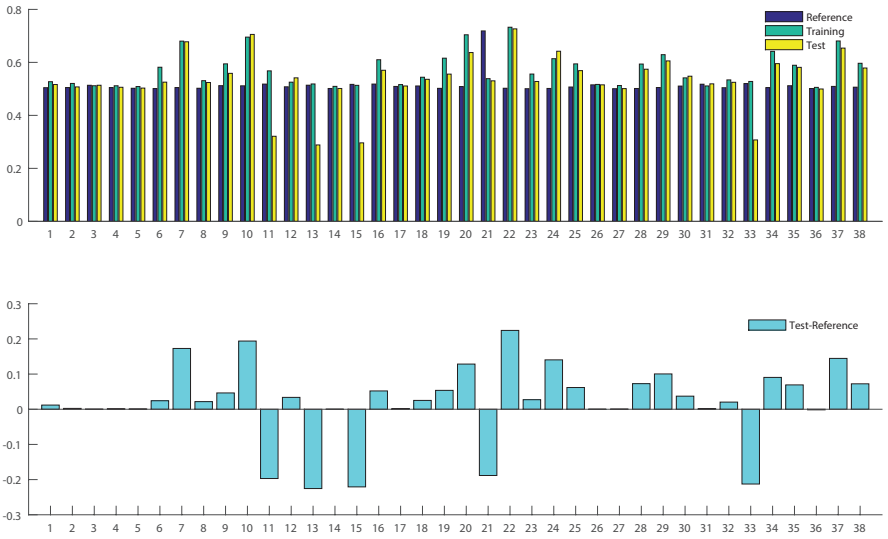


(a) RBM400

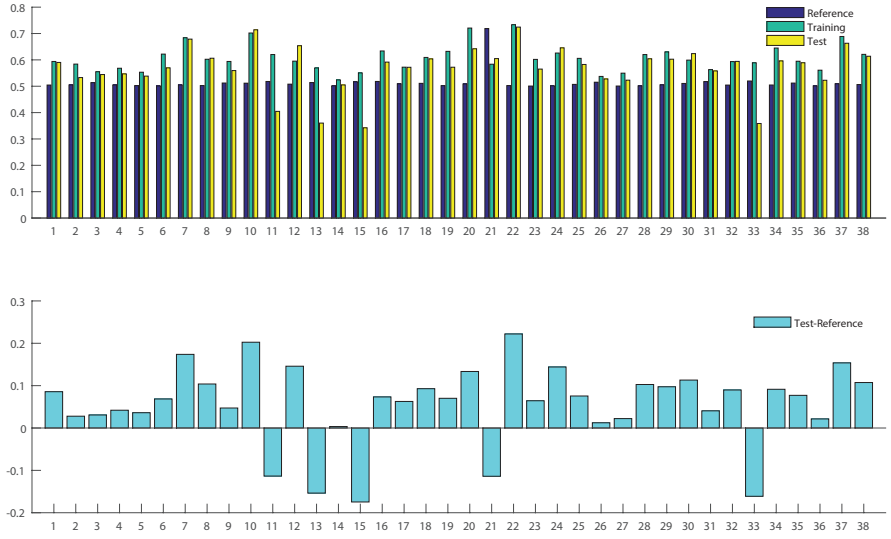


(b) RBM800

Figure 4: Up/down prediction accuracy of RBM400 and RBM800. Upper graph: prediction accuracies in each dataset, and the reference accuracies. Lower graph: The difference between the test set accuracy and the reference accuracy. x-axis represents the stock ID.



(a) AE400



(b) AE800

Figure 5: Up/down prediction accuracy of AE400 and AE800. Upper graph: prediction accuracies in each dataset, and the reference accuracies. Lower graph: The difference between the test set accuracy and the reference accuracy. x-axis represents the stock ID.

5 Training Deep Neural Networks

In this section, we construct the predictor functions, $\hat{r}_{i,t+1} = f_i(u_t),\ i = 1, \cdots, M$, using DNN, and compare their prediction performance with a univariate autoregressive model with ten lagged variables (AR(10)). We employ a three-layer network model of the form[3]:

$$\begin{aligned} h_1 &= ReLU(W_1 u_t + b_1) \\ h_2 &= ReLU(W_2 h_1 + b_2) \\ \hat{r}_{i,t+1} &= W_3 h_2 + b_3, \end{aligned} \tag{16}$$

where $ReLU$ is the rectified linear unit activation function defined as $ReLU(x) = \max(x, 0)$, with $\max$ being an element-wise operator. $ReLU$ is known to provide a much faster learning speed than standard sigmoid units, while maintaining or even improving the performance when applied to deep neural networks (Nair & Hinton, 2010). Nevertheless, we also test a standard ANN for comparison: we replace $ReLU$ with a sigmoid unit and keep the rest of the settings and parameters equal.

With four sets of features, RawData, PCA380, AE400, and RBM400, as the input, the DNN is trained minimizing the objective function defined in Equation (7) with 3,000 learning iterations (epochs), and the regularizer coefficient, $\lambda = 0.001$.[4] We choose 200 and 100 dimensions for the hidden variables h_1 and h_2, respectively. The last 20% of the training set is used as a validation set for early stopping to avoid overfitting. The training was conducted using Theano 0.7 in Python 3.5 running on a PC with an Intel Core i7-6700 CPU at 3.40GHz, and a GeForce GTX TITAN X GPU. The entire training procedure for the four representations consumed 14 hours in this experiment setting.

5.1 Performance Evaluation

We evaluate prediction performance using four measures: normalized mean squared error (NMSE), root mean squared error (RMSE), mean absolute error (MAE), and mutual information (MI).

Normalized Mean Squared Error, NMSE

Given a set of target returns and their predicted values, $\{r^n_{t+1}, \hat{r}^n_{t+1}\}_{n=1}^N$, NMSE is defined as

$$\text{NMSE} = \frac{1}{N} \frac{\sum_{n=1}^N \left(r^n_{t+1} - \hat{r}^n_{t+1}\right)^2}{var(r^n_{t+1})} \tag{17}$$

where $var(\cdot)$ denotes variance. Recall that $var(r^n_{t+1}) = \min_c \frac{1}{N} \sum_{n=1}^N (r^n_{t+1} - c)^2$; NMSE is a mean squared error (MSE) normalized by the least MSE obtained from a constant prediction.

[3] More than three layers did not improve the predictability and the results are not reported here.

[4] We also tested higher dimensional representation methods but the results were similar to their lower dimensional counterparts.

Root Mean Squared Error, RMSE

RMSE is the square root of MSE, defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(r_{t+1}^{n} - \hat{r}_{t+1}^{n}\right)^2} \tag{18}$$

Mean Absolute Error, MAE

MAE is defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}|r_{t+1}^{n} - \hat{r}_{t+1}^{n}| \tag{19}$$

Note that inequality holds for the last two measures, MAE $\leq$ RMSE $\leq \sqrt{N}$MAE; both error measures are known to informative, *e.g.,* while MAE gives the same weight to all error amounts, RMSE is more sensitive to outliers, and is more suit for normal distributed error. For more interpretations, see Chai and Draxler (2014).

Mutual Information, MI

MI measures dependency between r_{t+1} and u_t, and is defined as follows:

$$\begin{aligned}\text{MI}(r_{t+1}; u_t) &= \sum_{r_{t+1},u_t} p(r_{t+1}, u_t)\log\frac{p(r_{t+1}, u_t)}{p(r_{t+1})\,p(u_t)} \\ &\approx \frac{1}{N}\sum_{n=1}^{N}\log\frac{p(r_{t+1}^{n} \mid u_t^{n})}{p(r_{t+1}^{n})}.\end{aligned} \tag{20}$$

$\text{MI}(r_{t+1}; u_t)$ is zero when the two variables are independent, and bounded to the information entropy, $H(r_{t+1}) = -\sum_{r_{t+1}} p(r_{t+1})\log p(r_{t+1})$, when the two variables are fully dependent. From the assumption made earlier, we have $r_{t+1} \mid u_t \sim \mathcal{N}(\hat{r}_{t+1}, \beta)$. With an additional assumption, $r_{t+1} \sim \mathcal{N}(\mu, \sigma)$, we estimate the parameters β, μ and σ from the sample, and evaluate MI from (20).

5.2 Prediction Results

Table V and VI report NMSE in both training and test set, and Table VII reports MAE in the test set.[5] In the training set, all the prediction methods achieve NMSEs smaller than one, implying that these methods are more accurate than any constant prediction. DNN outperforms AR(10) in the training set, regardless of the choice of the features, but the advantage mostly disappears when applied to the test set: Only the DNNs with RawData and AE400 slightly outperform AR(10) in the test set: the DNN with RawData outperforms AR(10) in 16 stocks and the DNN with AE400 outperforms AR(10) in 14 stocks at 5% significantly level.[6] Interestingly, AR(10) achieves a similar level of performance in both sets, whereas the performance of DNN deteriorates in the test set to the level of the performance of AR(10).

[5] We only report performance measures to save the space. Parameter estimation results are available upon request.

[6] Significance is determined using the standard t-test on the difference of the squared predictions errors.

ANN performs worst in terms of RMSE and MI, and best in terms of MAE: ANN outperforms AR(10) in at most 3 stocks at 5% significance level. This is perhaps because of the slow learning speed of the sigmoid function; parameter estimates react slowly to new data and remain close to the initial values. This results in the poor performance of ANN in terms of RMSE, which penalizes large errors more. The performance of ANN could be improved by changing training parameters, but this is beyond the scope of this study and is not pursued further.

Since the unpredictable part in Equation (1) is assumed to be normally distributed and our objective function is to minimize squared error, MSE-based measures (NMSE and RMSE) are a more appropriate performance measure, and from this perspective, DNN using $ReLU$ outperforms ANN for the large and high dimensional stock market dataset.

Next, we investigate whether the predictability can be enhanced by combining DNN with AR(10). We apply DNN to the residuals of AR(10), and examine if the prediction errors are reduced. We repeat the same experiment in the opposite direction: apply AR(10) to the residuals of DNN. We call these methods AR-DNN and DNN-AR, respectively. The results of AR-DNN and DNN-AR for individual stocks are reported in Table VIII and IX, respectively, and summary reports are provided in Table X to XIII.

When DNN is applied to the residuals of AR(10), NMSE is slightly reduced. The effect is small but consistent across the features. Larger reductions are associated with RawData and AE400. These representations are also the representations with smaller errors when applied alone. Applying AR(10) to the residuals of DNN, on the other hand, does not improve the prediction performance in all representations. For instance, the DNN with RawData further reduces the prediction error of AR(10) in 20 stocks, whereas AR(10) further reduces the prediction error of the DNN with RawData only in 7 stocks, both at 5% significance level.

5.3 Sensitivity

We compute the gradient of each predictor function with respect to the input features to identify the contribution of each feature to the prediction. We consider only the RawData representation as our interest is to identify the informational contents of past returns. Denoting the first-order derivative of the predictor function of stock i with respect to the jth feature of RawData $J_{ij} = \partial f_i / \partial u_{j,t}$, we compute a sensitivity matrix $S \in \mathbb{R}^{M \times Mg}$ from the test set as follows:

$$S_{ij} = \frac{1}{N_2} \sum_{n=1}^{N_2} \left| J_{ij}^n \right|. \tag{21}$$

where J_{ij}^n is J_{ij} of the nth data in the test set. The sensitivities of DNN and AR-DNN are visualized in Figure 6. White color represents zero sensitivity and a darker color means a higher sensitivity. It is interesting to note that, for DNN, the sensitivity of f_i is particularly high with respect to $u_{j,t}$ for $j = i, i+M$, and $i+2M$. This implies that only the first three lagged returns of its own are the dominant features that determine the predicted value. This explains why DNN hardly outperforms AR(10) in our test. In fact, AR estimation results, which are not reported in this paper, also show that the first three lagged returns are the only significant variables for many stocks. On the other hand, the sensitivity of AR-DNN illustrated in Figure 6-(b) does not show any discernible pattern. Nevertheless, the dark regions in the figure indicate that the residuals from AR(10) contain information for the future returns.

Table V: Normalized mean squared error in the training set (04-Jan-2010∼24-Dec-2013).

Stock ID	**AR(10)**	**ANN (RawData)**	**DNN (RawData)**	**DNN (PCA380)**	**DNN (RBM400)**	**DNN (AE400)**
1	0.9810	1.0000	0.9246	0.9336	0.9401	0.9241
2	0.9746	0.9999	0.9314	0.9370	0.9453	0.9343
3	0.9908	0.9985	0.9379	0.9444	0.9479	0.9429
4	0.9816	1.0000	0.9234	0.9244	0.9304	0.9224
5	0.9817	1.0000	0.9431	0.9511	0.9546	0.9452
6	0.9396	0.9674	0.9436	0.9481	0.9490	0.9528
7	0.9288	0.9798	0.8854	0.8952	0.9122	0.8849
8	0.9618	1.0000	0.9198	0.9189	0.9228	0.9184
9	0.9750	1.0000	0.9671	0.9654	0.9661	0.9622
10	0.9164	0.9797	0.8537	0.8669	0.8699	0.8560
11	0.9322	0.9714	0.9389	0.9394	0.9458	0.9400
12	0.9816	0.9999	0.9010	0.9095	0.9165	0.9076
13	0.9771	0.9999	0.8975	0.9061	0.9143	0.8988
14	0.9929	1.0000	0.9644	0.9655	0.9676	0.9672
15	0.9902	1.0000	0.9371	0.9430	0.9487	0.9386
16	0.9492	0.9667	0.9489	0.9550	0.9621	0.9514
17	0.9837	0.9971	0.9218	0.9273	0.9269	0.9240
18	0.9685	1.0000	0.9142	0.9226	0.9297	0.9161
19	0.9571	1.0000	0.9440	0.9455	0.9444	0.9460
20	0.8747	0.9745	0.9089	0.9166	0.9160	0.9121
21	0.9871	1.0000	0.9635	0.9675	0.9683	0.9662
22	0.8737	0.9787	0.7496	0.7539	0.7832	0.7497
23	0.9695	1.0000	0.9545	0.9553	0.9570	0.9559
24	0.9681	1.0000	0.9427	0.9410	0.9432	0.9392
25	0.9645	0.9631	0.9584	0.9613	0.9612	0.9562
26	0.9936	0.9809	0.9670	0.9673	0.9664	0.9687
27	0.9887	1.0000	0.9344	0.9413	0.9463	0.9384
28	0.9444	0.9785	0.8781	0.8800	0.8856	0.8777
29	0.9559	0.9998	0.9546	0.9555	0.9573	0.9528
30	0.9731	1.0001	0.9197	0.9216	0.9244	0.9206
31	0.9880	1.0000	0.9450	0.9512	0.9551	0.9466
32	0.9737	1.0000	0.9473	0.9504	0.9558	0.9468
33	0.9376	0.9706	0.9347	0.9432	0.9462	0.9384
34	0.9539	0.9799	0.9133	0.9191	0.9223	0.9162
35	0.9841	0.9999	0.9448	0.9497	0.9548	0.9482
36	0.9848	1.0008	0.9675	0.9688	0.9693	0.9676
37	0.9281	0.9785	0.8148	0.8159	0.8429	0.8107
38	0.9720	1.0000	0.9293	0.9306	0.9406	0.9278
Average	0.9626	0.9912	0.9244	0.9287	0.9340	0.9256

Table VI: Normalized mean squared error in the test set (26-Dec-2013∼30-Dec-2014).

Stock ID	**AR(10)**	**ANN (RawData)**	**DNN (RawData)**	**DNN (PCA380)**	**DNN (RBM400)**	**DNN (AE400)**
1	0.9577	1.0000	0.9478	0.9549	0.9640	0.9465
2	0.9845	1.0000	1.0100	1.0022	1.0052	1.0049
3	0.9767	0.9992	0.9738	0.9776	0.9755	0.9746
4	0.9758	1.0000	0.9811	0.9813	0.9832	0.9853
5	0.9699	1.0001	0.9749	0.9793	0.9809	0.9757
6	0.9875	0.9813	0.9738	0.9766	0.9821	0.9766
7	0.9227	0.9805	0.9170	0.9236	0.9404	0.9146
8	0.9547	1.0000	0.9601	0.9577	0.9609	0.9635
9	0.9932	1.0003	0.9950	0.9956	0.9978	0.9962
10	0.8853	0.9801	0.8683	0.8831	0.8969	0.8732
11	0.9918	0.9842	0.9733	0.9764	0.9824	0.9786
12	0.9356	1.0000	0.8957	0.9052	0.9123	0.9028
13	0.9445	1.0000	0.9393	0.9480	0.9572	0.9418
14	0.9896	0.9999	0.9999	1.0030	1.0061	0.9996
15	0.9735	1.0001	0.9612	0.9670	0.9725	0.9602
16	1.0050	0.9849	0.9839	0.9886	0.9922	0.9859
17	0.9586	0.9974	0.9514	0.9550	0.9547	0.9546
18	0.9691	1.0001	0.9610	0.9663	0.9762	0.9611
19	1.0398	1.0000	1.0100	1.0085	1.0099	1.0074
20	0.9649	0.9835	0.9506	0.9560	0.9566	0.9529
21	0.9669	0.9997	0.9721	0.9781	0.9768	0.9790
22	0.8726	0.9797	0.8796	0.8791	0.8932	0.8776
23	0.9886	1.0002	0.9916	0.9893	0.9914	0.9904
24	0.9465	1.0000	0.9532	0.9494	0.9488	0.9452
25	0.9686	0.9774	0.9799	0.9837	0.9810	0.9780
26	0.9847	0.9837	0.9778	0.9813	0.9807	0.9815
27	0.9741	0.9998	0.9838	0.9863	0.9858	0.9866
28	0.9538	0.9831	0.9712	0.9711	0.9711	0.9716
29	0.9795	0.9999	0.9781	0.9814	0.9821	0.9795
30	0.9490	1.0002	0.9553	0.9603	0.9595	0.9596
31	0.9793	1.0000	0.9616	0.9728	0.9753	0.9667
32	0.9570	1.0000	0.9669	0.9734	0.9758	0.9630
33	0.9646	0.9795	0.9623	0.9701	0.9749	0.9670
34	0.9858	0.9888	0.9973	0.9912	0.9897	0.9937
35	0.9668	1.0000	0.9765	0.9794	0.9847	0.9777
36	1.0119	1.0008	1.0059	1.0024	1.0039	1.0042
37	0.8920	0.9764	0.8828	0.8841	0.9125	0.8821
38	0.9668	1.0001	0.9663	0.9677	0.9741	0.9654
Average	0.9655	0.9937	0.9629	0.9660	0.9702	0.9638

Table VII: Mean absolute error in the test set (26-Dec-2013∼30-Dec-2014).

Stock ID	**AR(10)**	**ANN (RawData)**	**DNN (RawData)**	**DNN (PCA380)**	**DNN (RBM400)**	**DNN (AE400)**
1	0.6219	0.6148	0.6379	0.6344	0.6402	0.6375
2	0.4733	0.4651	0.4886	0.4853	0.4867	0.4869
3	0.5694	0.5641	0.5855	0.5836	0.5855	0.5847
4	0.5831	0.5726	0.5988	0.5988	0.6018	0.6010
5	0.6053	0.5980	0.6186	0.6147	0.6192	0.6180
6	0.4219	0.4095	0.4154	0.4147	0.4201	0.4126
7	0.5605	0.5440	0.5714	0.5655	0.5737	0.5678
8	0.6874	0.6677	0.7003	0.6992	0.7025	0.7031
9	0.5317	0.5223	0.5327	0.5329	0.5360	0.5342
10	0.6690	0.6487	0.6832	0.6795	0.6900	0.6828
11	0.4389	0.4232	0.4318	0.4319	0.4358	0.4317
12	0.7509	0.7421	0.7670	0.7672	0.7725	0.7684
13	0.6101	0.6041	0.6307	0.6321	0.6350	0.6328
14	0.4647	0.4591	0.4795	0.4783	0.4816	0.4772
15	0.5676	0.5587	0.5878	0.5836	0.5873	0.5861
16	0.3756	0.3654	0.3724	0.3716	0.3745	0.3720
17	0.6100	0.5996	0.6282	0.6277	0.6311	0.6307
18	0.5700	0.5548	0.5826	0.5801	0.5863	0.5819
19	0.6537	0.6322	0.6516	0.6502	0.6552	0.6511
20	0.6201	0.5903	0.6031	0.6022	0.6087	0.6020
21	0.5565	0.5502	0.5668	0.5657	0.5683	0.5671
22	0.6906	0.6644	0.7068	0.7075	0.7114	0.7066
23	0.5894	0.5738	0.5937	0.5911	0.5947	0.5923
24	0.7260	0.7093	0.7329	0.7332	0.7371	0.7349
25	0.5944	0.5947	0.5951	0.5954	0.5984	0.5943
26	0.6100	0.6145	0.6188	0.6187	0.6237	0.6182
27	0.5428	0.5340	0.5653	0.5627	0.5641	0.5642
28	0.6748	0.6560	0.6921	0.6924	0.6940	0.6939
29	0.5974	0.5786	0.5933	0.5926	0.5968	0.5935
30	0.6649	0.6573	0.6838	0.6847	0.6867	0.6853
31	0.5930	0.5837	0.6029	0.6018	0.6033	0.6016
32	0.5723	0.5645	0.5805	0.5807	0.5830	0.5798
33	0.4295	0.4184	0.4270	0.4255	0.4303	0.4261
34	0.5607	0.5451	0.5719	0.5687	0.5706	0.5716
35	0.5011	0.4972	0.5154	0.5128	0.5162	0.5143
36	0.6639	0.6595	0.6661	0.6644	0.6666	0.6651
37	0.5597	0.5422	0.5812	0.5825	0.5913	0.5832
38	0.5511	0.5418	0.5606	0.5598	0.5633	0.5612
Average	0.5806	0.5690	0.5900	0.5888	0.5927	0.5899

Table VIII: AR-DNN normalized mean squared error in the test set (26-Dec-2013∼30-Dec-2014).

Stock ID	**AR(10)**	**AR-DNN (RawData)**	**AR-DNN (PCA380)**	**AR-DNN (RBM400)**	**AR-DNN (AE400)**
1	0.9577	0.9440	0.9476	0.9557	0.9484
2	0.9845	0.9912	0.9879	0.9869	0.9894
3	0.9767	0.9722	0.9745	0.9738	0.9709
4	0.9758	0.9823	0.9836	0.9755	0.9780
5	0.9699	0.9715	0.9705	0.9705	0.9705
6	0.9875	0.9771	0.9773	0.9758	0.9794
7	0.9227	0.9124	0.9133	0.9161	0.9106
8	0.9547	0.9525	0.9530	0.9547	0.9543
9	0.9932	0.9933	0.9955	0.9948	0.9951
10	0.8853	0.8791	0.8811	0.8823	0.8778
11	0.9918	0.9839	0.9838	0.9817	0.9876
12	0.9356	0.9234	0.9184	0.9190	0.9204
13	0.9445	0.9395	0.9412	0.9413	0.9390
14	0.9896	0.9925	0.9916	0.9919	0.9908
15	0.9735	0.9637	0.9628	0.9647	0.9600
16	1.0050	0.9972	0.9982	0.9987	1.0004
17	0.9586	0.9488	0.9535	0.9551	0.9528
18	0.9691	0.9603	0.9610	0.9647	0.9599
19	1.0398	1.0421	1.0423	1.0408	1.0446
20	0.9649	0.9667	0.9673	0.9650	0.9675
21	0.9669	0.9665	0.9667	0.9670	0.9667
22	0.8726	0.8750	0.8711	0.8692	0.8724
23	0.9886	0.9897	0.9887	0.9888	0.9894
24	0.9465	0.9447	0.9449	0.9461	0.9443
25	0.9686	0.9702	0.9690	0.9704	0.9685
26	0.9847	0.9736	0.9742	0.9750	0.9757
27	0.9741	0.9745	0.9759	0.9741	0.9743
28	0.9538	0.9540	0.9539	0.9537	0.9548
29	0.9795	0.9812	0.9813	0.9811	0.9795
30	0.9490	0.9496	0.9493	0.9493	0.9495
31	0.9793	0.9666	0.9679	0.9713	0.9655
32	0.9570	0.9546	0.9556	0.9566	0.9542
33	0.9646	0.9586	0.9565	0.9612	0.9627
34	0.9858	0.9895	0.9845	0.9842	0.9858
35	0.9668	0.9653	0.9648	0.9665	0.9658
36	1.0119	1.0126	1.0119	1.0109	1.0123
37	0.8920	0.8808	0.8906	0.8887	0.8800
38	0.9668	0.9620	0.9626	0.9630	0.9605
Average	0.9655	0.9622	0.9625	0.9628	0.9621

Table IX: DNN-AR normalized mean squared error in the test set (26-Dec-2013∼30-Dec-2014).

Stock ID	**AR(10)**	**DNN-AR (RawData)**	**DNN-AR (PCA380)**	**DNN-AR (RBM400)**	**DNN-AR (AE400)**
1	0.9577	0.9437	0.9446	0.9522	0.9430
2	0.9845	1.0118	1.0067	1.0082	1.0096
3	0.9767	0.9737	0.9727	0.9711	0.9717
4	0.9758	0.9815	0.9814	0.9834	0.9856
5	0.9699	0.9748	0.9723	0.9741	0.9730
6	0.9875	0.9845	0.9831	0.9889	0.9812
7	0.9227	0.9125	0.9132	0.9233	0.9112
8	0.9547	0.9591	0.9571	0.9597	0.9634
9	0.9932	0.9965	0.9974	1.0006	0.9996
10	0.8853	0.8653	0.8689	0.8800	0.8688
11	0.9918	0.9904	0.9935	0.9950	0.9970
12	0.9356	0.8959	0.8992	0.9015	0.9005
13	0.9445	0.9388	0.9451	0.9494	0.9414
14	0.9896	0.9998	1.0011	1.0043	0.9982
15	0.9735	0.9583	0.9581	0.9608	0.9559
16	1.0050	1.0038	1.0057	1.0067	1.0035
17	0.9586	0.9514	0.9507	0.9512	0.9530
18	0.9691	0.9603	0.9652	0.9738	0.9615
19	1.0398	1.0371	1.0373	1.0377	1.0344
20	0.9649	0.9628	0.9643	0.9663	0.9647
21	0.9669	0.9696	0.9700	0.9697	0.9727
22	0.8726	0.8791	0.8789	0.8932	0.8771
23	0.9886	0.9952	0.9930	0.9955	0.9945
24	0.9465	0.9424	0.9424	0.9423	0.9414
25	0.9686	0.9729	0.9746	0.9743	0.9711
26	0.9847	0.9721	0.9732	0.9738	0.9745
27	0.9741	0.9846	0.9835	0.9820	0.9854
28	0.9538	0.9705	0.9719	0.9731	0.9711
29	0.9795	0.9805	0.9807	0.9833	0.9808
30	0.9490	0.9556	0.9600	0.9575	0.9599
31	0.9793	0.9616	0.9668	0.9682	0.9639
32	0.9570	0.9611	0.9638	0.9626	0.9562
33	0.9646	0.9620	0.9629	0.9675	0.9635
34	0.9858	0.9959	0.9991	0.9986	0.9994
35	0.9668	0.9755	0.9724	0.9768	0.9738
36	1.0119	1.0127	1.0102	1.0124	1.0127
37	0.8920	0.8834	0.8837	0.9021	0.8825
38	0.9668	0.9650	0.9652	0.9687	0.9650
Average	0.9655	0.9643	0.9650	0.9682	0.9648

Table X: Average normalized mean squared error in the test set (26-Dec-2013∼30-Dec-2014). The figures in the parentheses are NMSE difference between AR-DNN and AR. DNN-AR row is similarly defined except the figures in the parentheses being NMSE difference between DNN-AR and DNN.

Method	**Data Representation for DNN**			
	(RawData)	**(PCA380)**	**(RBM400)**	**(AE400)**
AR(10)	0.9655			
ANN	0.9937	0.9990	0.9982	0.9976
DNN	0.9629	0.9660	0.9702	0.9638
AR-DNN	0.9622 (-0.0033)	0.9625 (-0.0030)	0.9628 (-0.0027)	0.9621 (-0.0034)
DNN-AR	0.9643 (0.0013)	0.9650 (-0.0010)	0.9682 (-0.0020)	0.9648 (0.0010)

Table XI: Average root mean squared error in the test set (26-Dec-2013∼30-Dec-2014). The figures in the parentheses are RMSE difference between AR-DNN and AR. DNN-AR row is similarly defined except the figures in the parentheses being RMSE difference between DNN-AR and DNN.

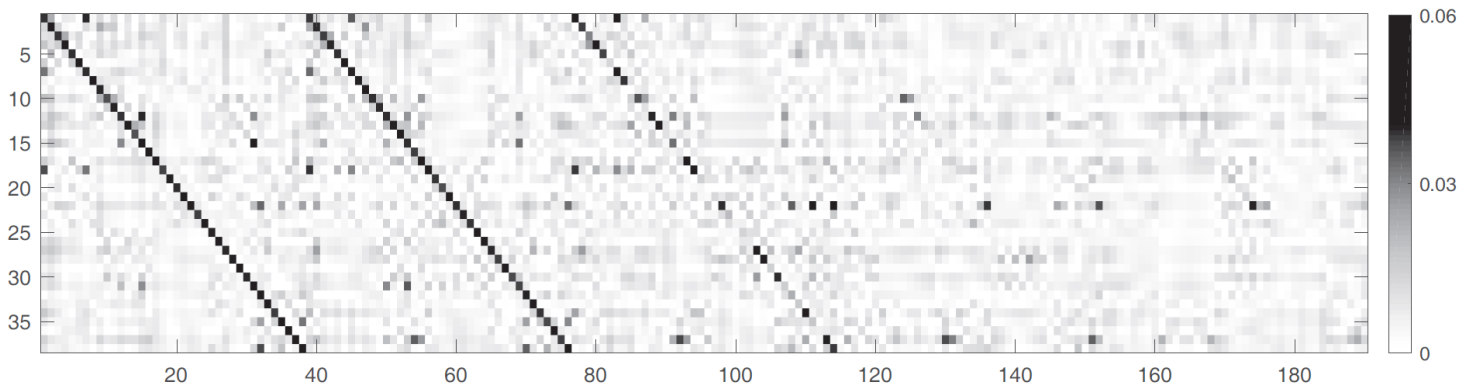
Method	**Data Representation for DNN**			
	(RawData)	**(PCA380)**	**(RBM400)**	**(AE400)**
AR(10)	0.8229			
ANN	0.8357	0.8379	0.8376	0.8374
DNN	0.8216	0.8230	0.8248	0.8220
AR-DNN	0.8215 (-0.0014)	0.8216 (-0.0013)	0.8217 (-0.0012)	0.8214 (-0.0015)
DNN-AR	0.8221 (0.0005)	0.8224 (-0.0006)	0.8238 (-0.0010)	0.8223 (0.0003)

Table XII: Average mean absolute error in the test set (26-Dec-2013∼30-Dec-2014). The figures in the parentheses are MAE difference between AR-DNN and AR. DNN-AR row is similarly defined except the figures in the parentheses being MAE difference between DNN-AR and DNN.

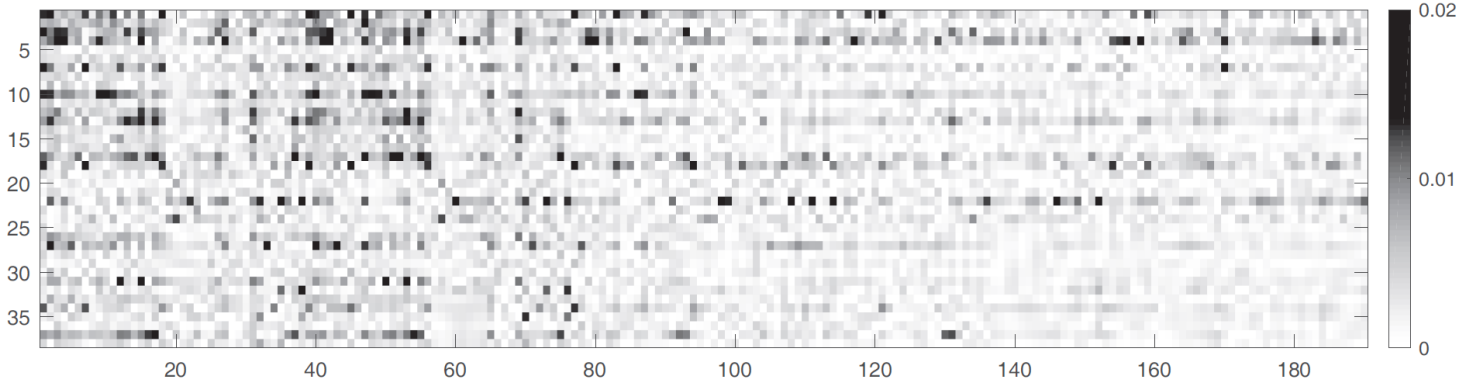
Method	**Data Representation for DNN**			
	(RawData)	**(PCA380)**	**(RBM400)**	**(AE400)**
AR(10)	0.5806			
ANN	0.5690	0.5675	0.5684	0.5678
DNN	0.5900	0.5888	0.5927	0.5899
AR-DNN	0.5852 (0.0046)	0.5850 (0.0044)	0.5838 (0.0032)	0.5852 (0.0046)
DNN-AR	0.5922 (0.0022)	0.5931 (0.0043)	0.5937 (0.0010)	0.5926 (0.0027)

Table XIII: Average mutual information in the test set (26-Dec-2013∼30-Dec-2014). The figures in the parentheses are MI difference between AR-DNN and AR. DNN-AR row is similarly defined except the figures in the parentheses being MI difference between DNN-AR and DNN.

Method	**Data Representation for DNN**			
	(RawData)	**(PCA380)**	**(RBM400)**	**(AE400)**
AR(10)	0.0179			
ANN	0.0032	0.0005	0.0009	0.0012
DNN	0.0193	0.0176	0.0154	0.0188
AR-DNN	0.0197 (0.0018)	0.0195 (0.0016)	0.0193 (0.0015)	0.0197 (0.0018)
DNN-AR	0.0187 (-0.0006)	0.0182 (0.0006)	0.0165 (0.0012)	0.0183 (-0.0005)



(a) DNN

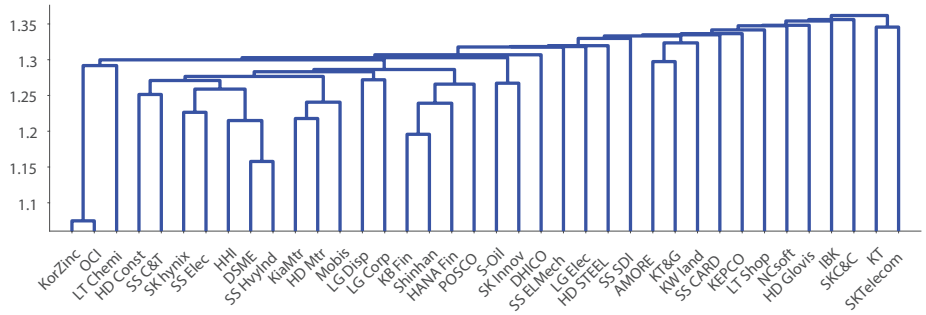


(b) AR-DNN

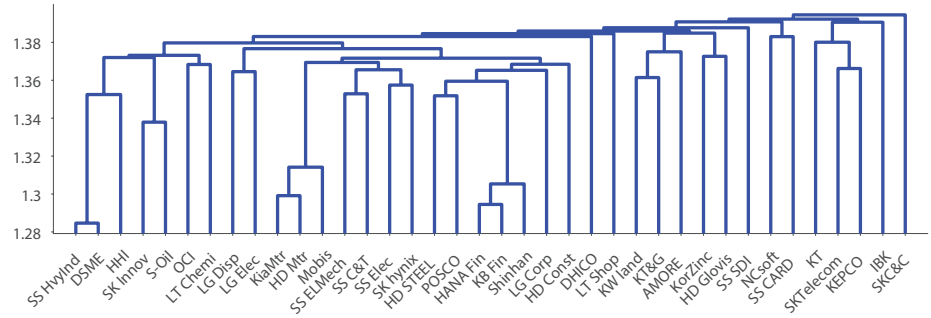
Figure 6: Heat map of the sensitivity matrix. This represents the sensitivity of the predictor functions with respect to the past returns. y-axis represents the prediction function of each stock return, and x-axis represents the past returns of the stocks in the order of $r_{1,t}, r_{2,t}, \cdots, r_{1,t-1}, r_{2,t-1}, \cdots, r_{38,t-4}$. Sensitivities to the lagged returns with lag > 5 are omitted as their values are close to 0. White represents zero sensitivity and darker color means higher sensitivity.

6 Application to Stock Market Structure Analysis

Identification of the correlation between stocks is important for understanding stock market structure and its behavior. For example, the correlation coefficient can be used to measure the distance between two stocks using the formula, $d(i,j) = \sqrt{2(1-\rho_{ij})}$, where ρ_{ij} is the correlation coefficient between stock i and stock j returns. This distance measure has been used in several researches (Mantegna, 1999; Bonanno, Lillo, & Mantegna, 2001; Onnela, Kaski, & Kertész, 2004; Tumminello, Lillo, & Mantegna, 2010). These researches normally assume that the stock returns are unpredictable, and use a sample covariance to estimate the covariance. However, the results in the previous sections suggest that the stock returns are predictable to some extent. This motivates us to incorporate the stock return prediction into covariance estimation to enhance the estimation accuracy.



(a) Training set (04-Jan-2010 $\sim$ 24-Dec-2013)



(b) Test set (26-Dec-2013 $\sim$ 30-Dec-2014)

Figure 7: MST-based hierarchical tree structures of the stocks in our sample. Group average method (Day & Edelsbrunner, 1984) is used for tree construction, where y-axis denotes the group average distance.

One way of analyzing the market structure is to construct a hierarchical structure of the stocks using the minimum spanning tree (MST) (Graham & Hell, 1985). Given the distances of all pairs of the stocks, $d(i,j),\ i,j = 1,\ldots,M$, a spanning tree (ST) is defined by a set of edges, so that there exist unique paths (serially connected edges) between every pair of data points (stocks, in our case). Then, MST is determined as the ST that minimizes the sum of the edge distances: $\text{MST} = \underset{\text{ST}}{\text{argmin}} \sum_{(i,j)\in \text{ST}} d(i,j)$, where (i,j) denotes an edge between stocks i and j. Once MST is found, different algorithms can be employed

for hierarchical tree analysis or clustering (Gordon, 1987; Grygorash, Zhou, & Jorgensen, 2006). As an example, Figure 7 shows hierarchical tree structures of the stocks in our sample, constructed using the group average method proposed in Day and Edelsbrunner (1984). The correlation coefficients are obtained from the sample covariance of each dataset. The hierarchical structures of the market appear to be in accordance with the general perception of the market: For instance, the banking industry stocks (KB Fin, Shinhan, and HANA Fin) are grouped together both in the training set and test set, and so do the motor industry stocks (KiaMtr, HD Mtr, and Mobis). Nevertheless, the two tree structures are not identical, which suggests the necessity of a forward-looking correlation estimation to anticipate the change of the market structure.

We define a forward-looking covariance using the stock return prediction as follows. Given the prediction of returns, $\hat{r}_{t+1}$, the forward-looking covariance, Σ_f, is defined as: [7]

$$\Sigma_f \triangleq E[r_{t+1}{r_{t+1}}^T|\hat{r}_{t+1}] - \mu\mu^T, \tag{22}$$

where $\mu = E[r_{t+1}]$. Since $E[r_{t+1}|\hat{r}_{t+1}] = \hat{r}_{t+1}$, it can be rewritten as

$$\Sigma_f = \mathcal{B} + \hat{r}_{t+1}\hat{r}_{t+1}^T - \mu\mu^T, \tag{23}$$

where $\mathcal{B} \triangleq E[(r_{t+1} - \hat{r}_{t+1})(r_{t+1} - \hat{r}_{t+1})^T]$ is the unpredictable component of the covariance, which is estimated from the past returns and their predictions, and the second term is the predictable component. We use our return prediction model to obtain the forward-looking covariance.

The performance of the forward-looking covariance is evaluated via the log-likelihood function: given the mean and covariance estimates at each time in the test set, the *out-of-sample* log-likelihood is given by:

$$\begin{aligned} \mathcal{L} &= \sum_{n=1}^{N_2} \ln \mathcal{N}({r_{t+1}}^n; \hat{\mu}^n, \hat{\Sigma}^n) \\ &= \sum_{n=1}^{N_2} -\frac{1}{2}\ln|\hat{\Sigma}^n| - \frac{1}{2}({r_{t+1}}^n - \hat{\mu}^n)^T(\hat{\Sigma}^n)^{-1}({r_{t+1}}^n - \hat{\mu}^n) + C, \end{aligned} \tag{24}$$

where r_{t+1}^n, $\hat{\mu}^n$, and $\hat{\Sigma}^n$ are the target returns (true one-period ahead returns) and their mean and covariance estimates for the nth data point, and C is a constant term. We test several estimation methods and compare them in Table XIV.

In the table, estimation methods M1-M4 estimate $\hat{\mu}$ and $\hat{\Sigma}$ using the sample analogues, but with different samples: Subscript "tr" refers to the training set and "ts" the test set. For example, M4 calculates the sample mean and covariance from the test set, *i.e.,* M4 assumes that the distribution of the returns in the test set is constant and known *a priori*. As expected, using the test set for covariance calculation (M2 and M4) gives larger log-likelihood values compared with using the training set (M1 and M3). Interestingly, however, using a prior information on the mean (M3 and M4) has little effect on the log-likelihood.

M5 uses the forward-looking covariance estimate in (23). $\mathcal{B}$ is estimated from the training set and assumed to be constant, while $\hat{r}_{t+1}$ is updated at each time. M5 outperforms M1 and M3 which are

[7] Note that in this section, r_{t+1} denotes a vector of stock returns, *i.e.,* $r_{t+1} = [r_{1,t+1}, \ldots, r_{38,t+1}]^T$. $\hat{r}_{t+1}$ and μ are also defined in a similar manner.

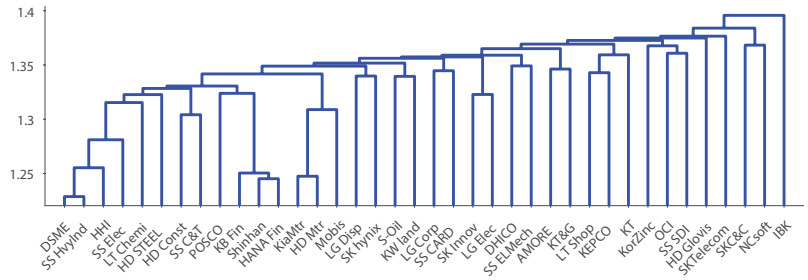
Table XIV: Comparison of different mean and covariance estimation methods. Subscripts tr, ts, and ma respectively refer to the test set, training set, and moving average. The log-likelihood values are calculated in the test set (26-Dec-2013∼30-Dec-2014), ignoring the constant term in (24).

Estimation Methods	$\hat{\mu}$	$\hat{\Sigma}$	**Log-likelihood,** $\mathcal{L}$ $(\times 10^5)$			
			AR-DNN (RawData)	**AR-DNN (PCA380)**	**AR-DNN (RBM400)**	**AR-DNN (AE400)**
M1	$\hat{\mu}_{\text{tr}}$	$\hat{\Sigma}_{\text{tr}}$		-1.8601		
M2	$\hat{\mu}_{\text{tr}}$	$\hat{\Sigma}_{\text{ts}}$		-1.6638		
M3	$\hat{\mu}_{\text{ts}}$	$\hat{\Sigma}_{\text{tr}}$		-1.8599		
M4	$\hat{\mu}_{\text{ts}}$	$\hat{\Sigma}_{\text{ts}}$		-1.6636		
M5	$\hat{\mu}_{\text{tr}}$	$\mathcal{B}_{\text{tr}} + \hat{r}_{t+1}\hat{r}_{t+1}^T - \hat{\mu}_{tr}\hat{\mu}_{tr}^T$	-1.7606	-1.7617	-1.7646	-1.7592
• Using rolling window (sample size of 3,000)						
M6	$\hat{\mu}_{\text{tr}}$	$\hat{\Sigma}_{\text{ma}}$		-1.6320		
M7	$\hat{\mu}_{\text{ma}}$	$\hat{\Sigma}_{\text{ma}}$		-1.6326		
M8	$\hat{\mu}_{\text{tr}}$	$\mathcal{B}_{\text{ma}} + \hat{r}_{t+1}\hat{r}_{t+1}^T - \hat{\mu}_{tr}\hat{\mu}_{tr}^T$	-1.5464	-1.5476	-1.5480	-1.5459
M9	$\hat{\mu}_{\text{ma}}$	$\mathcal{B}_{\text{ma}} + \hat{r}_{t+1}\hat{r}_{t+1}^T - \hat{\mu}_{\text{ma}}\hat{\mu}_{\text{ma}}^T$	-1.5575	-1.5588	-1.5590	-1.5570

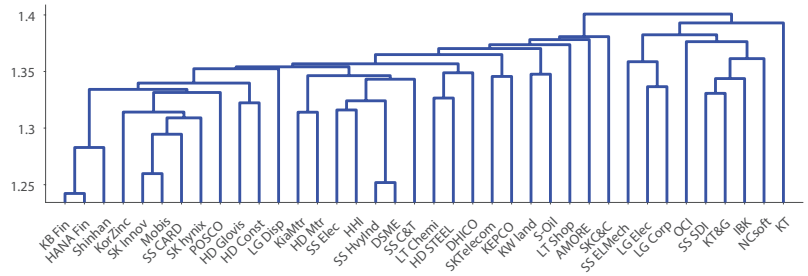
based on the training set sample covariance, but underperforms M2 and M4 which are based on the test set sample covariance.

From M6 to M9, the mean and covariance estimates are updated each time by rolling the sample window (denoted by subscript "ma"). The results in the table are from the sample window size of 3,000. Using other sample sizes, *e.g.*, 500 or 5,000, yield similar results (not reported here). The methods based on rolling covariance estimation (M6 and M7) perform slightly better than M2 and M4, which assume a prior information on the covariance of the test set. The most remarkable result, however, comes from M8 and M9, which incorporate return prediction and estimate $\mathcal{B}$ rolling the sample window. M8 and M9 clearly outperform all other methods no matter which data representation method is used. Again, using different mean estimates has little impact on the results.

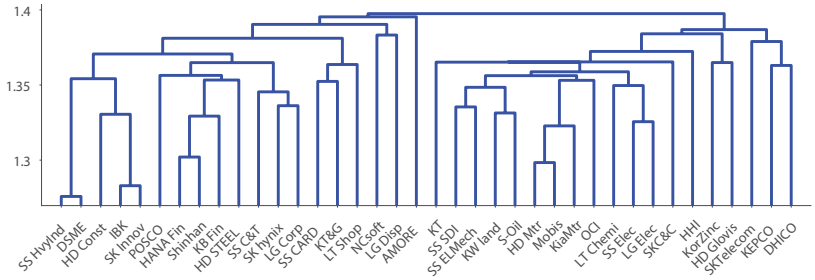
Based on the results in Table XIV, we believe that a structural change of the market can be better anticipated using our model. As an illustrative example, Figure 8 shows the change of the hierarchical structure of the stocks over time, obtained from M8 with AR-DNN (RBM400).



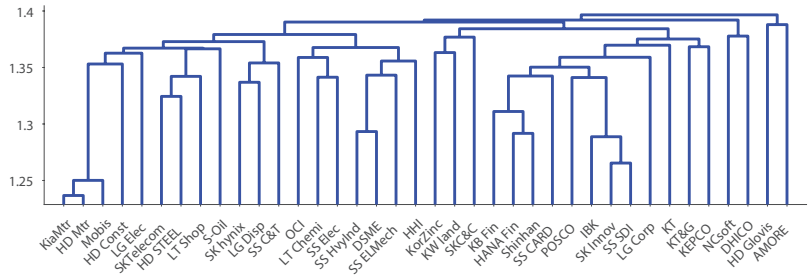
(a) 26-Dec-2013



(b) 07-Apr-2014



(c) 16-Jul-2014



(d) 28-Oct-2014

Figure 8: Cross-sectional hierarchical tree structure of the stocks for some selected sample periods, obtained from M8 with AR-DNN (RBM400).

7 Concluding Remarks

In this paper, we propose a deep feature learning-based stock market prediction model. From the raw level input consisting of lagged stock returns, we generate seven sets of features employing three data representation methods; principal component analysis, autoencoder, and restricted Boltzmann machine, and construct three-layer deep neural networks (DNN) to predict the future stock returns. Applied to the Korean stock market, we find that the DNNs perform better than a linear autoregressive model in the training set, but the advantage mostly disappears in the test set. This result can be partially explained by the sensitivity analysis which shows that the predicted part of a stock return is mostly governed by the first three lagged returns of itself, and the lagged returns of other stocks have negligible influence. Nevertheless, by applying the DNN to the residuals of the autoregressive model, we find that the DNN can extract additional information and improve prediction. This does not hold true when the autoregressive model is applied to the residuals of the DNN, demonstrating the advantages of DNNs over the autoregressive model. Furthermore, we apply our prediction model to covariance-based market structure analysis and find that our model improves covariance estimation effectively.

These results can help high-frequency traders improve stock return and covariance prediction. Combined together, they can forecast overall market return and risk, which can be used for trading market index derivatives such as index futures and options.

One of main advantages of DNNs is the ability to extract features from a large set of raw data without relying on prior knowledge of predictors. This makes deep learning particularly suitable for stock market prediction, in which numerous factors affect stock prices in a complex, nonlinear fashion. If there exist factors with strong evidence of predictability, exploiting those factors may likely give better performance than simply dumping a large raw dataset. However, we can also use these factors as part of the input data for deep learning, and let deep learning identify the relationship between the factors and stock prices. Nevertheless, with the ongoing release of newer deep learning algorithms, testing and choosing the most appropriate model can be a challenge in applying deep learning.

To the best of our knowledge, this is—at least in the public domain—one of the few comprehensive studies that examine the effectiveness of deep feature learning for stock market analysis and prediction. The empirical results presented here by no means offer any conclusive demonstration of the superiority of deep learning networks, but do suggest promising extensions and directions of further investigation. For instance, additional factors that are known to carry information about the future price movement, such as trading volume and the price of a derivative linked to the stock, can be augmented to the input. Identifying the complex relationship between the future price and all these factors is practically impossible without harnessing the power of deep feature learning; the method may be found to be more effective when applied to higher frequency data where market microstructure affects price dynamics. Data representation methods and network functions other than those considered may also offer better performance. These future tasks require vast amounts of data and computer resources, and are beyond the scope the current research given the limited data availability and hardware resources. We hope to see more researchers participating in this line of research in a near future.

Acknowledgement

Financial support from the Institute for Research in Finance and Economics of Seoul National University is gratefully acknowledged.

References

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, *2014*.

Agrawal, J., Chourasia, V., & Mittra, A. (2013). State-of-the-art in stock prediction techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, *2*(4), 1360–1366.

Ang, A., & Bekaert, G. (2007). Stock return predictability: Is it there? *Review of Financial studies*, *20*(3), 651–707.

Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing* (pp. 424–436).

Armano, G., Marchesi, M., & Murru, A. (2005). A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences*, *170*(1), 3–33.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques–part ii: Soft computing methods. *Expert Systems with Applications*, *36*(3), 5932–5941.

Bacchetta, P., Mertens, E., & Van Wincoop, E. (2009). Predictability in financial markets: What do survey expectations tell us? *Journal of International Money and Finance*, *28*(3), 406–426.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, *19*, 153.

Bishop, C. M. (2006). Pattern recognition.

Bogullu, V. K., Dagli, C. H., & Enke, D. L. (2002). Using neural networks and technical indicators for generating stock trading signals.

Bollerslev, T., Marrone, J., Xu, L., & Zhou, H. (2014). Stock return predictability and variance risk premia: statistical inference and international evidence. *Journal of Financial and Quantitative Analysis*, *49*(03), 633–661.

Bonanno, G., Lillo, F., & Mantegna, R. N. (2001). High-frequency cross-correlation in a set of stocks.

Bondt, W. F., & Thaler, R. (1985). Does the stock market overreact? *The Journal of finance*, *40*(3), 793–805.

Bradley, D. A. (1950). *Stock market prediction*. Llewellyn Publications.

Campbell, J. Y., & Hamao, Y. (1992). Predictable stock returns in the united states and japan: A study of long-term capital market integration. *The Journal of Finance*, *47*(1), 43–69.

Campbell, J. Y., & Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *Review of Financial Studies*, *21*(4), 1509–1531.

Campbell, S. D. (2012). Macroeconomic volatility, predictability, and uncertainty in the great moderation. *Journal of Business & Economic Statistics*.

Cao, Q., Leggio, K. B., & Schniederjans, M. J. (2005). A comparison between fama and french's model and artificial neural networks in predicting the chinese stock market. *Computers & Operations Research*, *32*(10), 2499–2512.

Carreira-Perpinan, M. A., & Hinton, G. (2005). On contrastive divergence learning. In *Aistats* (Vol. 10, pp. 33–40).

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., & Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with*

Applications, *55*, 194–211.

Cervelló-Royo, R., Guijarro, F., & Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the djia index with intraday data. *Expert Systems with Applications*, *42*(14), 5963–5975.

Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific Model Development*, *7*(3), 1247–1250.

Chen, A.-S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: forecasting and trading the taiwan stock index. *Computers & Operations Research*, *30*(6), 901–923.

Chen, T.-l., & Chen, F.-y. (2016). An intelligent pattern recognition model for supporting investment decisions in stock market. *Information Sciences*, *346*, 261–274.

Chiang, W.-C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications*, *59*, 195–207.

Cho, K., Ilin, A., & Raiko, T. (2011). Improved learning of gaussian-bernoulli restricted boltzmann machines. In *International conference on artificial neural networks* (pp. 10–17).

Chourmouziadis, K., & Chatzoglou, P. D. (2016). An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. *Expert Systems with Applications*, *43*, 298–311.

CireşAn, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, *32*, 333–338.

Coates, A., Lee, H., & Ng, A. Y. (2010). An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, *1001*(48109), 2.

Day, W. H., & Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, *1*(1), 7–24.

Deng, L., & Yu, D. (2014). Deep learning. *Signal Processing*, *7*, 3–4.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep learning for event-driven stock prediction. In *Ijcai* (pp. 2327–2333).

Enke, D., & Mehdiyev, N. (2013). Stock market prediction using a combination of stepwise regression analysis, differential evolution-based fuzzy clustering, and a fuzzy inference neural network. *Intelligent Automation & Soft Computing*, *19*(4), 636–648.

Enke, D., & Mehdiyev, N. (2014). A hybrid neuro-fuzzy model to forecast inflation. *Procedia Computer Science*, *36*, 254–260.

Ferreira, M. A., & Santa-Clara, P. (2011). Forecasting stock market returns: The sum of the parts is more than the whole. *Journal of Financial Economics*, *100*(3), 514–537.

Gordon, A. D. (1987). A review of hierarchical classification. *Journal of the Royal Statistical Society. Series A (General)*, 119–137.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing*, *7*(1), 43–57.

Granger, C. W. J., & Morgenstern, O. (1970). *Predictability of stock market prices* (Vol. 1). DC Heath Lexington, Mass.

Grygorash, O., Zhou, Y., & Jorgensen, Z. (2006). Minimum spanning tree based clustering algorithms. In *2006 18th ieee international conference on tools with artificial intelligence (ictai'06)* (pp. 73–81).

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011a). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*(8), 10389–10397.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011b). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*(8), 10389–10397.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, *14*(8), 1771–1800.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hussain, A. J., Knowles, A., Lisboa, P. J., & El-Deredy, W. (2008). Financial time series prediction using polynomial pipelined neural networks. *Expert Systems with Applications*, *35*(3), 1186–1199.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Kazem, A., Sharifi, E., Hussain, F. K., Saberi, M., & Hussain, O. K. (2013). Support vector regression with chaos-based firefly algorithm for stock market price forecasting. *Applied soft computing*, *13*(2), 947–958.

Khashei, M., & Bijari, M. (2011). A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, *11*(2), 2664–2675.

Kim, J. H., Shamsuddin, A., & Lim, K.-P. (2011). Stock return predictability and the adaptive markets hypothesis: Evidence from century-long us data. *Journal of Empirical Finance*, *18*(5), 868–879.

Kim, Y., & Enke, D. (2016a). Developing a rule change trading system for the futures market using rough set analysis. *Expert Systems with Applications*, *59*, 165–173.

Kim, Y., & Enke, D. (2016b). Using neural networks to forecast volatility for an asset allocation strategy based on the target volatility. *Procedia Computer Science*, *95*, 281–286.

Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, *42*, 11–24.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, C., Sehwan, Y., & Jongdae, J. (2007). Neural network model versus sarima model in forecasting korean stock price index (kospi). *Issues in Information System*, *8*(2), 372–378.

Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems* (pp. 1096–1104).

Lee, K. J., Chi, A. Y., Yoo, S., & Jin, J. J. (2008). Forecasting korean stock price index (kospi) using back propagation neural network model, bayesian chiao's model, and sarima model. *Academy of Information and Management Sciences Journal*, *11*(2), 53.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, *11*(1), 193–197.

Monfared, S. A., & Enke, D. (2014). Volatility forecasting using a hybrid gjr-garch neural network model. *Procedia Computer Science*, *36*, 246–253.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).

Niaki, S. T. A., & Hoseinzade, S. (2013). Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, *9*(1), 1.

Onnela, J.-P., Kaski, K., & Kertész, J. (2004). Clustering and information in correlation based financial networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, *38*(2), 353–362.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, *42*(4), 2162–2172.

Phan, D. H. B., Sharma, S. S., & Narayan, P. K. (2015). Stock return forecasting: some new evidence. *International Review of Financial Analysis*, *40*, 38–51.

Qiu, M., Song, Y., & Akagi, F. (2016). Application of artificial neural network for the prediction of stock market returns: The case of the japanese stock market. *Chaos, Solitons & Fractals*, *85*, 1–7.

Rather, A. M., Agarwal, A., & Sastry, V. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, *42*(6), 3234–3241.

Salakhutdinov, R., & Hinton, G. E. (2009). Deep boltzmann machines. In *Aistats* (Vol. 1, p. 3).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2006). Modeling human motion using binary latent variables. In *Advances in neural information processing systems* (pp. 1345–1352).

Thawornwong, S., & Enke, D. (2004). The adaptive selection of financial and economic variables for use with artificial neural networks. *Neurocomputing*, *56*, 205–232.

Ticknor, J. L. (2013). A bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, *40*(14), 5501–5506.

Tsai, C.-F., & Hsiao, Y.-C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, *50*(1), 258–269.

Tumminello, M., Lillo, F., & Mantegna, R. N. (2010). Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, *75*(1), 40–58.

Wang, J.-Z., Wang, J.-J., Zhang, Z.-G., & Guo, S.-P. (2011). Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, *38*(11), 14346–14355.

Yeh, C.-Y., Huang, C.-W., & Lee, S.-J. (2011). A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, *38*(3), 2177–2186.

Yoshihara, A., Fujikawa, K., Seki, K., & Uehara, K. (2014). Predicting stock market trends by recurrent deep neural networks. In *Pacific rim international conference on artificial intelligence* (pp. 759–769).

Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, *67*, 126–139.

Zhu, X., Wang, H., Xu, L., & Li, H. (2008). Predicting stock index increments by neural networks: The role of trading volume under different horizons. *Expert Systems with Applications*, *34*(4), 3043–3054.