Learning Universal Stock Price Patterns through Return-Volatility Normalization for Accurate Stock Price Prediction

Anonymous Author(s)

Abstract

How can we learn universal stock price patterns shared by all stocks to accurately predict future stock prices? Stock price prediction technology has garnered significant attention in both academia and industry due to its ability to recognize complex patterns. Most stock price prediction techniques aim to identify universal price patterns shared by all stocks from price data to predict future prices. However, it has been a challenging task to discover universal price patterns since price data reflect unique characteristics of individual stocks.

In this paper, we propose RVN-UP (Return-Volatility Normalization for Learning Universal Stock Price Patterns), a novel framework for stock price prediction. The goal of RVN-UP is to accurately predict stock prices through learning universal price patterns. RVN-UP achieves this goal by 1) normalizing price features to remove individual stock characteristics, including return, volatility, and price scale, and 2) a stock price modeling approach based on GBM that integrates universal pattern recognition with individual stock characteristics. RVN-UP achieves improvements of up to 12.8 times in information coefficient and 4.5 times in annualized rate of return compared to the previous state-of-the-art models across real-world datasets collected from various stock markets in U.S., China, and Korea. Additionally, through universal pattern learning, RVN-UP successfully recognizes price patterns of stocks not seen in the training set, unlike existing methods.

1 Introduction

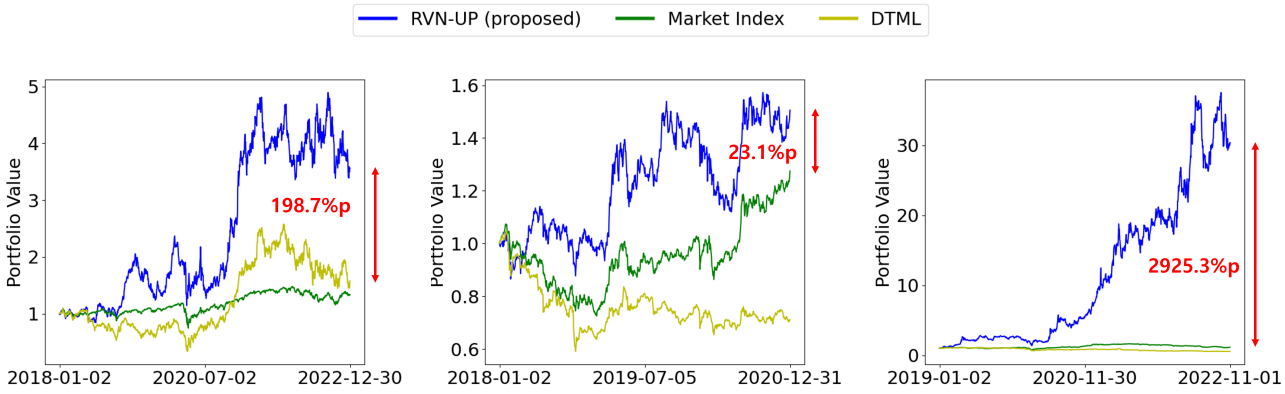
How can we learn universal stock price patterns for accurate stock price prediction? Accurate stock price prediction can lead to maximized investment returns. Additionally, with recent advancements in data mining and machine learning technologies, system trading has been widely used, making accurate stock price prediction methods a topic of great interest.

Many stock price prediction techniques aim to discover universal patterns shared by all stocks in price data. However, identifying universal stock price patterns is challenging since stock price data contain unique characteristics of individual stocks. Finding universal patterns is challenging in many pattern recognition tasks due to individual sample characteristics. For example, in speech recognition, characteristics such as tone differences based on speaker's gender and age, as well as background noise, make it difficult to identify universal patterns (fundamental phonemes) in speech. In this case, it is necessary to remove voice tone characteristics and background noise from each sample to accurately recognize fundamental phonemes [4], [15]. Similarly, in stock price prediction, it is essential to remove individual stock characteristics from each sample to recognize universal stock price patterns. Previous works on stock prediction have considered only price scale as a characteristic of individual stocks for data normalization [13], [3], [16]. However, normalizing price scale does not sufficiently eliminate individual stock characteristics. Therefore, existing methods struggle to learn universal patterns in stock prices since they fail to remove characteristics of individual stocks.

In this paper, we propose RVN-UP (Return-Volatility Normalization for Learning Universal Stock Price Patterns), a method for learning universal stock price patterns by removing characteristics of each individual stock for accurate stock price prediction. The main ideas of RVN-UP are as follows. First, RVN-UP learns universal stock price patterns by removing individual stock characteristics: return, volatility, and price scale from stock price. Second, RVN-UP presents a modeling approach that integrates stock characteristics and pure patterns based on geometric Brownian motion (GBM). We propose a neural network to replace the Brownian motion part of GBM, addressing the challenge of predicting future stock prices due to the independence assumption of GBM.

Our contributions are summarized as follows:

- **Learning Universal Stock Price Patterns.** RVN-UP learns universal stock price patterns by removing characteristics of individual stocks, such as return and volatility.
- **Integration of Short-Term Patterns and Long-Term Trends.** For accurate prediction, both long-term trends and short-term patterns must be considered together. RVN-UP identifies short-term patterns through a neural network and integrates them with long-term trends represented by return and volatility.
- **Experiments.** We demonstrate through experiments that RVN-UP improves information coefficient by up to 12.8 times and enhances annualized rate of return by up to 4.5 times compared to



(a) U.S. stock market (b) CN stock market (c) KR stock market

Figure 1: Performance on U.S., Chinese, and Korean markets for stocks the models have not seen during training. RVN-UP identifies universal patterns even in previously unseen stocks, generating significant profits.

Table 1: Symbols and descriptionss.

Symbol	Definition
S_t	the stock price at time t
B_t	the value of Brownian motion at time t
μ	the return of a stock
σ	the volatility of a stock
ϵ_t	the error term at time t
$\epsilon_t^{open}, \epsilon_t^{high}, \epsilon_t^{low}$	the error term of opening, highest, and lowest prices at time t
s, u, v	the time interval between the opening, highest, and lowest prices, and closing price
θ	the parameters of the neural network inside RVN-UP
f_θ	the neural network inside RVN-UP
$\mathbf{E}$	a return-volatility normalized feature matrix
w	the window size

the state-of-the-art method. Additionally, we show that RVN-UP learns universal stock price patterns, improving information coefficient by up to 7.0 times and annualized rate of return by up to 4.0 times for stocks not seen during training (see Figure 1).

The rest of this paper is organized as follows. We introduce related research on stock price prediction in Section 2. We propose RVN-UP in Section 3. We present the experimental results on real-world stock datasets in Section 4 and conclude at Section 5. We summarize the definitions of the symbols in this paper in Table 1.

2 Related Works

We introduce the related studies on stock price prediction. First, we present traditional stock price modeling methods, and then we discuss deep learning-based stock price prediction methods.

2.1 Traditional Stock Price Modeling Methods

In this subsection, we review several traditional methods used for stock price modeling. These methods primarily rely on statistical and mathematical foundations to capture the dynamics of stock prices over time.

Geometric Brownian motion (GBM) [2] is one of the most popular models for stock price dynamics. GBM is widely used in various financial tasks, such as option pricing and portfolio optimization, due to its suitability and simplicity. The GBM model assumes that the logarithm of stock prices follows a Brownian motion with drift, allowing for continuous price movements. In GBM, the stock price S_t at time t is modeled as:

$$S_t = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma B_t} \tag{2.1}$$

where μ, σ, and B_t are the return, volatility of the stock, and the value of Brownian motion at time t, respectively. Brownian motion (BM) B is a stochastic

process such that

1. $B_0 = 0$
2. if $0 \leq s_1 \leq t_1 \leq s_2 \leq t_2$, then $B_{t_1} - B_{s_1}$ and $B_{t_2} - B_{s_2}$ are independent.
3. $B_t - B_s \sim \mathcal{N}(0, t-s) \quad (\text{for } 0 \leq s \leq t)$
4. $t \to B_t$ is almost surely continuous

where B_t is the value of BM at time t. Therefore, when time is discretized, the stock price S_t at time t can be expressed as follows:

$$S_t = S_{t-1} e^{(\mu - \frac{1}{2}\sigma^2) + \sigma\epsilon_t} \tag{2.2}$$

where the error term $\epsilon_t \sim \mathcal{N}(0,1)$ and $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_t\}$ are mutually independent. The assumption of independence makes GBM inappropriate for predicting future stock prices. To address this issue, attempts have been made to predict stock prices using modeling approaches such as Hamilton regime-switching model [5], fractional Brownian motion (fBM) [11], and ARIMA [1]. However, these methods struggle to capture complex stock price patterns due to their simplicity.

2.2 Deep Learning-Based Stock Price Prediction Methods With advancement of deep neural networks, attempts to predict stock prices using the time series processing capabilities of recurrent neural networks have emerged [12], [14]. The attention mechanism has been used to enhance the performance of RNN-based models [13], [7], [3]. GCN or Transformer architectures have been used to leverage the correlations between stocks [9], [16], [8]. Methods to detect domain shifts that may occur during the evolution of stock time series and adapt to various market conditions have been also proposed [10], [17]. However, these methods struggle to learn universal stock price patterns since they fail to remove the unique characteristics of individual stocks from price data. In this work, we aim to learn universal stock price patterns by removing stock characteristics to improve the accuracy of stock price prediction.

3 Proposed Method

We propose RVN-UP (Return-Volatility Normalization for Learning Universal Stock Price Patterns), a novel method for accurate stock price prediction by learning universal stock price patterns.

3.1 Overview We aim to accurately predict future stock prices by learning universal stock price patterns. To achieve this goal, RVN-UP is designed to address the following challenges.

1. **Removing individual stock characteristics.** If stock price data contain individual stock characteristics, it becomes difficult to find universal stock price patterns shared by all stocks. However, existing methods learn from data that include individual stock characteristics. How can we remove individual stock characteristics from stock price data?
2. **Integrated stock price modeling.** The traditional stock price modeling method, geometric Brownian motion (GBM), cannot accurately predict stock prices due to the independence assumption that future prices are independent of past prices. How can we model stock prices by incorporating stock characteristics and pure patterns?

We address the challenges with the following ideas.

1. **Return-volatility normalization (Section 3.2).** We normalize stock price data using the stock characteristics: return, volatility, and price scale in order to remove them from the data. As a result, we learn universal stock price patterns shared by all stocks.
2. **Universal price pattern modeling (Section 3.3).** We propose a stock price modeling method by modifying GBM. We replace the Brownian motion (BM) part of GBM with a neural network that captures universal stock price patterns, thereby eliminating the independence assumption. Consequently, we model stock prices by integrating stock characteristics and the pure patterns.

3.2 Return-Volatility Normalization To learn universal stock price patterns shared by all stocks, individual stock characteristics must be removed from stock price data. To achieve this goal, we must first define the stock characteristics. In the widely used stock modeling method, GBM [2], stock characteristics are defined as return, volatility, and price scale. Observing Equation (2.2), the equation consists of the variables μ (return), σ (volatility), S_{t-1} (price scale), and ϵ_t (error term), among which μ, σ, and S_{t-1} represent stock characteristics. Among these, the error term represents values that do not reflect stock characteristics. The values of the error term over time form a pure pattern that is not influenced by stock characteristics. Therefore, we normalize the stock price by return, volatility, and price scale to obtain the value of the error term and use it as a feature for learning.

Return and Volatility Estimating. We first estimate the return and volatility of the stock. Since logarithmic daily return $\log \frac{S_t}{S_{t-1}} = \mu - \frac{1}{2}\sigma^2 + \sigma\epsilon_t \sim \mathcal{N}(\mu - \frac{1}{2}\sigma^2, \sigma^2)$, $\mu - \frac{1}{2}\sigma^2$ and σ are estimated as follows:

$$\widehat{\mu - \frac{1}{2}\sigma^2} = \mathbb{E}\left[\log \frac{S_t}{S_{t-1}}\right] \tag{3.3}$$

$$\hat{\sigma} = \text{Var}\left(\log \frac{S_t}{S_{t-1}}\right). \tag{3.4}$$

Obtaining Error Term. We obtain the error term by normalizing the logarithmic daily return using the estimated return and volatility:

$$\epsilon_t = \frac{1}{\hat{\sigma}}\left(\log \frac{S_t}{S_{t-1}} - \left(\widehat{\mu - \frac{1}{2}\sigma^2}\right)\right). \tag{3.5}$$

We normalize closing price according to the aforementioned method. We also want to normalize other price features (opening, highest, and lowest prices). However, we cannot normalize the price features using the same method, since we cannot precisely determine the time at which each price occurs.

To provide an example, by substituting the initial time in Equation (2.1) with $t - s$, the time at which the opening price occurs, we rewrite Equation (2.1) as equation about opening price at time $t - s$ (let t be the time at which the closing price occurs):

$$S_{t-s} = S_t e^{-\left(\mu - \frac{1}{2}\sigma^2\right)s - \sigma\epsilon_t^{open}}. \tag{3.6}$$

where ϵ_t^{open} is the error term and s is the time interval between the opening and closing prices. For clarity, we replace the BM part (B_s) with the error term (ϵ_t^{open}). ϵ_t^{open} can be determined if s is known since other variables ($S_{t-s}, S_t, \mu,$ and σ) are already known. However, determining the time interval between the opening and closing prices is not straightforward. We might consider setting s to 1 since the opening price is the immediate price following the previous day's closing price. On the other hand, due to price fluctuations during after-hours trading, the opening price may differ from the previous day's closing price, meaning that s cannot be set to 1. Thus, we estimate s by minimizing mean squared error as follows:

$$\hat{s} = \frac{\sum_t (\mu - \frac{1}{2}\sigma^2) \log \frac{S_t}{S_{t-s}}}{\sum_t (\mu - \frac{1}{2}\sigma^2)^2}. \tag{3.7}$$

The derivation process is in Appendix A.

As a result, we obtain ϵ_t^{open} by solving Equation (3.6):

$$\epsilon_t^{open} = \frac{1}{\hat{\sigma}}\left(\log \frac{S_t}{S_{t-s}} - \left(\widehat{\mu - \frac{1}{2}\sigma^2}\right)\hat{s}\right). \tag{3.8}$$

Highest price cannot be normalized as shown in Equation (3.5) since we do not know when it occurred during the day. By substituting the initial time in Equation (2.1) with $t - u$, the time at which the highest price occurs, we can rewrite Equation (2.1) as equation about highest price at time $t - u$ (let t be the time at which the closing price occurs):

$$S_{t-u} = S_t e^{-\left(\mu - \frac{1}{2}\sigma^2\right)u - \sigma\epsilon_t^{high}} \tag{3.9}$$

where ϵ_t^{high} is the error term and u is the time interval between the highest and closing prices. $\left|\mu - \frac{1}{2}\sigma^2\right| << \sigma$ since return is much smaller than volatility. Thus, we ignore the drift $-\left(\mu - \frac{1}{2}\sigma^2\right)u$:

$$S_{t-u} \approx S_t e^{-\sigma\epsilon_t^{high}}. \tag{3.10}$$

As a result, we normalize the highest price as follows:

$$\epsilon_t^{high} = \frac{1}{\hat{\sigma}}\left(\log \frac{S_t}{S_{t-u}}\right). \tag{3.11}$$

We also normalize the lowest price in the same way:

$$\epsilon_t^{low} = \frac{1}{\hat{\sigma}}\left(\log \frac{S_t}{S_{t-v}}\right) \tag{3.12}$$

where $t - v$ is the time at which the lowest price occurs.

3.3 Universal Price Pattern Modeling

We transformed raw price feature into a pure pattern, devoid of individual stock characteristics, through return-volatility normalization. We now need to recognize universal price pattern from the pure pattern and combine it back with individual stock characteristics. To achieve this, we replace the BM part of the GBM with a neural network. Due to the independence assumption of BM, GBM cannot predict future prices. However, by replacing the BM part with a neural network, the model gains predictive power. The neural network can be an existing time series prediction model or a stock price prediction model. By replacing the BM part with a neural network, Equation (2.2) changes as follows:

$$S_t = S_{t-1} e^{\left(\mu - \frac{1}{2}\sigma^2\right) + \sigma f_\theta(\mathbf{E})} \tag{3.13}$$

where θ is the neural network parameters, f_θ is the neural network,

Table 2: Summary of datasets.

Data	Stocks	Days	Dates
U.S.	224	7659	1992-08-04 to 2022-12-30
CN	171	2430	2011-01-06 to 2020-12-31
KR	471	5391	2001-01-03 to 2022-11-01

Table 3: An input feature vector ϵ_t at day t.

Feature	Calculation
ϵ_t^{open}	$\epsilon_t^{open} = open_t/close_t - 1$
ϵ_t^{high}	$\epsilon_t^{high} = high_t/close_t - 1$
ϵ_t^{low}	$\epsilon_t^{low} = low_t/close_t - 1$
ϵ_t^{close}	$\epsilon_t^{close} = close_t/close_{t-1} - 1$

$$\mathbf{E} = \begin{bmatrix} \epsilon_{t-w} & \epsilon_{t-w+1} & \cdots & \epsilon_{t-1} \\ \epsilon_{t-w}^{open} & \epsilon_{t-w+1}^{open} & \cdots & \epsilon_{t-1}^{open} \\ \epsilon_{t-w}^{high} & \epsilon_{t-w+1}^{high} & \cdots & \epsilon_{t-1}^{high} \\ \epsilon_{t-w}^{low} & \epsilon_{t-w+1}^{low} & \cdots & \epsilon_{t-1}^{low} \end{bmatrix} \tag{3.14}$$

is the return-volatility normalized feature matrix, and w is the window size.

Model Training. Since we want $f_\theta(\mathbf{E})$ to predict ϵ_t, the neural network is trained to minimize the following mean squared error loss:

$$L(\theta) = \mathbb{E}\left[\left(f_\theta(\mathbf{E}) - \epsilon_t\right)^2\right]. \tag{3.15}$$

4 Experiments

We present experimental results to answer the following research questions about our RVN-UP:

Q1. **Performance (Section 4.2).** Does RVN-UP outperform previous methods in stock price prediction?

Q2. **Universal Price Pattern Learning (Section 4.3).** Does RVN-UP identify universal patterns to predict the future prices of stocks it has never encountered before?

Q3. **Hyperparameter Sensitivity (Section 4.4).** How sensitive is RVN-UP to hyperparameters?

Q4. **Ablation Study (Section 4.5).** Does each module of RVN-UP contribute to improving the performance for stock price prediction?

4.1 Experimental Setup We present a dataset, competitors, and evaluation metrics for experiments on stock price prediction.

Datasets. We use three stock market datasets: U.S., CN, and KR markets. The summary of the datasets is presented in Table 2. The U.S. and KR datasets are used from [6]. The CN dataset is a new dataset that we collected from Chinese stock market. The goal is to predict the closing price on day $t+1$ given the price features up to day t.

Data Normalization. We generate an input vector $\boldsymbol{\epsilon_t}$ at day t for competitors using widely used price ratio-based normalization method [3], [16]. In contrast, RVN-UP exploits the normalization technique described in Section 3.2. The normalization method for competitors is shown in Table 3. ϵ_t^{open}, ϵ_t^{high}, and ϵ_t^{low} indicate the relative values of the opening, highest, and lowest prices in comparison to closing price at day t, respectively. ϵ_t^{close} indicates the relative value of the closing price at day t in comparison to closing price at day $t-1$.

Competitors. We compare RVN-UP with the following competitors.

- **MLP** is the simplest baseline with two hidden layers.
- **LSTM** [12] is a recurrent neural network designed to capture long-term dependencies in time series data.
- **ALSTM** [3] enhances the LSTM model by incorporating an attention mechanism, allowing it to focus on the most relevant time steps for improved stock price prediction.
- **DTML** [16] is the previous state-of-the-art model for stock price prediction, leveraging temporal and inter-stock correlations with global market context.

We compare RVN-UP, which employs DTML as a neural networks, against the baselines.

Evaluation Metrics. We conduct 10 experiments with different random seeds ranging from 0 to 9 and report the average across all cases. We evaluate the results of stock price prediction using information coefficient (IC) and rank information coefficient (RankIC). IC is the dominant metric in finance, which measures the linear relationship between the predicted prices and the actual prices. IC is calculated as:

$$IC = \frac{\mathrm{cov}(\hat{y}, y)}{\mathrm{std}(\hat{y})\mathrm{std}(y)} \tag{4.16}$$

where $\hat{y}$ and y are the predicted price and the actual price, respectively. RankIC measures the linear

Table 4: Stock prediction performance on unseen stocks of RVN-UP and competitors measured by Information Coefficient (IC), Rank Information Coefficient (RankIC), and Annualized Rate of Return (ARR). While other methods struggle to identify patterns in unseen stocks, RVN-UP demonstrates excellent performance on unseen stocks by learning universal patterns.

Method	U.S.			CN			KR		
	IC	RankIC	ARR	IC	RankIC	ARR	IC	RankIC	ARR
MLP	-	-	-%	-	-	-%	-	-	-%
LSTM [12]	-	-	-%	-	-	-%	-	-	-%
ALSTM [3]	-	-	-%	-	-	-%	-	-	-%
DTML [16]	0.0349	**0.0268**	11.53%	0.0093	0.0171	-2.30%	0.0113	0.0071	-10.81%
RVN-UP (proposed)	**0.0637**	0.0220	**24.70%**	**0.0353**	**0.0317**	**31.32%**	**0.0788**	**0.0878**	**260.49%**

relationship between the predicted stock ranks and the actual ranks, indicating the model's ability to predict relative stock performance.

We evaluate the portfolio's performance using the annualized rate of return (ARR). The portfolio is rebalanced at the close of each day to allocate a weight of 0.2 to the top 5 stocks with the highest predicted returns.

4.2 Performance Table 5 compares the performance of RVN-UP and the baselines for stock price prediction across three datasets. RVN-UP not only improves upon the performance of the previous state-of-the-art DTML but also achieves the state-of-the-art accuracy on all datasets through consistent improvements across all metrics compared to all other methods. RVN-UP learns only pure patterns by removing individual stock characteristics from price features, allowing it to identify universal patterns shared by all stocks more effectively.

4.3 Universal Price Pattern Learning Table 4 compares the performance of RVN-UP and the baselines on unseen stocks. We evaluate RVN-UP's ability to learn universal price patterns by testing its performance on stocks that were not included in the training set. We randomly split the stocks in the dataset into two halves, with one half used for training and the other half for testing. This approach is crucial for assessing how well RVN-UP generalizes to new, unseen data.

While the baselines struggle to recognize the price patterns of unseen stocks, RVN-UP successfully identifies the price patterns of these unfamiliar stocks. This success indicates that our approach to learning universal patterns is effective, as RVN-UP consistently identifies relevant price movements, even in unfamiliar stocks. The findings suggest that RVN-UP adapts more effectively to the diverse dynamics of stock markets by emphasizing learning of universal patterns.

4.4 Hyperparameter Sensitivity Figure 2 shows the comparison of hyperparameter sensitivity between RVN-UP and DTML. DTML is sensitive to hyperparameters due to its complexity. Return-volatility normalization technique of RVN-UP allows DTML to achieve maximum performance with minimal tuning.

4.5 Ablation Study We analyze the performance of RVN-UP and its variants by removing each of the two primary modules, as shown in Table 6:

- RVN-UP-RVN: RVN-UP without the return-volatility normalization (RVN)
- RVN-UP-NN: RVN-UP without the neural network (NN)

Each module contributes to improved prediction performance, with RVN-UP incorporating two modules achieving the highest IC and ARR. Our observations suggest that the neural network module NN is more crucial than RVN, as NN serves the role of finding universal price patterns. RVN removes individual stock characteristics from the price data to generate pure patterns.

5 Conclusion

We propose a novel framework, RVN-UP, for stock price prediction. RVN-UP learns universal price patterns shared by all stocks to accurately predict stock prices. RVN-UP learns universal patterns through two main ideas: a) removing price features by normalizing them with individual stock characteristics: return, volatility, and price scale, and b) presenting a stock price modeling approach modified from GBM to integrate universal pattern recognition through neural networks and individual stock characteristics. RVN-UP achieves the state-of-the-art performance in stock price prediction on three datasets collected from U.S., China, and Korea,

Table 5: Stock prediction performance of RVN-UP and competitors measured by Information Coefficient (IC), Rank Information Coefficient (RankIC), and Annualized Rate of Return (ARR). RVN-UP shows the best performance across all datasets in all three evaluation metrics, demonstrating up to 12.8 times higher IC and 4.5 times higher ARR compared to its competitors.

Method	U.S.			CN			KR		
	IC	RankIC	ARR	IC	RankIC	ARR	IC	RankIC	ARR
MLP	-	-	-%	-	-	-%	-	-	-%
LSTM [12]	-	-	-%	-	-	-%	-	-	-%
ALSTM [3]	-	-	-%	-	-	-%	-	-	-%
DTML [16]	0.0349	0.0248	18.20%	0.0039	0.0429	-14.44%	0.0146	0.0097	-10.63%
RVN-UP (proposed)	**0.0673**	**0.0256**	**26.10%**	**0.0498**	**0.0370**	**68.06%**	**0.0853**	**0.0785**	**301.74%**

Figure 2: Comparison of hyperparameter sensitivity between RVN-UP and DTML.

Table 6: An ablation study of RVN-UP on U.S. dataset. RVN and NN represent the return-volatility normalization and neural network, respectively.

Method	IC	RankIC	ARR
RVN-UP-RVN	0.0349	**0.0268**	11.53%
RVN-UP-NN	-	-	-
RVN-UP	**0.0637**	0.0220	**24.70%**

with improvements of up to 12.8 times in information coefficient and 4.5 times in annualized rate of return compared to the previous state-of-the-art models. Additionally, unlike existing methods, RVN-UP successfully recognizes price patterns of stocks not seen in the training set. Future research will include extending RVN-UP to remove individual stock characteristics from technical indicators (such as moving averages, relative strength index, etc.) beyond basic price features.

References

[1] D. Asteriou and S. G. Hall, *Arima models and the box–jenkins methodology*, Applied Econometrics, 2 (2011), pp. 265–286.

[2] T. Bjork, *Arbitrage theory in continuous time*, (2019).

[3] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, *Enhancing stock movement prediction with adversarial training.*, in IJCAI, vol. 19, 2019, pp. 5843–5849.

[4] I. Gat, H. Aronowitz, W. Zhu, E. Morais, and R. Hoory, *Speaker normalization for self-supervised speech emotion recognition*, in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 7342–7346.

[5] J. D. Hamilton, *Regime switching models*, Palgrave Macmillan UK, London, 2010, pp. 202–209.

[6] J. Jeon, J. Park, C. Park, and U. Kang, *Frequant: A reinforcement-learning based adaptive portfolio optimization with multi-frequency decomposition*, in Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 1211–1221.

[7] H. Li, Y. Shen, and Y. Zhu, *Stock price prediction using attention-based multi-input lstm*, in Asian conference on machine learning, PMLR, 2018, pp. 454–469.

[8] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, and S. Huang, *Master: Market-guided stock transformer for stock price forecasting*, in Proceedings of the AAAI

Conference on Artificial Intelligence, vol. 38, 2024, pp. 162–170.
[9] W. Li, R. Bao, K. Harimoto, D. Chen, J. Xu, and Q. Su, *Modeling the stock relation with graph network for overnight stock movement prediction*, in Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence, 2021, pp. 4541–4547.
[10] H. Lin, D. Zhou, W. Liu, and J. Bian, *Learning multiple stock trading patterns with temporal routing adaptor and optimal transport*, in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 1017–1026.
[11] B. B. Mandelbrot and J. W. Van Ness, *Fractional brownian motions, fractional noises and applications*, SIAM review, 10 (1968), pp. 422–437.
[12] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, *Stock market's price movement prediction with lstm neural networks*, in 2017 International joint conference on neural networks (IJCNN), Ieee, 2017, pp. 1419–1426.
[13] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, *A dual-stage attention-based recurrent neural network for time series prediction*, in Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, p. 2627–2633.
[14] M. A. I. Sunny, M. M. S. Maswood, and A. G. Alharbi, *Deep learning-based stock price prediction using lstm and bi-directional lstm model*, in 2020 2nd novel intelligent and leading emerging sciences conference (NILES), IEEE, 2020, pp. 87–92.
[15] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, *A regression approach to speech enhancement based on deep neural networks*, IEEE/ACM transactions on audio, speech, and language processing, 23 (2014), pp. 7–19.
[16] J. Yoo, Y. Soun, Y.-c. Park, and U. Kang, *Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2037–2045.
[17] D. Zhan, Y. Dai, Y. Dong, J. He, Z. Wang, and J. Anderson, *Meta-adaptive stock movement prediction with two-stage representation learning*, in Proceedings of the 2024 SIAM International Conference on Data Mining (SDM), SIAM, 2024, pp. 508–516.

A Estimating s

Rewriting Equation (3.6) gives us the following:

$$\log \frac{S_t}{S_{t-s}} = \left(\mu - \frac{1}{2}\sigma^2\right) s + \sigma \epsilon_t^{open}. \tag{A.1}$$

We need to minimize the following squared error:

$$\sum_t \left(\left(\mu - \frac{1}{2}\sigma^2\right) s - \log \frac{S_t}{S_{t-s}} \right)^2. \tag{A.2}$$

Equation (A.2) is convex with respect to s, so it attains its minimum at the point where the derivative with respect to s is equal to zero:

$$\begin{aligned} 0 &= \frac{d}{ds} \sum_t \left(\left(\mu - \frac{1}{2}\sigma^2\right) s - \log \frac{S_t}{S_{t-s}} \right)^2 \\ &= \sum_t 2 \left(\mu - \frac{1}{2}\sigma^2\right) \left(\left(\mu - \frac{1}{2}\sigma^2\right) s - \log \frac{S_t}{S_{t-s}} \right). \end{aligned} \tag{A.3}$$

Solving Equation (A.3), we obtain:

$$s = \frac{\sum_t (\mu - \frac{1}{2}\sigma^2) \log \frac{S_t}{S_{t-s}}}{\sum_t (\mu - \frac{1}{2}\sigma^2)^2}. \tag{A.4}$$