CSPO: Cross-Market Synergistic Stock Price Movement Forecasting with Pseudo-volatility Optimization

Sida Lin*‡
The Chinese University of Hong Kong, Shenzhen
China
sidalin1@link.cuhk.edu.cn

Yankai Chen*
Cornell University
Ithaca, United States
yankaichen@acm.org

Yiyan Qi
International Digital Economy Academy (IDEA)
Shenzhen, China
qiyiyan@idea.edu.cn

Chenhao Ma†
The Chinese University of Hong Kong, Shenzhen
China
machenhao@cuhk.edu.cn

Bokai Cao
The Hong Kong University of Science and Technology (Guangzhou)
China
mabkcao@connect.hkust-gz.edu.cn

Yifei Zhang
Nanyang Technological University
Singpore
yifei.zhang@ntu.edu.sg

Xue Liu
McGill University
Montreal, Canada
xueliu@cs.mcgill.ca

Jian Guo†
International Digital Economy Academy (IDEA)
Shenzhen, China
guojian@idea.edu.cn

Abstract

The stock market, as a cornerstone of the financial markets, places forecasting stock price movements at the forefront of challenges in quantitative finance. Emerging learning-based approaches have made significant progress in capturing the intricate and ever-evolving data patterns of modern markets. With the rapid expansion of the stock market, it presents two characteristics, i.e., *stock exogeneity* and *volatility heterogeneity*, that heighten the complexity of price forecasting. Specifically, while stock exogeneity reflects the influence of external market factors on price movements, volatility heterogeneity showcases the varying difficulty in movement forecasting against price fluctuations. In this work, we introduce the framework of Cross-market Synergy with Pseudo-volatility Optimization (CSPO). Specifically, CSPO implements an effective deep neural architecture to leverage external futures knowledge. This enriches stock embeddings with cross-market insights and thus enhances the CSPO's predictive capability. Furthermore, CSPO incorporates *pseudo-volatility* to model stock-specific forecasting confidence, enabling a dynamic adaptation of its optimization process to improve accuracy and robustness. Our extensive experiments, encompassing industrial evaluation and public benchmarking, highlight CSPO's superior performance over existing methods and effectiveness of all proposed modules contained therein.

CCS Concepts

• **Computing methodologies** → **Neural networks**; • **Mathematics of computing** → **Time series analysis**.

Keywords

Stock price movement forecasting; Bayesian Neural Networks

ACM Reference Format:
Sida Lin*‡, Yankai Chen*, Yiyan Qi, Chenhao Ma†, Bokai Cao, Yifei Zhang, Xue Liu, and Jian Guo†. 2025. CSPO: Cross-Market Synergistic Stock Price Movement Forecasting with Pseudo-volatility Optimization. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3701716.3715216

1 Introduction

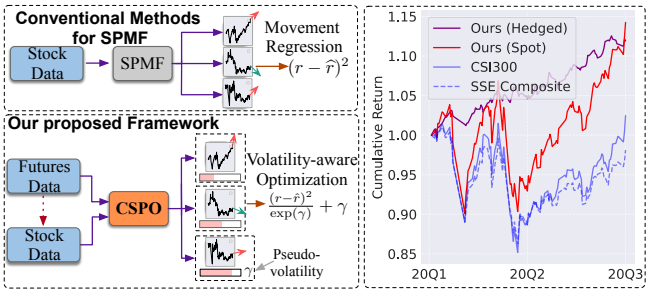
Stock price movement forecasting, with the goal of predicting future upward/downward price trends, is a core task in quantitative investment with significant research attention [34, 58, 69]. Traditional approaches rely on manually constructed features from basic financial indicators, such as moving averages, price-to-earnings ratios, and trading volumes [3, 20, 23]. While these features are with good interpretability, they may not capture the complex, dynamic, and nonlinear data patterns in markets.

In recent years, machine learning and deep learning methods have revolutionized the area by enabling models to learn directly from raw financial data. Machine learning algorithms such as decision trees [48, 49] and ensemble methods [7, 24, 62, 71] have been effectively applied to identify patterns with better robustness compared to traditional statistical methods. Deep learning models,

*Equal Contribution. † Corresponding Authors.
‡This work was conducted during his internship at IDEA Research.

WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/25/04
https://doi.org/10.1145/3701716.3715216



(a) Illustration of different methods. (b) Performance comparison.

Figure 1: (a) Stock price movement forecasting (SPMF). (b) Portfolio yield comparison between baselines, i.e., CSI300 Index and SSE Composite, and our method to use CSPO results as alpha-factors for hedged and spot return.

particularly neural networks tailored for handling time-series financial data [15, 53, 63], have made great strides in capturing temporal dependencies. These methods are highly responsive to the dynamic nature of financial markets. They have demonstrated strong performance in extracting localized price trend patterns [21, 51], thereby boosting the understanding of the market behaviors.

As forecasting price movements appear to be increasingly difficult, the market has become more sophisticated and diverse, exhibiting two characteristics: *stock exogeneity* and *volatility heterogeneity*. Specifically, (1) in the increasingly interconnected global economy, financial markets do not operate in isolation, where the stock market is affected by external events, e.g., policy changes, economic indicators, and geopolitical developments. Beyond the conventional methods solely based on stock market data [34, 63], recent deep forecasting models have started to integrate auxiliary information, such as exchanges [5], sales [65], and earnings calls [38, 43, 61], for performance improvement. This shows the promising potential of alleviating local limitations by incorporating spillover effects from other markets [36]. (2) Different stocks with varying company-specific fundamentals respond differently to market stimuli, exhibiting varied volatility patterns. Since volatility indicates the stability of stock price fluctuations, higher volatility thus increases the difficulty of price movement prediction. Although only a few methods [31, 72] avoid assuming stock homogeneity, they primarily consider volatility to adjust the predicted price. However, we argue that such volatility essentially presents the prediction confidence, not just adjustments to price values. This approach may still overlook potential prediction deviations in model optimization, underscoring the need for new non-uniform learning strategies tailored to volatility heterogeneity.

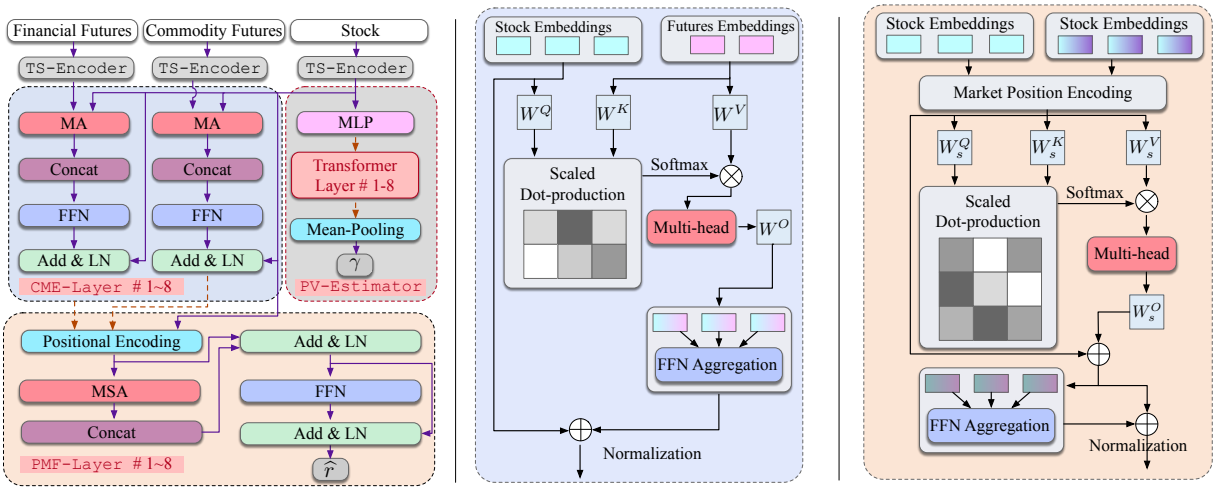
Aligning with the aforementioned characteristics, we push forward the study of price movement forecasting by proposing Cross-market Synergy with Pseudo-volatility Optimization framework (CSPO). We provide a high-level illustration of CSPO in Figure 1(a). (1) Firstly, to explore stock exogeneity, we propose incorporating *futures market* information for forecasting guidance. Unlike other additional information sources, futures markets are inherently forward-looking, as they represent contracts to buy or sell assets on future dates. This provides practical insights into market expectations about future movements, which can be directly relevant for forecasting stock prices. To achieve this, we design a transformer-based deep neural architecture, namely *Bi-level Dense Pricing Transformer* (`BDP-Former`). As the name suggests, `BDP-Former` progressively conducts cross-market knowledge synergy followed by price movement prediction. This emulates real-world trading practices, where traders consider futures-to-stock and stock-to-stock correlations in their historical data and future trends [13]. Therefore, it leverages a broader spectrum of market data to capture the complex interdependencies and enhances the model's predictive capability accordingly. (2) Secondly, for volatility heterogeneity, we propose to study it within the model optimization process. In practice, traders consider not only the macro market factors but also the stability of individual stock prices. Intuitively, volatility reflects price stability and thus indicates the confidence level in price forecasting. This motivates us to differentiate the loss contributions during optimization to minimize errors associated with low confidence predictions. To adapt to the time-varying and stock-specific volatility, we introduce the concept of *pseudo-volatility* that can be estimated alongside our CSPO framework. Then the estimated pseudo-volatility is incorporated into our final objective function. Compared to the conventional loss design, e.g., mean squared error, where they assume the equal loss contribution, our objective explicitly accounts for the loss variance inherent in different stock price predictions. This approach provides a more fine-grained learning optimization process, ultimately leading to more accurate price movement forecasting.

To comprehensively evaluate the performance of CSPO, we conduct extensive experiments on several real-world stock market datasets. We first evaluate it in the industrial setting, where we leverage detailed proprietary backtesting with different evaluation strategies and metrics. As shown in Figure 1(b), our methods achieve more satisfactory yield curves on the CSI300 Index, i.e., a widely evaluated stock market dataset. We also provide a detailed public benchmarking with several existing models and empirical analyses of CSPO. The results further demonstrate not only the superiority of our framework over baselines but also the effectiveness of all proposed module designs contained within. To summarize, we have made the following threefold contributions:

- We incorporate external futures information, and propose a deep neural architecture `BDP-Former` for effective cross-market knowledge synergy and enhanced stock price movement forecasting.
- We introduce the estimation of *pseudo-volatility* to capture price movement stability and forecasting confidence, which is further leveraged to differentiate their diversities in model optimization.
- Extensive experiments on both industrial evaluation and public benchmarking demonstrate the effectiveness of our proposed framework CSPO.

2 Problem Formulation

Assume that a stock market consists of k_s stock assets. Let t be the look-back window of the historical data. At different time steps, each stock asset exhibits unique states, such as prices, market shares, etc. Thus we use d' features to characterize each asset. We first introduce the data format for the stock market time series:



(a) A high-level workflow. (b) `CME-Layer` structure. (c) `PMF-Layer` structure.

Figure 2: An illustration of our framework overview (best view in color). In `CME-Layer`, the notations, e.g., W^Q, generalize to both cases of commodity futures and financial futures, e.g., W_c^Q and W_e^Q.

Definition 1 (Stock Market Time Series). Let $\boldsymbol{S} \in \mathbb{R}^{t \times k_s \times d'}$ denote the 3-dimensional tensor representing the stock market time series data. $\boldsymbol{T}_i^s$ denotes the market snapshot with all stocks at the time step i. $\boldsymbol{S}$ is formally defined as:

$$\boldsymbol{S} = \left[\boldsymbol{T}_1^s, \boldsymbol{T}_2^s, \ldots, \boldsymbol{T}_t^s\right], \text{ where } \boldsymbol{T}_i^s \in \mathbb{R}^{k_s \times d'}. \tag{1}$$

For the futures markets, we consider both commodity and financial futures data:

Definition 2 (Futures Market Time Series). Let $\boldsymbol{C} \in \mathbb{R}^{t \times k_c \times d'}$ and $\boldsymbol{E} \in \mathbb{R}^{t \times k_e \times d'}$ respectively represent the *commodity futures* and *financial futures* time series. $\boldsymbol{T}_i^c$ and $\boldsymbol{T}_i^e$ represent the snapshot of k_c commodity futures and k_e financial futures at time step i. These futures market time series are formulated as:

$$\begin{aligned} \boldsymbol{C} &= \left[\boldsymbol{T}_1^c, \boldsymbol{T}_2^c, \ldots, \boldsymbol{T}_t^c\right], \text{ where } \boldsymbol{T}_i^c \in \mathbb{R}^{k_c \times d'}, \\ \boldsymbol{E} &= \left[\boldsymbol{T}_1^e, \boldsymbol{T}_2^e, \ldots, \boldsymbol{T}_t^e\right], \text{ where } \boldsymbol{T}_i^e \in \mathbb{R}^{k_e \times d'}. \end{aligned} \tag{2}$$

The problem addressed in this paper is defined as follows:

Problem 1 (Stock Price Movement Forecasting). We aim to construct a deep neural predictive model, f, inputting t-size look-back window of data for stock market $\boldsymbol{S}$, commodity futures market $\boldsymbol{C}$, and financial futures market $\boldsymbol{E}$, and then predict the stock price movements, i.e., $\widehat{\boldsymbol{r}}_{t+1} \in \mathbb{R}^{k_s}$, on the market at the next time step, i.e., $\widehat{\boldsymbol{r}}_{t+1} = f_{1:t}(\boldsymbol{S}, \boldsymbol{C}, \boldsymbol{E})$. Please notice that, $\widehat{\boldsymbol{r}}_{t+1}$ could be positive or negative; for stock prices at the t-th time step $\boldsymbol{P}_t \in \mathbb{R}^{k_s}$, $\widehat{\boldsymbol{r}}_{t+1} = \frac{\boldsymbol{P}_{t+1}}{\boldsymbol{P}_t} - \mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^{k_s}$. We use $\widehat{\boldsymbol{r}}$ to refer to $\widehat{\boldsymbol{r}}_{t+1}$ for brevity whenever unambiguous.

3 CSPO Framework

3.1 Overview

In complex financial markets, futures and stocks are often interrelated, with mutual influences on pricing; our approach aims to capture these relationships for enhanced stock price movement forecasting. Firstly, futures and stock time series data are encoded, as described in § 3.2. Then in § 3.3, we propose the *Bi-level Dense Pricing Transformer* architecture to capture both futures-stock and stock-stock correlations for stock price movement forecasting. In § 3.4, we propose the *pseudo-volatility* to associate with stock price prediction confidence, which improves model optimization through our volatility-aware objective function. The overall framework is illustrated in Figure 2.

3.2 Market Time Series Encoding

For the input market time series data, e.g., stocks $\boldsymbol{S}$, commodity futures $\boldsymbol{C}$, and financial futures $\boldsymbol{E}$, a standard procedure is to encode them with d-dimensional representations for further model learning. Therefore, we follow recent stock price prediction models [17, 53, 57] to adopt the encoder, denoted by `TS-Encoder`, for market information encoding:

$$\begin{aligned} \boldsymbol{S}^* &= \texttt{TS-Encoder}(\boldsymbol{S}), \text{ where } \boldsymbol{S}^* \in \mathbb{R}^{k_s \times d}, \\ \boldsymbol{C}^* &= \texttt{TS-Encoder}(\boldsymbol{C}), \text{ where } \boldsymbol{C}^* \in \mathbb{R}^{k_c \times d}, \\ \boldsymbol{E}^* &= \texttt{TS-Encoder}(\boldsymbol{E}), \text{ where } \boldsymbol{E}^* \in \mathbb{R}^{k_e \times d}. \end{aligned} \tag{3}$$

3.3 Bi-level Dense Pricing Transformer

In this work, we draw inspiration from real-world practices where traders usually aggregate multiple sources of market information for stock future pricing. We thus introduce our *Bi-level Dense Pricing Transformer* structure, i.e., `BDP-Former`, for effective market information fusion and stock price forecasting:

$$\widehat{\boldsymbol{r}} = \texttt{BDP-Former}(\boldsymbol{S}^*, \boldsymbol{C}^*, \boldsymbol{E}^*). \tag{4}$$

`BDP-Former` mainly consists of two levels of stacked layers, i.e., *Cross-market Embedding Layer* and *Price Movements Forecasting Layer*. Generally, the first level of the learning layer captures latent relationships between futures and stocks, propagating futures-stock

information to enrich stock embeddings with cross-market positional knowledge. The second level then takes these aggregated stock representations as input to model stock-stock correlations, which ultimately enables stock price movement forecasting.

3.3.1 ***Cross-market Embedding Layer.*** In practice, futures and stock markets possess interconnections that share correlations in historical information and may impact future price movements. To leverage these futures-stock connections, we introduce a dense connectivity module between futures and stocks, designed to aggregate diverse information from futures markets into stock embeddings. In our work, it is stacked by 8 layers of *Cross-market Embedding Layer* (`CME-Layer`) as follows:

$$\overline{S}_c = \texttt{CME-Layer}^{[8]}(S^*, C^*), \text{ where } \overline{S}_c \in \mathbb{R}^{k_s \times d}. \tag{5}$$

Here stock embeddings are derived from the commodity futures. We use commodity futures as the example for following explanations.

To implement `CME-Layer`, we leverage the Self-attention (SA) mechanism [52] followed by the simplified forward networks. Specifically, we weigh the futures-stock connectivity with the scaled dot-product as follows:

$$\text{SA}(S^*, C^*) = \frac{1}{\sqrt{d}} \text{Softmax}\left(S^* \boldsymbol{W}_c^Q \cdot (C^* \boldsymbol{W}_c^K)^\intercal\right) \cdot C^* \boldsymbol{W}_c^V. \tag{6}$$

$\boldsymbol{W}_c^Q, \boldsymbol{W}_c^K$, and $\boldsymbol{W}_c^V$ are three matrices with $\mathbb{R}^{d \times d}$. Then `CME-Layer` further concatenates multiple heads (MA) as follows:

$$\text{MA}(S^*, C^*) = \text{Concat}(\text{SA}_1^\intercal, \text{SA}_2^\intercal, \ldots, \text{SA}_8^\intercal) \cdot \boldsymbol{W}_c^O. \tag{7}$$

In our work, the matrix $\boldsymbol{W}_c^O \in \mathbb{R}^{8d \times d}$ is for transformation. Then we directly apply the Feed Forward Network (FFN) and residual connection with Layer Normalization (LN) as follows:

$$S_c^{[l+1]} = \text{LN}\Big(S_c^{[l]} + \text{FFN}\big(\text{MA}(S_c^{[l]}, C^*)\big)\Big), \tag{8}$$

where $S_c^{[1]}$ is initialized by S^* and the output $\overline{S}_c = S_c^{[8]}$. In our work, FFN is implemented with two linear layers and ReLU activation. Similarly, to aggregate financial futures information, the stock can be encoded as $\overline{S}_e = \texttt{CME-Layer}^{[8]}(S^*, E^*)$. Both $\overline{S}_c$ and $\overline{S}_e$ play a crucial role in capturing stock information based on their relationships with commodity and financial futures markets. Therefore, we integrate them as the *market position encoding*, which serves as the input for the subsequent *Price Movement Forecasting Layer*.

3.3.2 ***Price Movement Forecasting Layer.*** The obtained embeddings $\overline{S}_c$ and $\overline{S}_e$ encapsulate rich futures-stock information, revealing the stocks' market positional knowledge within complex financial environments. Therefore, the stock representations can be further updated as follows:

$$S^+ = S^* + \overline{S}_c + \overline{S}_e. \tag{9}$$

Based on S^+, we thus further implement our second level of modules, i.e., the 8 layers of *Price Movement Forecasting Layer* (`PMF-Layer`):

$$\widehat{\boldsymbol{r}} = \texttt{PMF-Layer}^{[8]}(S^+). \tag{10}$$

`PMF-Layer` attentively captures the stock-stock correlations and predicts the stock price movement. Concretely, we implement the following Multi-head Stock Attention (MSA):

$$\text{MSA}(S^+) = \text{Concat}(\text{SSA}_1^\intercal, \text{SSA}_2^\intercal, \cdots, \text{SSA}_8^\intercal) \cdot \boldsymbol{W}_s^O, \tag{11}$$

where Single-head Stock Attention (SSA) is implemented as follows:

$$\text{SSA}(S^+) = \frac{1}{\sqrt{d}} \text{Softmax}\left(S^+ \boldsymbol{W}_s^Q \cdot (S^+ \boldsymbol{W}_s^K)^\intercal\right) \cdot S^+ \boldsymbol{W}_s^V. \tag{12}$$

Here $\boldsymbol{W}_s^Q, \boldsymbol{W}_s^K, \boldsymbol{W}_s^V$ are transformation matrices with $\mathbb{R}^{d \times d}$, and $\boldsymbol{W}_s^O \in \mathbb{R}^{8d \times d}$. We then follow the standard forward procedure [52] with the FFN and LN as follows:

$$S^{+[l+1]} = \text{LN}\big(\widetilde{S^+}^{[l]} + \text{FFN}(\widetilde{S^+}^{[l]})\big), \tag{13}$$

where $\widetilde{S^+}^{[l]}$ is obtained via:

$$\widetilde{S^+}^{[l]} = \text{LN}\big(S^{+[l]} + \text{MSA}(S^{+[l]})\big). \tag{14}$$

Finally, the movements $\widehat{\boldsymbol{r}}$ are derived by adopting the linear transformation to the output $S^{+[8]}$. In summary, our Bi-level Dense Pricing Transformer captures both the futures-to-stock and stock-wise correlations, enabling cross-market synergistic approach to stock price forecasting.

3.4 Pseudo-volatility Optimization

Due to varying positions in the stock market, different stocks exhibit differing levels of revenue-generating capability and resilience to market risks. These factors result in their distinct levels of stock price volatility. Stock volatility indicates the stability of stock price fluctuations and uncertainty in predictions. Intuitively, higher volatility often signifies greater uncertainty in price forecasting. Therefore, in this work, we are motivated to capture this concept and propose learning the "*pseudo-volatility*" for more accurate stock price forecasting.

3.4.1 ***Pseudo-volatility Estimation.*** To capture such volatility, one possible solution is to leverage Bayesian deep learning [42, 54], which originally offers a practical framework for modeling uncertainty [25]. Inspired by these works, we propose estimating the stock pseudo-volatility within the Bayesian framework to have the following deep neural architectures, namely `PV-Estimator`:

$$\boldsymbol{\gamma} = \texttt{PV-Estimator}(S), \text{ where } \boldsymbol{\gamma} \in \mathbb{R}^{k_s \times d}. \tag{15}$$

S is the raw stock market data and $\boldsymbol{\gamma}$ is the estimated pseudo-volatility. Our designed `PV-Estimator` differs from regular deterministic neural networks by incorporating volatility modeling and their variational inference. Specifically, we first process $\boldsymbol{S}$ via a two-layer MLP with ReLU activation, denoted by $\texttt{MLP}^{[2]}$, as follows:

$$\boldsymbol{V} = \texttt{MLP}^{[2]}(\boldsymbol{S}), \text{ where } \boldsymbol{V} \in \mathbb{R}^{t \times k_s \times d}. \tag{16}$$

Then we pass it through the eight-layer vanilla Transformer [52], denoted by $\texttt{Trm-layer}^{[8]}$:

$$\widetilde{\boldsymbol{V}} = \texttt{Trm-Layer}^{[8]}(\boldsymbol{V}), \text{ where } \widetilde{\boldsymbol{V}} \in \mathbb{R}^{t \times k_s \times d}. \tag{17}$$

Then the pseudo-volatility $\boldsymbol{\gamma}$ is empirically estimated with the mean-pooling operation as:

$$\boldsymbol{\gamma} = \texttt{Mean-Pooling}(\widetilde{\boldsymbol{V}}), \text{ where } \boldsymbol{\gamma} \in \mathbb{R}^{k_s \times d}. \tag{18}$$

In our work, we utilize Monte Carlo dropout throughout Eqn.'s (16) and (17) to achieve the variational volatility estimation. Intuitively, $\boldsymbol{\gamma}$ is calculated via an independent computational pipeline mainly to capture the hidden volatility information from raw market data in a less biased manner. Since stock assets exhibit varying levels of

volatility, and given that a higher value of γ indicates a higher uncertainty, the model should adapt its learning process accordingly. Along with the forecasted price discussed in the previous section, we propose a volatility-aware learning objective to eventually enable a more fine-grained optimization approach as follows.

3.4.2 ***Volatility-aware Regression Optimization.*** In conventional learning paradigms, regression objectives like mean squared error (MSE) are typically used. However, since we consider pseudo-volatility to distinguish stocks with varying prediction confidences, we incorporate this knowledge into model optimization. Intuitively, higher pseudo-volatility γ indicates lower confidence in price forecasting and should therefore reduce its contribution to the accumulated loss. A straightforward way to achieve this is by:

$$\mathcal{L} = \frac{1}{k_s} \sum \frac{(\boldsymbol{r} - \widehat{\boldsymbol{r}})^2}{\boldsymbol{\gamma}}, \tag{19}$$

where $\boldsymbol{r}$ is the ground-truth prices of all k_s stocks at the $(t+1)$-th time step. Eqn. (19) differentiates the standard MSE where the MSE essentially assumes equal variance for all stock samples.

However, Eqn. (19) still have some inadequacies. Firstly, during the loss minimization of Eqn. (19), it may easily minimize the loss by optimizing γ into negative values, which is inappropriate for γ as it should be a positive value. To fix this issue, we simply apply the numerical scaling with exponentiation as:

$$\mathcal{L} = \frac{1}{k_s} \sum \frac{(\boldsymbol{r} - \widehat{\boldsymbol{r}})^2}{\exp(\boldsymbol{\gamma})}. \tag{20}$$

Secondly, we want to reward the stock samples with low pseudo-volatility. However, our design may not perfectly align with Eqn. (20) as it may also maximize γ values, rather than solely optimizing the discrepancy between $\boldsymbol{r}$ and $\widehat{\boldsymbol{r}}$, which may disturb the optimization direction. Therefore, we update it with the following regularization:

$$\mathcal{L} = \frac{1}{k_s} \sum \frac{(\boldsymbol{r} - \widehat{\boldsymbol{r}})^2}{\exp(\boldsymbol{\gamma})} + \boldsymbol{\gamma}. \tag{21}$$

Lastly, due to the stochastic process of variational volatility estimation in `PV-Estimator`, we further propose using ensembling to stabilize the pseudo-volatility estimation and optimization:

$$\mathcal{L} = \frac{1}{k_s H} \sum_{h=1}^{H} \sum \left(\frac{(\boldsymbol{r} - \widehat{\boldsymbol{r}})^2}{\exp(\boldsymbol{\gamma}_h)} + \boldsymbol{\gamma}_h \right). \tag{22}$$

γ_h is output from an independent `PV-Estimator`. In our work, setting $H = 2$ already achieves satisfactory performance. As our empirical analysis in § 4.3.3 demonstrates, compared to our initial design in Eqn. (19), Eqn. (22) eventually provides more appropriate loss scaling, balanced optimization directions, and improved movement forecasting performance.

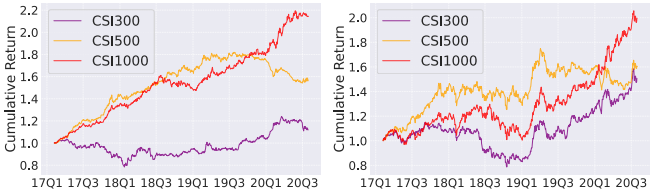
4 Experiments

We aim to answer the following research questions:

- **RQ1**: How does our model help in real-world trading scenarios to enhance proprietary profitability?
- **RQ2**: How does our model perform compared to existing models on real-world datasets?
- **RQ3**: How to systematically evaluate designs within CSPO?

Table 1: Data statistics for proprietary trading evaluation. "#" denotes the size. "C-futures" and "F-futures" denote commodity and financial futures. We use quarter time for data splitting, e.g., "08Q1" means the first quarter of 2008. "Ins." and "Trans." denote the "Instruments" and "Transaction days".

Time / Asset	Task 1				Task 2			
	Training 08Q1-16Q4		Evaluation 17Q1-20Q3		Training 17Q1-19Q2		Evaluation 19Q3-20Q3	
	#Ins.	#Trans.	#Ins.	#Trans.	#Ins.	#Trans.	#Ins.	#Trans.
C-futures	2,770	2,190	2,243	871	1,588	606	1,149	266
F-futures	424	2,190	344	871	255	606	142	266
CSI300	300	2,190	300	871	300	606	300	266
CSI500	500	2,190	500	871	500	606	500	266
CSI1000	1,000	2,190	1,000	871	1,000	606	1,000	266



(a) Hedged return curve. **(b) Spot return curve.**

Figure 3: Portfolio yield curves of our stock trading executor.

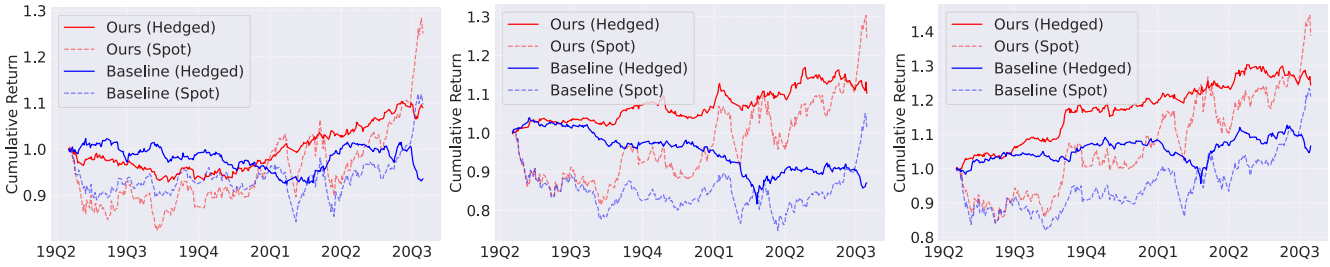
4.1 Proprietary Backtesting Evaluation (RQ1)

4.1.1 ***Overview.*** To assess how our model supports real-world trading, we introduce two evaluation approaches:

- **Task 1: trading executor**. We directly follow our model's predicted price movement as signals to trigger stock trades, and then integrate these trades into our internal platform for backtesting.
- **Task 2: alpha factor producer.** The other approach incorporates the model's output features as additional alpha factors to enhance our holistic trading strategies. We then evaluate whether these factors can contribute to improved system performance in the general portfolio investment.

4.1.2 ***Evaluation Data.*** We use real-world financial market data, i.e., futures and stocks, for model training and evaluation. We collect the daily frequency futures data, i.e., commodity and financial futures, from the Chinese futures market. For stock market data, we include two most frequently used index data, i.e., CSI300 and CSI500, and one most diverse index, i.e., CSI1000, from Shanghai Stock Exchange and the Shenzhen Stock Exchange. CSI300 and CSI500 are capitalization-weighted stock market indexes designed to replicate the performance of the top traded 300 and 500 stocks. And CSI1000 focuses on small-cap companies. We collect all data within 2008-2020 and use the data split of 2008-2017 for training and 2018-2020 for evaluation. Data statistics are reported in Table 1.

4.1.3 ***Evaluation Metrics.*** We introduce the following series of evaluation metrics: (1) Annualized Return (AR), (2) Winning Rate (WR), (3) Sortino Ratio (SoR), (4) Sharpe Ratio (ShR), (5) Maximum Drawdown (MD), (6) Maximum Drawdown Duration (MD-D), (7) Turnover Rate (TR). *Note that for the first four metrics, a higher value indicates better performance, whereas for the last three metrics, a lower value suggests a more favorable outcome.* Due to page limit, we explain them in detail in Appendix A.1 with evaluation configuration details in Appendix A.2.



(a) CSI300 yield curve. (b) CSI500 yield curve. (c) CSI1000 yield curve.

Figure 4: Yield curves of portfolio investments with (in warm colors) or without (in cold colors) our additional Alpha factors.

Table 2: Market backtesting of stock trading executor. (1) The strategy refers to whether it is for "Hedged Return" or "Spot Return". (2) The symbols ↑ and ↓ denote cases where higher and lower values, respectively, indicate better performance. (3) Colors indicate better-performing cases, i.e., better.

Data	Strategy	AR ↑	WR ↑	SoR ↑	ShR ↑	MD ↓	MD-D ↓	TR ↓
CSI300	Hedged	0.0336	0.5071	0.5401	0.3205	0.2544	196	0.1057
	Spot	0.1449	0.5259	1.1547	0.7002	0.3617	271	
CSI500	Hedged	0.1602	0.5112	3.1518	1.6386	0.2783	224	0.0357
	Spot	0.1718	0.5206	1.4232	0.8412	0.3366	200	
CSI1000	Hedged	0.3261	0.5512	4.4248	2.3853	0.1355	118	0.0429
	Spot	0.2837	0.5312	1.8849	1.1036	0.2930	108	

4.1.4 ***Task 1: Stock Trading Executor.*** As we briefly introduced, we first rely on our model's price predictions to execute stock trades, by following our proprietary protocol: *enhanced indexing with daily rebalancing*. This is widely adopted in quantitative active management due to its capacity to manage large amounts of capital and relatively stable returns. Specifically, we rank all candidate stocks based on our model's predictions, select a certain number of stocks to buy under a specific budget, and adjust the portfolio daily based on the model's new predictions (either increasing or decreasing positions). We apply two types of trading strategies for *hedged return* and *spot return*. To achieve stable returns, we could neutralize the market risks by simultaneously *shorting index derivatives*. Thus, our hedged return depends on the relative performance between the selected stocks and the index shorts. Conversely, the spot return applies without hedging. We curve the whole portfolio backtesting yields from 2017Q1 to 2020Q3 in Figure 3 and report metrics in Table 2. From these detailed metrics, we notice that:

- For CSI300 and CSI500 indices, we notice that the spot return strategy yields better performance with higher annual returns and winning rates. Since these two indices comprise more "blue chip" stocks with lower volatility, the spot return strategy is generally a suitable choice for achieving higher profits.
- Compared to CSI300 and CSI500, the CSI1000 index includes a higher proportion of "small cap stocks", which are associated with larger price volatility. Therefore, the hedged return strategy demonstrates better performance in backtesting and may be more beneficial in practice for hedging market risks.

4.1.5 ***Task 2: Alpha Factor Producer.*** In quantitative finance, analysts normally use a combination of trading signals, i.e., *Alpha factors*, to manage risk more effectively and make data-driven portfolio investment decisions. To further assess our CSPO, we specifically validate its capability in producing Alpha factors. We integrate our model into our internal quantitative analysis framework, serving as a source of factor production. The output factors are then utilized as inputs in the downstream portfolio investment system. Based on the existing factors, we then compare the performances between solely using these existing factors and using our new factors in addition. The backtesting yield curves on three stock pools are shown in Figure 4 and detailed evaluation results are reported in Table 3. We have the following twofold observations:

Table 3: Evaluation of Alpha factor production. (1) "✓" and "×" respectively denote the cases where the additional factors produced from our model are integrated or not.

Data	Strategy	Alpha	AR ↑	WR ↑	SoR ↑	ShR ↑	MD ↓	MD-D ↓	TR ↓
CSI300	Hedged	×	-0.0891	0.4536	-1.8021	-1.0611	0.1365	289	0.0398
		✓	0.0715	0.5298	1.5488	0.9280	0.0731	87	0.0168
	Spot	×	0.0393	0.5397	0.4775	0.3175	0.1427	226	0.0398
		✓	0.1999	0.5497	1.3434	0.8929	0.1826	76	0.0168
CSI500	Hedged	×	-0.1114	0.4437	-1.9357	-1.1405	0.2085	194	0.0515
		✓	0.0818	0.5132	1.5821	0.9276	0.0696	15	0.0190
	Spot	×	0.0029	0.5001	0.1843	0.1240	0.2274	224	0.0515
		✓	0.1961	0.5497	1.2341	0.7940	0.1880	76	0.0190
CSI1000	Hedged	×	0.0528	0.4868	1.1124	0.6058	0.1348	80	0.0937
		✓	0.1947	0.5232	3.5369	1.9198	0.0591	53	0.0225
	Spot	×	0.1678	0.5262	1.1034	0.7555	0.1824	76	0.0937
		✓	0.3097	0.5397	1.8087	1.1243	0.1649	12	0.0225

- We observe from Figure 4 that, the warm-colored curves (for both hedged and spot return strategies), representing the cases with our model's Alpha factors, exhibit a more stable upward trend over time compared to our original baseline, i.e., in cold-colored lines. This suggests that our model consistently provides positive returns with less volatility, making it a reliable choice for achieving steady investment growth in these stock pools.
- As shown in Table 3, strategies for both hedged and spot returns that incorporate our additional alpha factors consistently outperform the original settings across all evaluation metrics. Furthermore, the hedged return strategy is usually more favorable in practice due to its anticipated stability. Additionally, our original factor settings on CSI300 and CSI500 may yield unsatisfied performances, e.g., negative annual returns; in contrast, our model effectively turns losses into profits, which demonstrates its capability to generate impactful and effective Alpha factors.

4.2 Public Benchmarking Evaluation (RQ2)

4.2.1 ***Evaluation Data.*** For a comprehensive benchmarking with existing models, we also include CSI100 index data for comparison, in addition to the two most widely studied indexes, CSI300 and CSI500. Data statistics are reported in Table 4.

Table 4: Data statistics for public benchmarking evaluation.

Asset	Training			Evaluation		
	#Ins.	Time	#Trans.	#Ins.	Time	#Trans.
C-futures	2,770	08Q1-16Q4	2,190	2,243	17Q1-20Q3	871
F-futures	424	08Q1-16Q4	2,190	344	17Q1-20Q3	871
CSI100	100	08Q1-16Q4	2,190	100	17Q1-20Q3	871
CSI300	300	08Q1-16Q4	2,190	300	17Q1-20Q3	871
CSI500	500	08Q1-16Q4	2,190	500	17Q1-20Q3	871

4.2.2 ***Competing Methods.*** We compare CSPO with fourteen representative models of stock price forecasting, i.e., deep-neural-network-based and decision-tree-based methods. Due to page limits, we attach their detailed introduction in Appendix A.3.

- **Deep-neural-network-based (DNN-based) methods.** These approach analyzes sequential time steps through diverse neural architectures to forecast prices, encompassing nine methods: LSTM [19], GRU [27], Transformer [52], ALSTM [46], SFM [64], TCN [2], TabNet [1], TFT [4], TRA [34] and Localformer [22].
- **Decision-tree-based methods.** These approach sequentially builds decision trees where successive models focus on correcting predecessor errors, maintaining strong predictive power as classic trading models. The included representatives are XGBoost [7], CatBoost [39], LightGBM [24], and DoubleEnsemble [62].

4.2.3 ***Evaluation Metrics.*** We follow [59] to include four commonly used metrics: (1) Information Coefficient (IC), (2) Rank Information Coefficient (RIC), (3) Information Ratio of IC (IR_{IC}), and (4) Information Ratio of Rank IC ($IR_{Rank\ IC}$). Higher values indicate better performance. Explanations are detailed in Appendix A.1.

4.2.4 ***Overall Performance Analysis.*** We report the five-time averaged results in Table 5 with the following discussions:

- DNN-based methods show relatively competitive performance but some of them rank lower than decision-tree-based methods, particularly in IC and IR_{IC} metrics. Decision-tree-based methods, e.g., DoubleEnsemble, perform well in RIC and $IR_{rank\ IC}$, with some instances highlighted, suggesting that they are robust in ranking-related tasks. Particularly, TRA and DoubleEnsemble are representative models with outstanding performance.
- CSPO consistently achieves the best performance, as shown by the highest metric values, i.e., IC, IR_{IC}, RIC, $IR_{rank\ IC}$. Furthermore, the performance gains are substantial, with improvements from 8.87% to 139.62% across different data and metrics. We also conduct the paired t-tests. Each associated p-value also confirms that the performance improvements of our model over others are statistically significant.
- Furthermore, CSPO performs significantly better on CSI100. This is because CSI100 represents the Top-100 most stable blue-chip stocks and CSPO can more easily capture patterns and dependencies among them and market futures. For larger stock pools CSI300 and CSI500, the greater stock diversity introduces increased variability and complexity. And our model is still competitive, indicating its adaptability across different stock pool sizes in practical deployment.

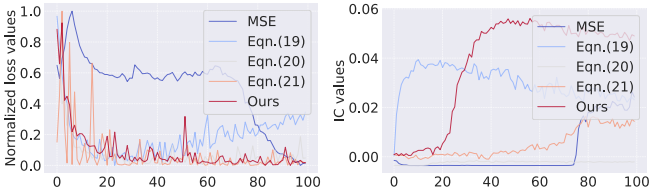


Figure 5: Empirical comparison of objective functions.

4.3 Empirical Analyses of CSPO (RQ3)

4.3.1 ***Ablation Study.*** To study the effect of each proposed module, we introduce several model variants by disabling their functionality. We report the results of experimenting on CSI300 data in Table 6 and provide the analyses accordingly.

(1) w/o futures information: This variant disables the encoding of commodity and financial futures information aggregation in CME-Layer, relying solely on the stock features. The performance drops significantly across all metrics, ranging from 8.78% to 24.76%, highlighting the importance of futures information synergy in stock price forecasting.
(2) w/o `BDP-Former`: This variant replaces our entire `BDP-Former` by first adding futures and stock features and then passing them through a two-layer MLP before predicting final prices. The performance gap between this variant and our CSPO is substantial, demonstrating the effectiveness of `BDP-Former` in capturing latent correlations between diverse futures and stocks for price movement forecasting.
(3) w/o pseudo-volatility: Lastly, we remove pseudo-volatility estimation, and replace the associated volatility-aware optimization with the MSE objective. Our experimental results confirm that the integration of such pseudo-volatility optimization is crucial for improving prediction accuracy. We provide a more detailed empirical analysis of this learning paradigm in § 4.3.3.

4.3.2 ***Pseudo-volatility Compatibility on Existing Models.*** We also explore the compatibility of our pseudo-volatility (PV) design with other DNN-based models, e.g., LocalFormer [22] and TRA [34]. Specifically, we retain their original feature encoding and decoding parts but incorporate our `PV-Estimator` and adjust their loss function accordingly. As shown in Table 7, both PV-integrated models demonstrate notable performance improvements, verifying the design effectiveness of our pseudo-volatility optimization.

4.3.3 ***Evaluation of Objective Function Designs.*** Lastly, in Figure 5, we compared our final objective function of Eqn. (22), with MSE and other functions of Eqn.'s (19)~(21) on 100 epochs training. For the left-hand side figure of loss values, we apply the min-max normalization for value scaling between [0,1]. For the right-hand side figure, we report corresponding IC values. We observe that, compared to other loss designs, while our final objective and Eqn. (21) present better convergence, our objective shows a more stable training process and superior IC value performance.

5 Related Work

Stock Price Movement Forecasting. Stock price movement forecasting (SPMF) has been a longstanding challenge in the financial domain due to the temporal and non-linear complexities inherent in

Table 5: Performance comparison across different models on CSI100, CSI300, and CSI500 data. Colors indicate the best and second-best performing models, i.e., best and second-best .

Model	CSI100				CSI300				CSI500			
	IC ↑	IR_{IC} ↑	RIC ↑	$IR_{Rank\ IC}$ ↑	IC ↑	IR_{IC} ↑	RIC ↑	$IR_{Rank\ IC}$ ↑	IC ↑	IR_{IC} ↑	RIC ↑	$IR_{Rank\ IC}$ ↑
LSTM	0.0280 ± 0.00	0.1489 ± 0.02	0.0401 ± 0.00	0.2207 ± 0.02	0.0323 ± 0.00	0.2296 ± 0.04	0.0411 ± 0.00	0.3401 ± 0.03	0.0389 ± 0.00	0.3904 ± 0.05	0.0493 ± 0.00	0.5310 ± 0.03
GRU	0.0299 ± 0.00	0.1667 ± 0.02	0.0401 ± 0.00	0.2284 ± 0.02	0.0329 ± 0.00	0.2403 ± 0.05	0.0423 ± 0.00	0.3399 ± 0.03	0.0414 ± 0.00	0.3919 ± 0.04	0.0565 ± 0.00	0.5812 ± 0.03
Transformer	0.0239 ± 0.01	0.1411 ± 0.01	0.0349 ± 0.00	0.2124 ± 0.03	0.0254 ± 0.00	0.2040 ± 0.02	0.0427 ± 0.00	0.3181 ± 0.02	0.0302 ± 0.00	0.2884 ± 0.03	0.0472 ± 0.00	0.4811 ± 0.03
ALSTM	0.0364 ± 0.00	0.2124 ± 0.03	0.0423 ± 0.01	0.2562 ± 0.02	0.0371 ± 0.01	0.2697 ± 0.05	0.0455 ± 0.01	0.3786 ± 0.06	0.0396 ± 0.01	0.3946 ± 0.05	0.0562 ± 0.00	0.5576 ± 0.04
SFM	0.0344 ± 0.01	0.1776 ± 0.03	0.0436 ± 0.01	0.2349 ± 0.02	0.0370 ± 0.00	0.2879 ± 0.04	0.0463 ± 0.00	0.3775 ± 0.04	0.0353 ± 0.00	0.3007 ± 0.04	0.0510 ± 0.00	0.4728 ± 0.03
TCN	0.0179 ± 0.00	0.1132 ± 0.02	0.0146 ± 0.00	0.0813 ± 0.02	0.0279 ± 0.00	0.2181 ± 0.01	0.0421 ± 0.04	0.3429 ± 0.01	0.0103 ± 0.02	0.0933 ± 0.08	0.0087 ± 0.01	0.0870 ± 0.07
TabNet	0.0279 ± 0.00	0.1596 ± 0.01	0.0360 ± 0.00	0.2142 ± 0.02	0.0199 ± 0.01	0.1477 ± 0.07	0.0351 ± 0.00	0.2693 ± 0.05	0.0321 ± 0.00	0.3562 ± 0.03	0.0406 ± 0.00	0.4425 ± 0.03
TFT	0.0278 ± 0.01	0.1333 ± 0.03	0.0114 ± 0.02	0.0551 ± 0.01	0.0338 ± 0.00	0.1999 ± 0.03	0.0129 ± 0.01	0.0913 ± 0.04	0.0398 ± 0.01	0.2900 ± 0.04	0.0117 ± 0.01	0.0894 ± 0.09
Localformer	0.0289 ± 0.00	0.1747 ± 0.02	0.0351 ± 0.00	0.2147 ± 0.01	0.0373 ± 0.00	0.2983 ± 0.03	0.0488 ± 0.00	0.3869 ± 0.03	0.0362 ± 0.00	0.3374 ± 0.03	0.0551 ± 0.00	0.5584 ± 0.03
TRA	0.0458 ± 0.01	0.2543 ± 0.01	0.0534 ± 0.01	0.3037 ± 0.03	0.0445 ± 0.01	0.3653 ± 0.05	0.0533 ± 0.00	0.4403 ± 0.03	0.0396 ± 0.00	0.4133 ± 0.03	0.0529 ± 0.01	0.5849 ± 0.05
XGBoost	0.0548 ± 0.01	0.2913 ± 0.03	0.0473 ± 0.00	0.2977± 0.02	0.0500 ± 0.00	0.3767 ± 0.00	0.0511 ± 0.00	0.4344 ± 0.00	0.0409 ± 0.00	0.3428 ± 0.01	0.0428 ± 0.00	0.4071 ± 0.01
CatBoost	0.0552 ± 0.00	0.2811 ± 0.01	0.0455 ± 0.00	0.2639 ± 0.01	0.0494 ± 0.00	0.3467 ± 0.00	0.0473 ± 0.00	0.3507 ± 0.01	0.0419 ± 0.00	0.3324 ± 0.01	0.0423 ± 0.00	0.3770 ± 0.02
LightGBM	0.0403 ± 0.01	0.2391 ± 0.04	0.0409 ± 0.00	0.2515 ± 0.04	0.0466 ± 0.00	0.3790 ± 0.01	0.0510 ± 0.01	0.4037 ± 0.01	0.0381 ± 0.00	0.3654 ± 0.03	0.0482 ± 0.00	0.4910 ± 0.02
DoubleEnsemble	0.0478 ± 0.00	0.2933 ± 0.00	0.0461 ± 0.00	0.2843 ± 0.00	0.0533 ± 0.00	0.4461 ± 0.01	0.0499 ± 0.00	0.4228 ± 0.01	0.0408 ± 0.00	0.3710 ± 0.01	0.0464 ± 0.00	0.4450 ± 0.01
CSPO	0.0671 ± 0.01	0.7028 ± 0.08	0.0704 ± 0.01	0.5621 ± 0.07	0.0832 ± 0.01	0.7176 ± 0.13	0.0786 ± 0.00	0.7262 ± 0.07	0.0560 ± 0.00	0.6108 ± 0.11	0.0625 ± 0.01	0.6368 ± 0.08
Gain (%)	**22.45%**	**≥100%**	**31.84%**	**85.08%**	**56.10%**	**60.86%**	**47.47%**	**64.93%**	**33.65%**	**47.79%**	**10.62%**	**8.87%**
p-value	$6.9{\cdot}10^{-3}$	$3.0{\cdot}10^{-5}$	$1.7{\cdot}10^{-4}$	$3.0{\cdot}10^{-6}$	$1.8e{\cdot}10^{-5}$	$4.3{\cdot}10^{-4}$	$4.6e{\cdot}10^{-6}$	$2.1e{\cdot}10^{-5}$	$4.7e{\cdot}10^{-5}$	$3.1e{\cdot}10^{-4}$	$6.3e{\cdot}10^{-3}$	$7.9e{\cdot}10^{-3}$

Table 6: Ablation study on CSI300 data.

Variant	IC	IR_{IC}	RIC	$IR_{Rank\ IC}$
w/o futures information	0.0626	0.6546	0.0627	0.6263
	-24.76%	-8.78%	-20.23%	-13.76%
w/o BDP-Former	0.0570	0.3179	0.0514	0.2722
	-31.49%	-55.70%	-34.61%	-62.52%
w/o pseudo-volatility	0.0645	0.6465	0.0647	0.6344
	-22.48%	-9.91%	-17.68%	-12.64%
CSPO	**0.0832**	**0.7176**	**0.0786**	**0.7262**

Table 7: Performance of pseudo-volatility-integrated models.

Method	IC	IR_{IC}	RIC	$IR_{Rank\ IC}$
LocalFormer	0.0356	0.2756	0.0468	0.3784
LocalFormer$_{PV}$	**0.0415** (+16.57%)	**0.3165** (+14.84%)	**0.0508** (+8.55%)	**0.4114** (+8.72%)
TRA	0.0440	0.3535	0.0540	0.4451
TRA$_{PV}$	**0.0520** (+18.18%)	**0.4243** (+20.03%)	**0.0553** (+2.41%)	**0.4821** (+8.31%)

financial data. Traditional approaches primarily rely on fundamental analysis, utilizing manually engineered features and macroeconomic indicators [3, 20, 23]. The advent of machine learning introduced models such as decision trees [48, 49] and gradient boosting trees (GBTs) [7, 24] improves predictive performance by capturing the dynamic and nonlinear nature of market behavior [44, 62]. Deep learning models have recently revolutionized SPMF by enabling the direct utilization of raw time-series data, reducing reliance on manually engineered features. Specifically, RNN-based methods [28, 53, 63] have shown success in modeling temporal dependencies; CNN-based methods [21, 41] treat historical price data as structured input, effectively extracting localized patterns. Graph-based approaches [30, 50] studies the intricate interdependence among different stocks, which is a a common methodology in various applications [9, 10]. Furthermore, recent works have incorporated additional data sources, such as exchanges [5], sales [65], and earnings calls [38, 43, 61], for information enhancement [5, 18]. Some other efforts try to adjust and stabilize model predictions with consideration of stock price volatility [16, 31, 72].

Transformer-based Models for SPMF. Transformer structures [26, 52] are widely used in many applications [8, 11, 12, 35, 47, 56]. It further has emerged as a leading approach in financial time-series forecasting [6, 22, 45, 55, 70]. Recent studies have introduced various enhancements to Transformer architectures. For instance, methods [14, 53] capture multi-scale financial dependencies through enhanced feature extraction, showing strong prediction performance across markets. Other works have explored the integration of external data sources to enhance prediction. TEANet [66] fuses social media text and prices for temporal modeling. StockFormer [17] adopts a hybrid approach by integrating predictive coding with reinforcement learning. Meanwhile, MASTER [29] addresses cross-time and momentary stock correlation through market-guided feature selection. Generally, these transformer-based models capture long-range dependencies and complex interactions and have demonstrated remarkable performance superiority in stock price forecasting.

6 Conclusions and Future Work

We propose CSPO, featuring the BDP-Former architecture that jointly models temporal patterns from futures/stock markets and price inter-correlations. The framework introduces pseudo-volatility guided loss weighting to enhance stability under real trading conditions. Extensive evaluations demonstrate superiority in both industrial backtesting and academic benchmarks. For future work, two directions emerge as critical: (1) Integrating rigorously filtered LLM insights [32, 33, 37, 68] to enhance market analysis while ensuring information credibility; (2) Developing continual learning protocols [60, 67] for efficient model adaptation to streaming financial data while maintaining prediction accuracy.

Acknowledgments

This work was partially supported by NSFC under Grant 62302421, Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280, Ant Group through CCF-Ant Research Fund, Shenzhen Research Institute of Big Data under grant SIF20240004, and the Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen.

References

[1] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. In *AAAI*, Vol. 35. 6679–6687.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* (2018).

[3] George EP Box and David A Pierce. 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *JASA* 65, 332 (1970), 1509–1526.

[4] Nicolas Loeff Tomas Pfister Bryan Lim, Sercan O. Arik. 2021. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *IJF* 37, 4 (2021), 1748–1764.

[5] Guangxi Cao, Longbing Xu, and Jie Cao. 2012. Multifractal detrended cross-correlations between the Chinese exchange market and stock market. *Physica A: Statistical Mechanics and its Applications* 391 (2012), 4855–4866.

[6] Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. 2024. Pathformer: Multi-scale Transformers with Adaptive Pathways for Time Series Forecasting. In *ICLR*.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *SIGKDD*. 785–794.

[8] Yankai Chen, Yixiang Fang, Qiongyan Wang, Xin Cao, and Irwin King. 2024. Deep Structural Knowledge Exploitation and Synergy for Estimating Node Importance Value on Heterogeneous Information Networks. In *AAAI*, Vol. 38. 8302–8310.

[9] Yankai Chen, Quoc-Tuan Truong, Xin Shen, Jin Li, and Irwin King. 2024. Shopping Trajectory Representation Learning with Pre-training for E-commerce Customer Understanding and Recommendation. *SIGKDD* (2024), 385–396.

[10] Yankai Chen, Quoc-Tuan Truong, Xin Shen, Ming Wang, Jin Li, Jim Chan, and Irwin King. 2023. Topological Representation Learning for E-commerce Shopping Behaviors. In *MLG-KDD*.

[11] Yankai Chen, Yaozu Wu, Shicheng Ma, and Irwin King. 2020. A literature review of recent graph embedding techniques for biomedical data. In *ICONIP*. Springer, 21–29.

[12] Yankai Chen, Yifei Zhang, Huifeng Guo, Ruiming Tang, and Irwin King. 2022. An Effective Post-training Embedding Binarization Approach for Fast Online Top-K Passage Matching. In *AACL*. 102–108.

[13] Chin Man Chui and Jian Yang. 2012. Extreme correlation of stock and bond futures markets: international evidence. *Financial Review* 47, 3 (2012), 565–587.

[14] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. 2020. Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *IJCAI*. 4640–4646.

[15] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. 2021. Adarnn: Adaptive learning and forecasting of time series. In *CIKM*. 402–411.

[16] Yitong Duan, Lei Wang, Qizhong Zhang, and Jian Li. 2022. Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns. In *AAAI*. 4468–4476.

[17] Siyu Gao, Yunbo Wang, and Xiaokang Yang. 2023. StockFormer: Learning Hybrid Trading Machines with Predictive Coding. In *IJCAI*. 4766–4774.

[18] Kim Hiang Liow, Joseph Ooi, and Yantao Gong. 2005. Cross-market dynamics in property stock markets: Some international evidence. *Journal of Property Investment & Finance* 23 (2005), 55–75.

[19] S Hochreiter. 1997. Long Short-term Memory. *Neural Computation MIT-Press* (1997).

[20] Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *IJF* 20, 1 (2004), 5–10.

[21] Ehsan Hoseinzade and Saman Haratizadeh. 2019. CNNpred: CNN-based stock market prediction using a diverse set of variables. *ESA* 129 (2019), 273–285.

[22] Juyong Jiang, Peiyan Zhang, Yingtao Luo, Chaozhuo Li, Jae Boum Kim, Kai Zhang, Senzhang Wang, Xing Xie, and Sunghun Kim. 2023. AdaMCT: adaptive mixture of CNN-transformer for sequential recommendation. In *CIKM*. 976–986.

[23] G Kavitha, A Udhayakumar, and D Nagarajan. 2013. Stock market trend analysis using hidden markov models. *arXiv* (2013).

[24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *NeurIPS* 30 (2017).

[25] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *NeurIPS* 30 (2017).

[26] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, Vol. 1. 2.

[27] Dzmitry Bahdanau Yoshua Bengio Kyunghyun Cho, Bart van Merriënboer. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *SSST*. 103–111.

[28] Hao Li, Yanyan Shen, and Yanmin Zhu. 2018. Stock price prediction using attention-based multi-input LSTM. In *ACML*. PMLR, 454–469.

[29] Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. 2024. MASTER: Market-Guided Stock Transformer for Stock Price Forecasting. In *AAAI*, Vol. 38. 162–170.

[30] Wei Li, Ruihan Bao, Keiko Harimoto, Deli Chen, Jingjing Xu, and Qi Su. 2021. Modeling the stock relation with graph network for overnight stock movement prediction. In *IJCAI*. 4541–4547.

[31] Xiangyu Li, Xinjie Shen, Yawen Zeng, Xiaofen Xing, and Jin Xu. 2024. FinReport: Explainable Stock Earnings Forecasting via News Factor Analyzing Model. In *WebConf*. 319–327.

[32] Yangning Li, Jiaoyan Chen, Yinghui Li, Yuejia Xiang, Xi Chen, and Hai-Tao Zheng. 2023. Vision, deduction and alignment: An empirical study on multi-modal knowledge graph alignment. In *ICASSP*. IEEE, 1–5.

[33] Yinghui Li, Qingyu Zhou, Yuanzhen Luo, Shirong Ma, Yangning Li, Hai-Tao Zheng, Xuming Hu, and S Yu Philip. 2024. When LLMs meet cunning texts: A fallacy understanding benchmark for large language models. In *NeurIPS*.

[34] Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. 2021. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *SIGKDD*. 1017–1026.

[35] Sida Lin, Zhouyi Zhang, Yankai Chen, Chenhao Ma, Yixiang Fang, Shan Dai, and Guangli Lu. 2024. Effective Job-market Mobility Prediction with Attentive Heterogeneous Knowledge Learning and Synergy. In *CIKM*. 3897–3901.

[36] Sharon Xiaowen Lin and Michael N Tamvakis. 2001. Spillover effects in energy futures markets. *Energy Economics* 23, 1 (2001), 43–56.

[37] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024. A Survey of Text Watermarking in the Era of Large Language Models. *ACM Comput. Surv.* 57, 2 (2024).

[38] Mengpu Liu, Mengying Zhu, Xiuyuan Wang, Guofang Ma, Jianwei Yin, and Xiaolin Zheng. 2024. ECHO-GL: Earnings Calls-Driven Heterogeneous Graph Learning for Stock Movement Prediction. In *AAAI*, Vol. 38. 13972–13980.

[39] Aleksandr Vorobev Anna Veronika Dorogush Andrey Gulin Liudmila Prokhorenkova, Gleb Gusev. 2018. CatBoost: Unbiased Boosting with Categorical Features. *NeurIPS* 31 (2018).

[40] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. ICLR.

[41] Wenjie Lu, Jiazheng Li, Jingyang Wang, and Lele Qin. 2021. A CNN-BiLSTM-AM method for stock price prediction. *NCA* 33, 10 (2021), 4741–4753.

[42] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. *NeurIPS* 32 (2019).

[43] Sourav Medya, Mohammad Rasoolinejad, Yang Yang, and Brian Uzzi. 2022. An exploratory study of stock price movements from earnings calls. In *WebConf*. 20–31.

[44] Rudra Kalyan Nayak, Debahuti Mishra, and Amiya Kumar Rath. 2015. A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices. *ASC* 35 (2015), 670–680.

[45] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*.

[46] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *IJCAI* (Melbourne, Australia). 2627–2633.

[47] Zexuan Qiu, Jieming Zhu, Yankai Chen, Guohao Cai, Weiwen Liu, Zhenhua Dong, and Irwin King. 2024. EASE: Learning Lightweight Semantic Feature Adapters from Large Language Models for CTR Prediction. In *CIKM*. 4819–4827.

[48] J. Ross Quinlan. 1986. Induction of decision trees. *ML* (1986), 81–106.

[49] J Ross Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.

[50] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *AAAI*, Vol. 35. 497–504.

[51] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *ICACCI*. IEEE, 1643–1647.

[52] A Vaswani. 2017. Attention is all you need. *NeurIPS* (2017).

[53] Heyuan Wang, Tengjiao Wang, Shun Li, Jiayi Zheng, Shijie Guan, and Wei Chen. 2022. Adaptive Long-Short Pattern Transformer for Stock Investment Selection.. In *IJCAI*. 3970–3977.

[54] Andrew G Wilson and Pavel Izmailov. 2020. Bayesian deep learning and a probabilistic perspective of generalization. *NeurIPS* 33 (2020), 4697–4708.

[55] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *NeurIPS* 34 (2021), 22419–22430.

[56] Yaozu Wu, Yankai Chen, Zhishuai Yin, Weiping Ding, and Irwin King. 2023. A survey on graph embedding techniques for biomedical data: Methods and applications. *Information Fusion* 100 (2023), 101909.

[57] Hongjie Xia, Huijie Ao, Long Li, Yu Liu, Sen Liu, Guangnan Ye, and Hongfeng Chai. 2024. CI-STHPAN: Pre-trained Attention Network for Stock Selection with Channel-Independent Spatio-Temporal Hypergraph. *AAAI* 38, 8 (March 2024).

[58] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. Rest: Relational event-driven stock trend forecasting. In *WebConf*. 1–10.

[59] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. 2020. Qlib: An AI-oriented Quantitative Investment Platform. *arXiv* (2020).
[60] Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S Yu, and Irwin King. 2024. Recent Advances of Multimodal Continual Learning: A Comprehensive Survey. *arXiv preprint arXiv:2410.05352* (2024).
[61] Zixuan Yuan, Yada Zhu, Wei Zhang, and Hui Xiong. 2023. Earnings Call Analysis Using a Sparse Attention Based Encoder and Multi-Source Counterfactual Augmentation. In *IJCAI*. 331–339.
[62] Chuheng Zhang, Yuanqi Li, Xi Chen, Yifei Jin, Pingzhong Tang, and Jian Li. 2020. DoubleEnsemble: A new ensemble method based on sample reweighting and feature selection for financial data analysis. In *ICDM*. IEEE, 781–790.
[63] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *SIGKDD*. 2141–2149.
[64] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In *SIGKDD*. 2141–2149.
[65] Lei Zhang, Wang Xiang, Chuang Zhao, Hongke Zhao, Rui Li, and Runze Wu. 2022. Co-promotion Predictions of Financing Market and Sales Market: A Cooperative-Competitive Attention Approach. In *AAAI*, Vol. 36. 9040–9047.
[66] Qiuyue Zhang, Chao Qin, Yunfeng Zhang, Fangxun Bao, Caiming Zhang, and Peide Liu. 2022. Transformer-based attention network for stock movement prediction. *ESA* 202 (2022), 117239.
[67] Xinni Zhang, Yankai Chen, Chenhao Ma, Yixiang Fang, and Irwin King. 2024. Influential Exemplar Replay for Incremental Learning in Recommender Systems. In *AAAI*, Vol. 38. 9368–9376.
[68] Huaqin Zhao, Zhengliang Liu, Zihao Wu, Yiwei Li, Tianze Yang, Peng Shu, Shaochen Xu, Haixing Dai, Lin Zhao, Gengchen Mai, et al. 2024. Revolutionizing finance with llms: An overview of applications and insights. *arXiv preprint arXiv:2401.11641* (2024).
[69] Lifan Zhao, Shuming Kong, and Yanyan Shen. 2023. Doubleadapt: A meta-learning approach to incremental learning for stock trend forecasting. In *SIGKDD*. 3492–3503.
[70] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, Vol. 35. 11106–11115.
[71] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman & Hall/CRC.
[72] Mengying Zhu, Yan Wang, Fei Wu, Mengyuan Yang, Cheng Chen, Qianqiao Liang, and Xiaolin Zheng. 2022. WISE: Wavelet based Interpretable Stock Embedding for Risk-Averse Portfolio Management. In *WebConf*. 1–11.

A Experimental Details

A.1 Evaluation Metrics

- **Annualized Return (AR)**: AR is defined as the annualized rate of return on an investment.
- **Winning Rate (WR)**: WR is defined as the percentage of profitable trades relative to the total trades executed.
- **Sharpe Ratio (ShR)**: ShR measures the risk-adjusted return of an investment by comparing the return to its standard deviation. A higher ratio signifies better risk-adjusted performance.
- **Sortino Ratio (SoR)**: Similar to ShR, SoR evaluates risk-adjusted returns uses only downside deviation. This metric focuses on minimizing losses.
- **Maximum Drawdown (MD)**: MD reflects the largest peak-to-trough decline over a specified period. It measures the extent of the worst loss sustained before recovery.
- **Maximum Drawdown Duration (MD-D)**: MD-D indicates the time taken for an investment to recover from its maximum drawdown to its previous peak.
- **Turnover Rate (TR)**: TR proportion of assets replaced with which assets within a portfolio are traded over a specified period.
- **Information Coefficient (IC)**: IC is a key metric in financial analysis that measures the correlation between predictions and realized returns.
- **Rank IC (RIC)**: RIC is similar to IC but is computed using rank-based correlation. It is useful for long-short strategy which rely on the model's ranking capacity.
- **Information Ratio of IC ($\mathbf{IR_{IC}}$)**: $\mathrm{IR_{IC}}$ is the ratio of the IC to its standard deviation, quantifying the stability of the model's performance over time.
- **Information Ratio of Rank IC ($\mathbf{IR_{Rank\ IC}}$)**: The ratio of Rank IC to its standard deviation, reflecting the stability of the rank-based performance.

A.2 Evaluation Configurations

We implement using Python 3.8 and PyTorch 1.13.1 with non-distributed training. The experiments are run on a Linux machine with 8 NVIDIA A100 GPUs and 6 Intel(R) Xeon(R) Platinum 8350C CPUs with 2.60GHz. We adhere to all baselines' officially reported hyper-parameter settings and conduct a grid search for models without prescribed configurations. For a fair comparison, we fix the embedding dimension at 512. The learning rate is tuned in the range $\{10^{-5}, 10^{-4}, 10^{-3}\}$. Optimization for all models is performed using the default AdamW optimizer [40].

A.3 Details of Competing Methods

- **Deep-neural-network-based (DNN-based) methods.**
  (1) LSTM [19] captures long-term dependencies through memory cells, effectively modeling sequential data.
  (2) GRU [27] uses gating mechanisms to capture dependencies in sequential data with fewer parameters than LSTM.
  (3) Transformer [52] utilizes self-attention to capture long-range dependencies, suitable for time series forecasting.
  (4) ALSTM [46] integrates attention mechanisms into LSTM, selectively focusing on relevant time steps.
  (5) SFM [64] captures temporal features by encoding state and frequency representations.
  (6) TCN [2] employs causal convolutions and flexible receptive fields for modeling sequential data.
  (7) TabNet [1] dynamically selects features with sequential attention for effective tabular data modeling.
  (8) Localformer [22] enhances transformers by focusing on local temporal dynamics.
  (9) TRA [34] applies Transformer-based relational attention to capture long-term dependencies in graph-structured data.
- **Gradient-boosting-based methods.**
  (10) XGBoost [7] proposes a sparsity-aware algorithm and weighted quantile sketch for approximate tree learning.
  (11) CatBoost [39] efficiently handles categorical features using ordered boosting to improve the performance.
  (12) LightGBM [24] employs histogram-based methods and leaf-wise tree growth for better scalability and prediction accuracy.
  (13) DoubleEnsemble [62] combines two ensemble strategies to improve generalization, particularly for imbalanced data.