Digger-Guider: High-Frequency Factor Extraction for Stock Trend Prediction

Yang Liu, Chang Xu, Min Hou, Weiqing Liu, Jiang Bian, Qi Liu, *Member, IEEE* and Tie-Yan Liu, *Fellow, IEEE*

Abstract—Recent years have witnessed increasing attention being paid to AI-based quantitative investment. Compared to traditional low-frequency data (e.g., daily, weekly), high-frequency data (e.g., minute-level) is often underutilized for low-frequency stock trend prediction, leaving the vast potential for improvement. However, valuable and noisy information coexist in high-frequency data. The learning process of high-frequency factor extractors can easily be overwhelmed by noise, leading to overfitting. Moreover, common techniques used to prevent overfitting often result in poor performance on this task since they usually roughly restrict the model's capacity, making it challenging to model complex trading signals in high-frequency data. When designing high-frequency factor extractors, we face a tough dilemma. A high-capacity model may easily overfit to noise, while a simple but robust model may not capture complex high-frequency patterns. To address these problems, we propose maintaining model capacity while preventing overfitting by constructing two components that balance information and noise through interactions between them. Specifically, we propose a novel learning framework called *Digger-Guider* to extract informative stock representations from noisy high-frequency data. We develop a high-capacity model called *Digger* to extract local and detailed features from the high-frequency data, and we design a robust model called *Guider* to capture global tendency features and help the Digger overcome the noise. The Digger and Guider enhance each other through mutual distillation during training, serving as data-driven regularizations that work well on this task. Extensive experiments on real-world datasets demonstrate that our framework can produce powerful high-frequency stock factors that significantly improve stock trend prediction performance and our understanding of the finance market.

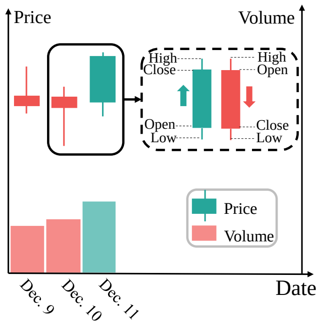
Index Terms—Stock Factor Extraction, Knowledge Distillation, Financial Investment, Deep Learning, Artificial Intelligence

1 Introduction

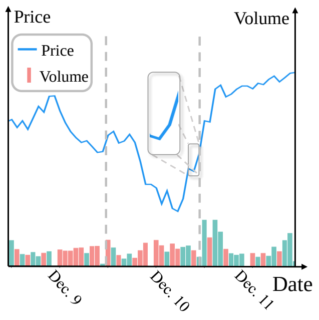
As one of the most well-known and complicated financial assets, stocks have always been considered key investment vehicle for growing wealth. Numerous speculative investors attempt to forecast future stock movements and make profits through spread (price differences). An accurate prediction of stock trends and a comprehensive understanding of the market are key to high and stable yields.

Relying on statistical theory and computer technology, investors attempt to extract effective information relevant to the future price. According to *Behavioral Finance* [1], [2], one of the main elements that influence stock prices is the biases of investor behavior, which is usually reflected in the widely used historical price-volume sequences. In the past, due to the limitation of the computational resources that previous work can hardly afford to train high capacity models, most of them [3], [4], [5], [6] mainly leverage low-frequency price-volume information. Nowadays, with the increased digitization of finance, the ability to collect high-frequency data together with affordable computational resources makes it possible to leverage high-frequency data. In practice, we observe that high-frequency data contain instructive investment signals overlooked by low-frequency data. For instance, after analyzing massive high-frequency data, we find that the stock price is likely to go up the next day if there is an obvious pulling up before the close. Figure 1 shows a specific example. This phenomenon may result from those investors who know some bull news (news that may lead to a price increase) in advance. To reduce the influence of price fluctuation, they choose to buy shares right before closing with the help of automated trading systems, leading to a dramatic pulling up before the close. Such subtle patterns cannot be discovered in low-frequency data. Therefore, high-frequency data brings opportunities to improve stock trend prediction further.

This paper focuses on extracting high-frequency (e.g., minutely) stock factors for relatively low-frequency (e.g., daily) stock trend prediction, which has rarely been studied



(a) Daily-Frequency stand time bars



(b) Minute-Frequency stock chart

Fig. 1. The same 3-day stock price-volume series represented in different frequencies. Dramatic pulling up of price appears before the close on Dec. 10.

- Yang Liu, Chang Xu, Weiqing Liu, Jiang Bian and Tie-Yan Liu are with Microsoft Research Asia, Beijing, 100080, China. E-mail: {yangliu2, chanx, weiqing.liu,jiang.bian,Tie-Yan.Liu}@microsoft.com
- Min Hou and Qi Liu are with the Anhui Province Key Laboratory of Big Data Analysis and Application (BDAA), School of Data Science, University of Science and Technology of China, Hefei, Anhui 230027, China. E-mail: minho@mail.ustc.edu.cn, qiliuql@ustc.edu.cn.

Manuscript received Aug 21, 2021.(Corresponding author: Chang Xu.)

before. Most previous work is based on daily-frequency data for daily trend prediction. Daily-frequency data are widely used in many traditional stock analysis methods, such as Fourier Analysis [7], and Technical Analysis [8], [9], [10]. These works create technical indicators from charts and fixed mathematical formulas and then fit the stochastic stock trend. However, such hand-crafted indicators are usually criticized for their poor generalization of dynamic markets. Other recent works [3], [4], [5], [6] only focus on modeling correlations between trading days, but overlook subtle patterns hidden in intraday data and cannot be applied to high-frequency data directly. Although there exist a few studies that take high-frequency data as input [11], [12], [13], they require input data and output prediction to be of the same frequency in their models, which cannot be used for low-frequency prediction directly.

With shorter sampling intervals, the influence of a small number of trading individuals' random behavior would be amplified in high-frequency data [14]. Therefore, high-frequency data typically produces a lot of price fluctuations that distort the overall trend, which is meaningless noisy information for future trend prediction. In fact, directly using high-frequency data for stock trend prediction leads to worse performance than using the daily frequency model (see Section 5.2.1). Specifically, since high-frequency data carry more noisy signals than low-frequency data, directly training the factor extraction model would over-fit to noise, resulting in the model's poor generalization [3]. However, since common tricks for preventing over-fitting usually work on restricting the model capacity, complex trading signals in high-frequency data cannot be well modeled, leading to poor performance (see Section 5.2.3). To design an effective stock factor extractor over high-frequency data, the main challenge is to model complex high-frequency signals while avoiding being overwhelmed by noise. That is, when designing high-frequency factor extractors, we face a tough dilemma – a complex and high-capacity model would easily over-fit to noise, while a simple but robust model could not capture diverse high-frequency patterns.

To address the above challenges, we propose a learning framework called *Digger-Guider* to extract decent factors from informative but noisy high-frequency data. We construct two major components with different functions and different granularities of information utilization, conducting interactions between them. The first part, Digger, is designed as a high-capacity model to dig valid market signals. It processes high-frequency data in a fine-grained way to extract local and detailed price trend patterns. However, since valid patterns and noise coexist in high-frequency data, the training process of Digger would easily over-fit the noise. This is why we design the second part, Guider, as a simple but robust model working in a coarse-grained way. In contrast to Digger, Guider focuses more on leveraging coarse-grained information to extract global and robust features, which contain less noise and less detailed information. For interaction, mutual distillation is proposed to conduct knowledge sharing and data-driven regularization between the two components. Specifically, Guider distillates knowledge and transfers it to Digger as a regularization term, and guides the training process of Digger to overcome over-fitting; then Guider learns from Digger to make itself knowledgeable. The above process is carried out iteratively. As a result, the model can extract complex high-frequency factors while avoiding being overwhelmed by noise.

We analyze our method in a mathematical view and dissect its effects into three aspects: i) Knowledge sharing within the iterative learning between Digger and Guider can enhance each other. ii) The distillation term in the loss function acts as a data-driven regularization to prevent over-fitting. iii) The updated prior knowledge at each iteration allows the two components of Digger-Guider to be improved progressively. The algorithm analysis reveals how our model keeps capacity while preventing over-fitting, and extensive experiments on stock markets demonstrate the effectiveness of our framework. We verify that our framework can mine valid underlying patterns from high-frequency data and survive from submerging into the noise. Our method significantly improves performance, which further leads to a profitable trading strategy.

Our main contributions are as follows: 1) We propose a novel Digger-Guider framework to extract informative stock factors from noisy high-frequency data. Our work is one of the first few studies of feature extraction from high-frequency price-volume data. 2) We mathematically analyze the learning mechanism behind the mutual distillation process of Digger-Guider, revealing the intuition to prevent over-fitting while keeping model expressiveness. 3) We conduct extensive experiments on real-world datasets. Our approach significantly outperforms baselines on both price trend prediction and trading simulation.

2 Related work

2.1 Stock Factor Extraction

Previous works on stock factor extraction can be roughly divided into two classes: traditional hand-crafted technical indicator extraction and deep learning-based factor extraction. Traditional technical indicator approaches [8], [9], [10] consist of hand-crafted features based on historical price-volume data, etc. They heavily rely on domain knowledge and cannot be generalized to capture dynamic market trends.

Recent research proposes exploiting deep neural networks to learn stock factors and predict stock trends [15], [16], [17], [18], [19], [20]. Despite increasing efforts in this area, previous work mainly leverages low-frequency (say daily) information while few of them pay enough attention to high-frequency stock data. Most previous works [3], [5], [6], [7], [21], [22] are based on daily-frequency bars. They focus on modeling correlations between trading days but overlook subtle patterns hidden in intraday data. These methods cannot be adapted to high-frequency data directly since there are no specific designs for digging into detailed local information. Although there exist a few works taking high-frequency data as input [11], [12], [13], they do not match our application scenario. Specifically, they require input data and output prediction to be of the same frequency in their models, being unable to predict daily-level stock trends using high-frequency data. Besides, in recent years, there have been some representative time series forecasting works such as Informer [23] and ETSformer [24]. However, they were not designed for stock factor mining tasks and also require the input data and output prediction to be of the

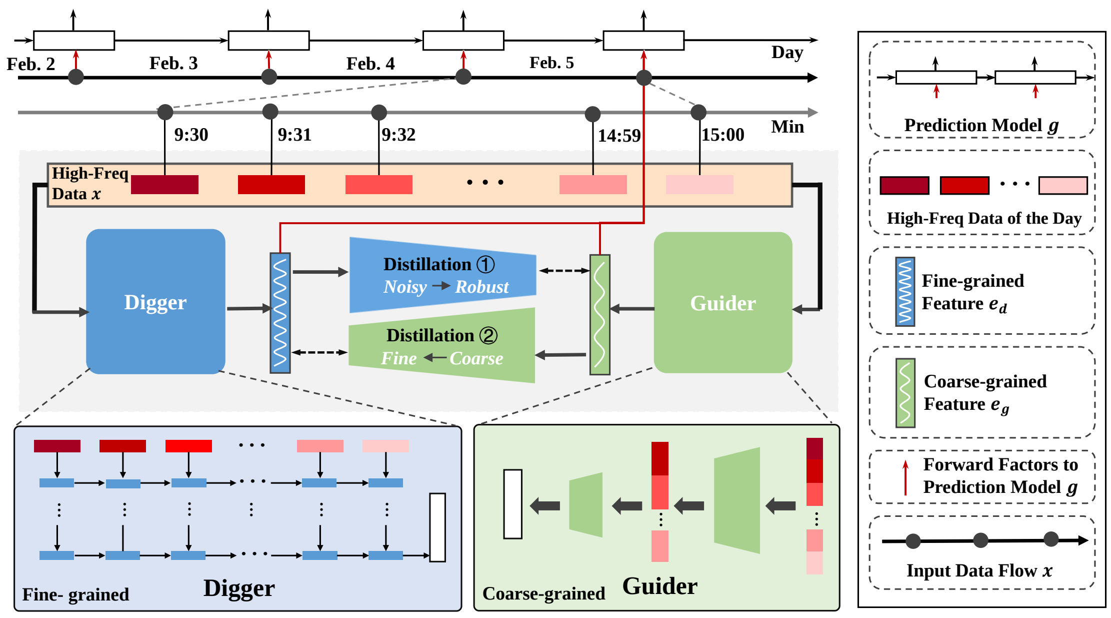


Fig. 2. Overview of Digger-Guider framework, which consists of two components: a high-capacity model, Digger, and a robust model, Guider. Digger works in a fine-grained way on information-digging of high-frequency data by extracting local and detailed features. Guider works in a coarse-grained way to capture global tendency features and help Digger overcome the noise. Digger and Guider enhance each other by mutual distillation during training.

same frequency. These works also pay insufficient attention to the noise problem in high-frequency stock data.

2.2 Knowledge Distillation

Knowledge distillation (KD) is a common method of knowledge transfer. It starts with training a powerful "teacher" model (or ensemble of models) followed by encouraging the "student" model to mimic the teacher's behavior. Initially, the common application of knowledge distillation is model compression [25]. Then KD is re-popularised by Hinton et al. [26]. They propose feeding the output probabilities of a larger network directly to a smaller one through KL Divergence loss. Beyond model compression, KD has also been applied to enhancing model performance [9], [27]. Inspired by these methods, we propose Digger-Guider framework that conducts mutual distillation to improve both sides. Compared with the one-way transfer in most KD approaches that transfers knowledge from the powerful model to the weak model to improve the weak one, Digger-Guider transfers knowledge bilaterally by mutual distillation to enhance both models.

3 High-Frequency Factor Extractor

In this paper, we take daily frequency as low frequency and minute (15-minute) frequency as high frequency, to illustrate our method. Note that our framework can be applied to any frequency data, even higher-frequency data. Before introducing the learning approach, we first provide a detailed problem description. A series of low-frequency data of one stock over T days is traditionally defined as $\boldsymbol{x} := \{x^1, x^2, \ldots, x^T\}$, where x^t is usually specified by few pre-defined daily indicators, including the highest price, opening price, lowest price, closing price, volume-weighted average price and trading volume. Regarding high-frequency data, each day can be split into N time slots, i.e., $x^t := \{x_1^t, x_2^t, \ldots, x_N^t\}$, where x_n^t consists of high-frequency indicators analogous to the daily ones.

Our goal is to learn a *factor extractor* $f(\cdot;\theta)$ that extracts predictive stock trend signals, i.e., the *price-volume embedding (stock factor)* e^t of day t for a single stock, denoted as $e^t = f(x^t;\theta), \quad \forall t \in [T]$, where θ represents the parameters. We define stock factor of T days as $\boldsymbol{e} := \{e^1, e^2, \ldots, e^T\}$ and we have $\boldsymbol{e} = f(\boldsymbol{x};\theta)$.

The factor extractor $f(\cdot;\theta)$ is evaluated by the performance of the extracted $\boldsymbol{e} = f(\boldsymbol{x};\theta)$ on stock trend prediction task. We train a prediction model $g(\cdot;\phi)$ that maps the stock factor to future price trend by

$$\mathcal{L}(\theta,\phi) = \sum_{(\boldsymbol{x},y)\in D_{train}} \mathcal{L}(g(f(\boldsymbol{x};\theta);\phi), y), \tag{1}$$

where y is the future price trend label, $\mathcal{L}$ is the loss function. These notations will be used consistently throughout this paper. To be specific, we take the rate of change of prices as the price trend, i.e., $y = p_{T+2}/p_{T+1} - 1$, where p_t represents the volume-weighted average price of the stock at day t. We take the daily price trend as our label because it is consistent with the transaction frequency of most real-world trading. Here we take the mean squared error (MSE) as the loss function. D_{train} is the training set and ϕ represents the parameters of the prediction model.

We adopt the straightforward method to jointly train the factor extractor $f(\cdot;\theta)$ and the prediction model $g(\cdot;\phi)$ by minimizing the total loss $\mathcal{L}$, to capture the underlying market signals from high-frequency data. The detailed framework and training strategy is elaborated as follows.

3.1 Digger-Guider Framework

We propose a novel learning framework over high-frequency data, named *Digger-Guider*, to extract effective factors from informative but noisy high-frequency data. We maintain the model capacity while preventing over-fitting by constructing two components and balancing the valuable and noisy information through interactions between them. Specifically, we construct two major components with different functions and various granularities of information utilization, and conduct interactions between them.

The *Digger* $f_d(\cdot;\theta_d)$ is used to dig informative signals from high-frequency data, which is a model of high capacity and great expressiveness to capture detailed signals. It processes high-frequency data in a fine-grained way to extract local and detailed price trend patterns. We use a two-layer GRU (Gate Recurrent Unit) [28] as Digger in our work. Since valid patterns and noise coexist in high-frequency data, the training process of Digger would easily over-fit to the noise. To overcome the over-fitting issue, the *Guider* is introduced as a simple but robust model with relatively low capacity. We use a simple CNN (Convolutional Neural Networks) [29] as Guider in our implementation. In contrast to Digger, Guider processes high-frequency data in a coarse-grained way. It places more emphasis on extracting global and robust features, which contain less noise and less detailed information. We present the architecture details of Digger and Guider we leverage in Section 5.1.3. Note that since our method is a general framework, other implementation(s) of model architectures can also be developed. The overall framework is shown in Figure 2.

3.2 Mutual Distillation

Mutual distillation is proposed to conduct iterative optimizations between the two components. Guider distills knowledge and transfers it to Digger as a regularization term, and guides the training process of Digger to overcome over-fitting; Then we let Guider learn from Digger to make itself knowledgeable. The above process is carried out iteratively. As a result, the model can extract complex high-frequency factors while avoiding being overwhelmed by noise. The effect of mutual distillation is illustrated in Figure 3.

3.2.1 *Guider → Digger*

We propose distilling knowledge from Guider to Digger to overcome the influence of noise. Specifically, we take the factor $\boldsymbol{e}_g$ of robust knowledge from Guider as a fixed target to enforce the embedding $\boldsymbol{e}_d$ from Digger to be close to it. The loss function of Digger $f_d(\cdot;\theta_d)$ is

$$\begin{aligned}\mathcal{L}_d(\theta_d,\phi) = \sum_{(\boldsymbol{x},y)\in D_{train}} &(1-\lambda_1)\mathcal{L}(g(f_d(\boldsymbol{x};\theta_d);\phi),y) \\ &+\lambda_1 \operatorname{dist}(f_d(\boldsymbol{x};\theta_d),\boldsymbol{e}_g),\end{aligned} \tag{2}$$

where $\boldsymbol{e}_d := f_d(\boldsymbol{x};\theta_d)$, and $\operatorname{dist}(\boldsymbol{e}_d,\boldsymbol{e}_g)$ is the knowledge distillation loss. Note that $\operatorname{dist}(\cdot,\cdot)$ can be in any form of distance metrics. As MSE is one of the most popular metrics, we use it in our work. λ_1 is the coefficient to balance original loss and the regularization from Guider.

In this way, Digger learns to refer robust and coarse-grained knowledge from Guider. Moreover, the distillation can regularize the function class of Digger to prevent over-fitting. Note that our approach is slightly different from the common practice of the teacher-student model. Commonly, the teacher is a large model (such as very deep neural networks) with higher knowledge capacity than the student model. The target of knowledge distillation is usually for model compression, which transfers knowledge from the teacher model to the student model to deploy it with limited computational resources. While in our approach, Digger and Guider are in equal status to teach each other. Mutual distillation in Digger-Guider works as a data-driven regularization to prevent over-fitting and to lead to better generalization ability.

3.2.2 *Digger → Guider*

For Guider, we first try to find the useful indicators of a day and then embed them into the feature space, that is $f_g = h_{emb}\circ h_{ind}$ where $\circ$ represents the operator of the composition function. h_{ind} is the indicator generator and h_{emb} is the embedding function. The indicator generator can be rule-based that leverages existing indicators (i.e., highest price, opening price, lowest price, closing price, volume-weighted average price, trading volume) And the embedding function takes the indicators as input and yields meaningful price-volume embeddings for stock trend prediction. The corresponding parameters are $\theta_g=\theta_{emb}\cup\theta_{ind}$.

When the indicator generator is rule-based and outputs traditional indicators (aggregated statistics, such as the highest price of the day), we call the corresponding Guider Rule-based Guider (RG). For the Rule-based Guider, h_{ind} is rule-based, thus the loss can be written as

$$\begin{aligned}&\mathcal{L}_{RG}(\theta_{emb},\phi) \\ &=\sum_{(\boldsymbol{x},y)\in D_{train}}\mathcal{L}(g(h_{emb}(h_{ind}(\boldsymbol{x});\theta_{emb});\phi),y),\end{aligned} \tag{3}$$

which is exactly consistent with the training loss of the daily frequency model.

More generally, the indicator generator can be a learnable model that generates useful new indicators. We call the corresponding Guider the Parametric Guider (PG). For the PG, h_{ind} is parameterized by θ_{ind}, thus the loss can be written as

$$\begin{aligned}&\mathcal{L}_{PG}(\theta_{emb},\theta_{ind},\phi) \\ &=\sum_{(\boldsymbol{x},y)\in D_{train}}\mathcal{L}(g(h_{emb}(h_{ind}(\boldsymbol{x};\theta_{ind});\theta_{emb});\phi),y).\end{aligned} \tag{4}$$

In the experiment, we use a mixed one that outputs the concatenation of the existing and new indicators as the indicator generator, which is a learnable special case of PG.

We can also improve Guider by knowledge transferred from Digger. The final loss of Guider is written as:

$$\begin{aligned}\mathcal{L}_g(\theta_g,\phi) = &(1-\lambda_2)\mathcal{L}_{PG}(\theta_{emb},\theta_{ind},\phi) \\ &+\lambda_2\sum_{(\boldsymbol{x},y)\in D_{train}}\operatorname{dist}(f_g(\boldsymbol{x};\theta_{ind},\theta_{emb}),\boldsymbol{e}_d),\end{aligned} \tag{5}$$

where $\boldsymbol{e}_g := f_g(\boldsymbol{x};\theta_{ind},\theta_{emb})$. And $\operatorname{dist}(\boldsymbol{e}_g,\boldsymbol{e}_d)$ here brings the embedding from Guider $\boldsymbol{e}_g$ close to the fixed $\boldsymbol{e}_d$. The detailed features learned by Digger can be transferred to Guider, which can enrich the knowledge of Guider and enhance its performance.

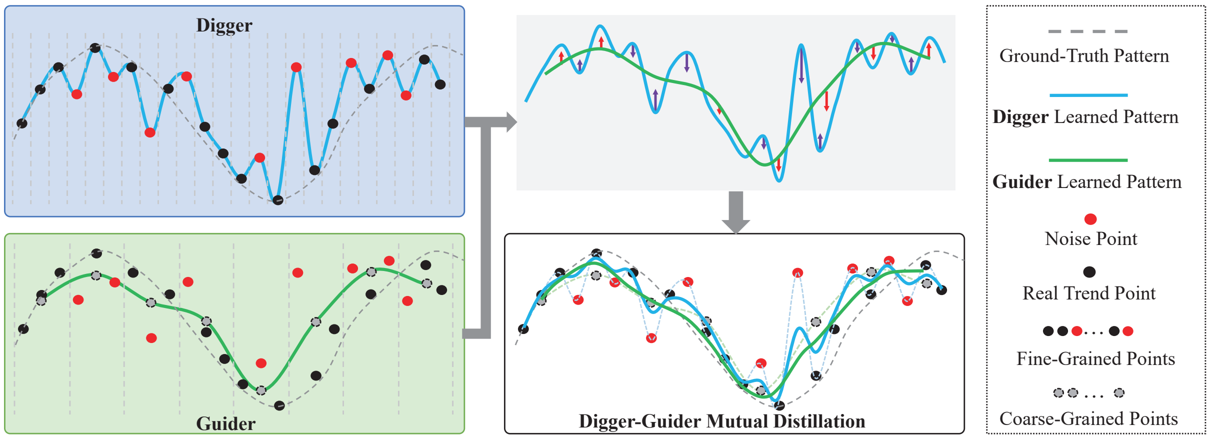


Fig. 3. Illustration of the effect of mutual distillation of Digger-Guider framework.

Algorithm 1: Digger-Guider Algorithm

```
1 Train Rule-based Guider by minimizing
    $\mathcal{L}_{RG}(\theta_{emb}, \phi)$ and get factor $\boldsymbol{e}_g$;
2 Train Digger by minimizing $\mathcal{L}_d(\theta_d, \phi)$ and get factor
    $\boldsymbol{e}_d$;
3 repeat
4     Train Parametric Guider by minimizing $\mathcal{L}_g(\theta_g, \phi)$
       and get new Guider factor $\boldsymbol{e}_g$;
5     Train Digger by minimizing $\mathcal{L}_d(\theta_d, \phi)$ and get
       new Digger factor $\boldsymbol{e}_d$;
6 until evaluation (current $\boldsymbol{e}_d$) ¡ evaluation (previous $\boldsymbol{e}_d$);
  Output: The best digger factor $\boldsymbol{e}_d$.
```

3.3 Algorithm

The overall learning process is described in Algorithm 1. We conduct iterative learning for training. We first train Guider using the rule-based indicator generator and then use it to guide Digger. Then, the price-volume embeddings learned by Digger can be transferred to Guider to make its embeddings rich in information. After that, we let Digger and Guider be optimized alternatively by using Eqn. (2) and (5). In this way, Digger and Guider share the fine-grained and coarse-grained information with each other, which enables them to enhance both sides. When the performance of Digger can not be improved anymore on the validation set, we output its best factor.

4 Algorithm Analysis

In this section, we mathematically study how our approach works. We first prove that the distillation item in the loss function can be regarded as an adaptive version of Label Smoothing, which works as a data-driven regularization to prevent over-fitting. We then explain how the updated prior knowledge at each iteration allows the two components of Digger-Guider to be calibrated progressively.

To present our conclusions more clearly, we consider the stock trend prediction a classification problem, where the labels are one-hot coded classes $y_i \in \{rising, falling, unchanged\}$, as defined in previous work [3], [11], [12]. We reach similar conclusions under regression settings following the same method. For a given example, the model outputs the likelihood assigned to the i-th class as $p_i = \mathrm{softmax}(z_i) = \frac{exp\{z_i\}}{\sum_{j=1}^{k} exp\{z_j\}}$, where z_i denotes the logit output. That is, $\boldsymbol{p} = g(\boldsymbol{e}; \phi)$, where $g(\cdot; \phi)$ indicates the prediction model. We choose to use cross-entropy $\mathcal{H}$ as loss in our framework as it's very popular. Then Eqn.(1) can be embodied as: $\mathcal{L}(g(f(\boldsymbol{x}; \theta); \phi), \boldsymbol{y}) = \mathcal{H}(g(\boldsymbol{e}; \phi), \boldsymbol{y}) = \mathcal{H}(\boldsymbol{p}, \boldsymbol{y}) = \sum_{i=1}^{M} -y_i \log p_i$, where M is the number of samples.

4.1 Data-Driven Regularization Prevents Over-Fitting

Label Smoothing (LS) is a widespread method for a multi-class neural network to improve generalization ability, which can be viewed as a modular regularization [30], [31]. For LS, it minimizes loss between smoothed labels y_i^{LS} and network output p_i, where y_i^{LS} is formulated as:

$$y_i^{LS} = (1-\lambda) y_i + \lambda u_i, \tag{6}$$

which is a mixture of label $\boldsymbol{y}$ and a fixed distribution $\boldsymbol{u}$, with hyper-parameter $\lambda \in [0,1]$. Usually, $\boldsymbol{u}$ is an uniform distribution (e.g., $u_i = \frac{1}{M}$). The cross-entropy loss $\mathcal{L}^{LS}$ defined over $\boldsymbol{y}^{LS}$ is

$$\begin{aligned} \mathcal{L}^{LS} &:= \mathcal{H}\left(\boldsymbol{p}, \boldsymbol{y}^{LS}\right) \\ &= \sum_{i=1}^{M} -((1-\lambda) y_i + \lambda u_i) \log p_i \\ &= (1-\lambda)\mathcal{H}(\boldsymbol{p}, \boldsymbol{y}) + \lambda \mathcal{H}(\boldsymbol{p}, \boldsymbol{u}). \end{aligned} \tag{7}$$

For our framework, when the $\mathrm{dist}(\cdot,\cdot)$ denotes knowledge distillation computed by the prediction model and cross-entropy, i.e., $\mathrm{dist}(\boldsymbol{e_d}, \boldsymbol{e_g}) = \mathcal{H}(g(\boldsymbol{e_d}), g(\boldsymbol{e_g}))$, the losses of Digger (See Eqn.(2)) and Guider (See Eqn.(5)) can both be simply written as:

$$\mathcal{L}^{DG} = (1-\lambda)\mathcal{H}(\boldsymbol{p}, \boldsymbol{y}) + \lambda \mathcal{H}(\boldsymbol{p}, \boldsymbol{p}'), \tag{8}$$

where $\boldsymbol{p}'$ is the predicted probability from Digger/Guider which is used as the target of distillation. Comparing Eqn.(7) with Eqn.(8), we find the two loss functions have similar forms. We define the loss of Digger-Guider $\mathcal{L}^{DG} := \mathcal{H}\left(\boldsymbol{p}, \boldsymbol{y}^{DG}\right)$, where $\boldsymbol{y}^{DG}$ is formulated as

$$y_i^{DG} = (1-\lambda) y_i + \lambda p_i'. \tag{9}$$

Comparing Eqn.(9) with Eqn.(6), we find that Digger-Guider can be taken as an adaptive case of LS. Distribution $\boldsymbol{p}'$ is learned rather than fixed $\boldsymbol{u}$, suggesting that the output of the generalized *Teacher* model can be regarded as data-driven prior knowledge to smooth the label for the *Student* model.

We conclude Digger-Guider can inherit most of the benefits of LS, such as model regularization and better calibration [30]. It is also more flexible than LS, since its smoothed labels are learnable.

4.2 Prior Knowledge Update Helps Learning

We look deep into the iteration details to better understand how iterative learning works. Taking Guider of iteration $k (k > 1)$ as an example, we rewrite Eqn.(5) as:

$$\begin{aligned}\mathcal{L}_g^k &= (1-\lambda)\mathcal{H}(\boldsymbol{p}_g^k, \boldsymbol{y}) + \lambda\mathcal{H}(\boldsymbol{p}_g^k, \boldsymbol{p}_d^{k-1}) \\ &= \mathcal{H}(\boldsymbol{p}_g^k, \boldsymbol{p'}_g^k) = \sum_{i=1}^{M} -{p'}_{g,i}^k \log p_{g,i}^k,\end{aligned} \tag{10}$$

where $\boldsymbol{p}_g^k$ denotes the output of Guider and $\boldsymbol{p}_d^{k-1}$ denotes the output from the trained Digger of iteration $k-1$. Note that ${p'}_{g,i}^k$ is a newly defined smoothed label, which is formulated as ${p'}_{g,i}^k = (1-\lambda)y_i + \lambda p_{d,i}^{k-1}$. This indicates that adding distillation item (i.e., $\lambda\mathcal{H}(\boldsymbol{p}_d^{k-1}, \boldsymbol{p}_g^k)$) into Guider's loss is actually equivalent to softening the hard label by Digger. In this way, detailed embeddings learned by Digger can be transferred to Guider in terms of adaptive label smoothing, resulting in embeddings with richer and robust patterns.

Similarly, for Digger of iteration $k (k > 1)$, we can rewrite Eqn.(2) as below:

$$\begin{aligned}\mathcal{L}_d^k &= (1-\lambda)\mathcal{H}(\boldsymbol{p}_d^k, \boldsymbol{y}) + \lambda\mathcal{H}(\boldsymbol{p}_d^k, \boldsymbol{p}_g^k) \\ &= \mathcal{H}(\boldsymbol{p}_d^k, \boldsymbol{p'}_d^k) = \sum_{i=1}^{M} -{p'}_{d,i}^k \log p_{d,i}^k,\end{aligned} \tag{11}$$

where ${p'}_{d,i}^k = (1-\lambda)y_i + \lambda p_{g,i}^k$ is a similar softened label. Let $p_{g,i}^k = {p'}_{g,i}^{k-1} + \Delta_{g,i}^k$. Without loss of generality, we regard $\Delta_{g,i}^k$ as gains (transferred knowledge) of Guider in the iteration k. We expand ${p'}_{d,i}^k$ according to the definition of $p_{g,i}^k$ and $\Delta_{g,i}^k$ as below:

$$\begin{aligned}{p'}_{d,i}^k &= (1-\lambda)y_i + \lambda p_{g,i}^k \\ &= (1-\lambda)y_i + \lambda({p'}_{g,i}^{k-1} + \Delta_{g,i}^k) \\ &= (1-\lambda)y_i + \lambda((1-\lambda)y_i + \lambda p_{d,i}^{k-2} + \Delta_{g,i}^k) \\ &= (1-\lambda^2)y_i + \lambda^2 p_{d,i}^{k-2} + \lambda\Delta_{g,i}^k.\end{aligned} \tag{12}$$

We find that the embedding ${p'}_{d,i}^k$ in the last equation can be decomposed into two components: label smoothing items and prior knowledge from Guider. Label smoothing items $(1-\lambda^2)y_i + \lambda^2 p_{d,i}^{k-2}$ indicate the targets here are actually softened by historical Digger at iteration $k-2$, which acts as a data-driven regularization, leading to better generalization [30]. $\Delta_{g,i}^k$ is the updated prior knowledge at iteration k. The prior knowledge transferred from Guider helps Digger to overcome the noise.

In summary, Digger-Guider is more than an adaptive LS. The benefits of our proposed framework are threefold: (1) Digger and Guider share the global and fine-grained information to obtain complementary signal alternatively; (2) The distillation item added into the loss functions (see Eqn.(2) and Eqn.(5)) not only lets Digger and Guider assimilate to each other, but also acts as a data-driven regularization in terms of an adaptive version of label smoothing to improve generalization abilities. (3) The prior knowledge (i.e., $\Delta_{g,i}^k$) used for distillation is incremented and updated at each iteration, allowing the model to be calibrated progressively.

5 EXPERIMENTS

In this section, we empirically evaluate our Digger-Guider framework in real stock markets. Our learned high-frequency factors are applied to the stock trend prediction and trading simulation tasks. We also conduct a series of analytical experiments to get in-depth insight into how it works.

5.1 Experiment Setup

We introduce the datasets, comparison methods, implementation details, and evaluation metrics in this section. We take daily frequency as low frequency and 15-minute frequency as high-frequency to predict daily trend labels, though our framework can be applied to any frequency data, even higher-frequency data.

5.1.1 *Dataset*

We evaluate our models on real-world stock data. We collect stock sequences from Qlib[1], an AI-oriented quantitative investment platform. Our dataset consists of low-frequency (daily) and high-frequency (15-min) price-volume stock data over constituent stocks from two major global stock indices: CSI300 and NASDAQ100. They are calculated by the stock prices and capitals of the selected 300 and 100 stocks in different stock exchanges. In our experiment, CSI300 includes 770 stocks with 968,908 samples while NASDAQ100 includes 171 stocks with 559,586 samples. According to Qlib, the stock data are adjusted for dividends and splits. We select high-frequency stock sequences in the past 20 days as inputs. CSI300 and NASDAQ100 consist of six [2] and five [3] features at each time step and there are 4 and 6.5 hours of continuous transaction time for each trading day, respectively. Thus the input lengths of the 15-min sequences are $20 \times 16 = 320$ and $20 \times 26 = 520$.

The CSI300 training set ranges from Feb. 16, 2007 to Dec. 31, 2013, while the NASDAQ100 training set ranges from Jan. 01, 2005 - Dec. 31, 2013. The time ranges of the validation set and test set are the same for both datasets. The validation set ranges from Jan. 01, 2014 to Dec. 31, 2015 and the test set ranges from Jan. 01, 2016 to Jun. 01, 2020.

5.1.2 *Comparison Methods*

We compare our method with other baseline models to evaluate performance. The first group is classical time series forecasting models, including *ARIMA* [32] and *Linear Regression*. The second group is recent deep learning models, including *MLP* [33], *Transformer* [34], *DailyRNN* that uses

1. https://github.com/microsoft/qlib
2. highest price, opening price, lowest price, closing price, volume-weighted average price, trading volume
3. except volume-weighted average price

TABLE 1
Performance of stock trend prediction on CSI300 and NASDAQ100.

Method	CSI300		NASDAQ100	
	RMSE(%)	MAE(%)	RMSE(%)	MAE(%)
ARIMA	2.4481 ± –	1.3679 ± –	6.0908 ± –	1.5285 ± –
Linear Regression	3.3470 ± 0.1549	2.2473 ± 0.1199	5.2025 ± 0.2377	2.2899 ± 0.0777
MLP	2.3033 ± 0.0888	1.7507 ± 0.1048	4.3010 ± 0.6439	3.6776 ± 0.7352
Transformer	3.0789 ± 0.0731	2.4248 ± 0.0988	3.7831 ± 0.3748	2.8390 ± 0.4777
Daily RNN	2.1261 ± 0.0441	1.5636 ± 0.0543	3.0970 ± 0.1970	2.7963 ± 0.1566
SFM	3.1299 ± 0.1631	2.6757 ± 0.1769	2.9470 ± 0.1867	2.0908 ± 0.2318
ALSTM	2.0538 ± 0.0108	1.4675 ± 0.0141	3.6060 ± 0.1186	2.6112 ± 0.2055
Adv-ALSTM	2.1160 ± 0.0283	1.5508 ± 0.0343	3.3874 ± 0.0800	2.5481 ± 0.1251
Informer	2.0416 ± 0.0228	1.4410 ± 0.0300	2.8891 ± 0.1111	1.9432 ± 0.0966
ETSformer	2.0323 ± 0.0210	1.4324 ± 0.0278	2.8060 ± 0.0954	1.8463 ± 0.1188
Coarse-Grained Model	2.2864 ± 0.0349	1.6444 ± 0.0388	3.7599 ± 0.3772	2.9251 ± 0.4785
Fine-Grained Model	2.3159 ± 0.0520	1.6455 ± 0.0462	3.7340 ± 0.2722	2.9096 ± 0.3631
Ensemble	2.2560 ± 0.0406	1.6030 ± 0.0427	3.4325 ± 0.3955	2.5958 ± 0.4918
Digger-Guider	**1.9669 ± 0.0021**	**1.3499 ± 0.0025**	**2.4431 ± 0.0049**	**1.4783 ± 0.0055**

daily-frequency data only, *SFM* [7], *ALSTM* [35], *Adv-ALSTM* [3], *Informer* [23] and *ETSformer* [24]. Among them, Informer and ETSformer are very representative works in the field of time series prediction in recent years. The third group consists of variants of our methods. *Coarse-Grained Model* has same structure as Guider. *Fine-Grained Model* has the same structure as Digger. They are used to verify the effectiveness of knowledge distillation and mutual distillation. *Ensemble* is the ensemble of Fine-Grained Model and Coarse-Grained Model. We carefully select the hyper-parameters of the baselines and report the performance of the best settings.

- **ARIMA** [32]: *Autoregressive Integrated Moving Average Model*, a traditional time series forecast model.
- **Linear Regression**: the straightforward method for regression.
- **MLP** [33]: *Multilayer Perceptron*, the multi-layer fully connected neural network model.
- **Transformer** [34]: composed of the Attention mechanism and is designed to handle sequential data.
- **Daily RNN**: two-layer GRU using daily data as input.
- **SFM** [7]: *State Frequency Memory* is a recent work inspired by Fourier Transform. It aims to capture trading patterns from investors with different trading modes.
- **ALSTM** [35]: *Attentive LSTM* consists of a normal LSTM with an input attention layer and a temporal attention layer.
- **Adv-ALSTM** [3]: *Adv-ALSTM*leverages adversarial training during model training, which is a variant of ALSTM and seen as state-of-the-art in using daily-frequency stock data.
- **Informer** [23]: *Informer* is an efficient transformer-based model for long sequence time-series forecasting that uses a novel self-attention mechanism, a self-attention distilling technique, and a generative style decoder, which won the best paper award at AAAI 2021.
- **ETSformer** [24]: *ETSformer* is a novel time-series Transformer architecture that leverages the exponential smoothing attention (ESA) and frequency attention (FA) to improve the accuracy and efficiency of time-series forecasting.
- **Coarse-Grained Model**: has the same structure as the Guider.
- **Fine-Grained Model**: has the same structure as the Digger. These two models take high-frequency stock sequences as inputs. They are used to verify the effectiveness of knowledge distillation and mutual distillation.
- **Ensemble**: The above baselines are the single-model method. We also develop an ensemble of the Fine-Grained Model and Coarse-Grained Model. We tried all possible average weights of Fine-Grained Model and Coarse-Grained Model and reported the best result. Specifically, we used a grid search in range[0,1] to try different weight combinations and selected the combination that achieved the best performance.

5.1.3 Implementation Details

Though we develop the Digger-Guider framework with the following structures, other implementations are also possible. We use an RNN model to construct the Digger since it can process temporal data step by step. We use a CNN model to build the Guider since it can process the temporal data in several steps together within the same conception field. Specifically, we implement Digger f_d as a two-layer GRU, with a hidden sizes of 64 for both layers. The prediction model g is also set as a two-layer GRU with a hidden sizes of 64. Digger and the prediction model are stacked together to form an end-to-end structure for training. Since CNN is usually not as strong as RNN in modeling temporal data, we design a coarse-Grained CNN as Guider f_g. For the CSI300 dataset, each day we transform the high-frequency data into a 2D frame sized 16×6, whose rows denote the minute timeline and columns denote attributes. We use 1 layer of 2D convolution with a kernel size of 16×1 and 3 layers of 1D convolution with kernel sizes of $\{4, 2, 2\}$ as h_{ind} to extract indicators. Similarly, for the NASDAQ100 dataset, each day we transform the high-frequency data into a 2D frame with size 26×5, whose rows denote the minute timeline and columns denote attributes. We use 1 layer of 2D convolution with a kernel size of 26×1 and 2 layers of 1D convolution with kernel sizes of $\{13, 2\}$ as h_{ind} to extract indicators. Other hyper-parameters are the same for both datasets. A fully connected layer with hidden size of 64 is used as h_{emb}. The indicators in Rule-based Guider are original daily-frequency features. Guider and the prediction model are also stacked to form an end-to-end structure for training. We set λ_1 and λ_2 as 0.5 for all experiments.

We delicately select the architectures of baseline models and report the best performance with the following hyper-parameters: *ARIMA*: p,d,q = 1,2,1. *MLP*: 3 layers with 256, 128, and 64 hidden sizes respectively. *Transformer*: 4 heads and 2 encoder layers. *Daily RNN*: 2 layers GRU with 20 time steps. *SFM*: the dimension of the state module is 64 and the dimension of the frequency module is 10. *ALSTM*: the hidden size of GRU is 64 and the hidden size for attention net is 32. *Adv-ALSTM*: has the same parameters as ALSTM, and the λ parameter for adversarial training loss is 0.01. We adjusted part of the hyper-parameters of *Informer* and *ETSformer* to make the model adapt to our task and kept the rest unchanged. Other important hyper-parameters are the same for our approach and baseline methods. They are set as follows: The input window size T is 20. We train each daily data in a batch and randomly shuffle the training dataset before training. For data normalization, we prevent data leakage carefully. In practice, we use the mean and standard deviation values of the training set to do the z-score normalization for the whole dataset. All the weights and biases are initialized from the Xavier uniform distribution. All the models are optimized by an Adam optimizer with a learning rate of 10^-3. We repeated each experiment 20 times to report the average results and the standard deviations.

5.1.4 Evaluation Metrics

We take classical forecasting criteria like *Root Mean Squared Error (RMSE)* and *Mean Absolute Error (MAE)* as the evaluation metrics. Note that the smaller RMSE and MAE mean the better performance of the model. We calculate them as follow: $\mathrm{RMSE} = \sqrt{\frac{1}{NT}\sum_t^T \sum_i^N \left(y_t^{(i)} - \hat{y}_t^{(i)}\right)^2}$, $\mathrm{MAE} = \frac{1}{NT}\sum_t^T \sum_i^N \left|\left(y_t^{(i)} - \hat{y}_t^{(i)}\right)\right|$

5.1.5 Quantitative Performance Indicators

We introduce the quantitative performance indicators in the following.

- **Rank Information Coefficient (Rank IC)** means the rank correlation coefficient between the two sequences. We calculate the average Rank IC among the testing interval and all stocks in the dataset as follow: $Rank\ IC = \frac{\sum_t^T corr(\hat{\boldsymbol{y}}_t^{(i)}, \boldsymbol{y}_t^{(i)})}{T}$, where $\hat{\boldsymbol{y}}_t^{(i)}$ and $\boldsymbol{y}_t^{(i)}$ denote the predicted and real stock sequence of stock i at day t respectively. T denotes the number of days in test set. And corr($\hat{\boldsymbol{y}}_t^{(i)}, \boldsymbol{y}_t^{(i)}$) is the *spearman correlation coefficient* [36] between above two sequences.
- **Rank Information Ratio (Rank IR)** is obtained by Rank IC taking into standard deviation, which measures the stability of Rank IC: $Rank\ IR = \frac{Rank\ IC}{\sigma[corr(\hat{\boldsymbol{y}}_t^{(i)}, \boldsymbol{y}_t^{(i)})]}$.
- **Annualized Return (AR)** means the equivalent annual return an investor receives over a given period: $AR = \left(\frac{P_{end}}{P_{start}}\right)^{\frac{n_0}{n}} - 1$, where P means the total value of the held stocks and cash, n_0 means the trading days per year, and n means the trading days over the trading period.
- **Information Ratio (IR)** considers benefits and risks synthetically and reflects returns above the returns of the benchmark: $IR = \frac{\mathbb{E}[\mathbf{R}]}{\sigma[\mathbf{R}]}$, where $\mathbf{R}$ denotes the sequence of active returns (i.e., returns above the returns of the benchmark), $\mathbb{E}[\mathbf{R}]$ is the expected value of the active return, and $\sigma[\mathbf{R}]$ is the standard deviation of the active return.
- **Maximum Drawdown (MDD)** measures the largest decline over the test period and shows the worst scenario: $MDD = \max \frac{P_i - P_j}{P_i}, j > i$ where i and j denote two different moments over the test period, and P means the total value of the held stocks and cash.

5.2 Main Experiment Results

In this section, we conduct stock trend prediction to evaluate our factor extractor model and conduct a market trading simulation to verify our model's profitability. Moreover, we compare our model with other methods to analyze the effectiveness of Digger-Guider in preventing over-fitting.

5.2.1 Price Trend Prediction

We evaluate our factor extractor model by using the extracted embeddings on stock trend prediction. The performance of our method and baseline models are shown in Table 1. From the table, we have some observations and inferences. (1) None of the single model high-frequency baselines (Fine-Grained Model, Coarse-Grained Model, SFM, ALSTM, Adv-ALSTM, Informer, ETSformer) achieve stable and significant improvement compared to the low-frequency baseline (Daily RNN). These results show that the noise contained in higher-frequency data is overwhelming. The poor performance of the Fine-Grained Model indicates that it suffers from over-fitting due to the noise buried in high-frequency data. Meanwhile, the low accuracy of the Coarse-Grained Model denotes that it is unable to generate robust static features independently due to its low expressiveness. (2) Directly combining models with different granularities cannot bring gains compared to the Fine-Grained Model and the Coarse-Grained Model. The effect of the Ensemble model is not stable since it is sensitive to noise. As another approach to overcoming noise and fusing models, our method can resist noise and improve performance steadily.

Digger-Guider achieves the best RMSE and MAE on both CSI300 and NASDAQ100 datasets, exceeding SFM, ALSTM and Adv-ALSTM to a large margin. It also outperforms the Informer and TSformer, representative works on time series prediction. This is an exciting result for stock embedding learning, as our Digger-Guider is the first model designed for stock embedding from high-frequency data and achieves significant improvement. All these results indicate that our proposed model can overcome noise and generate good stock embedding. Therefore, it can help investors to make a more accurate price trend prediction.

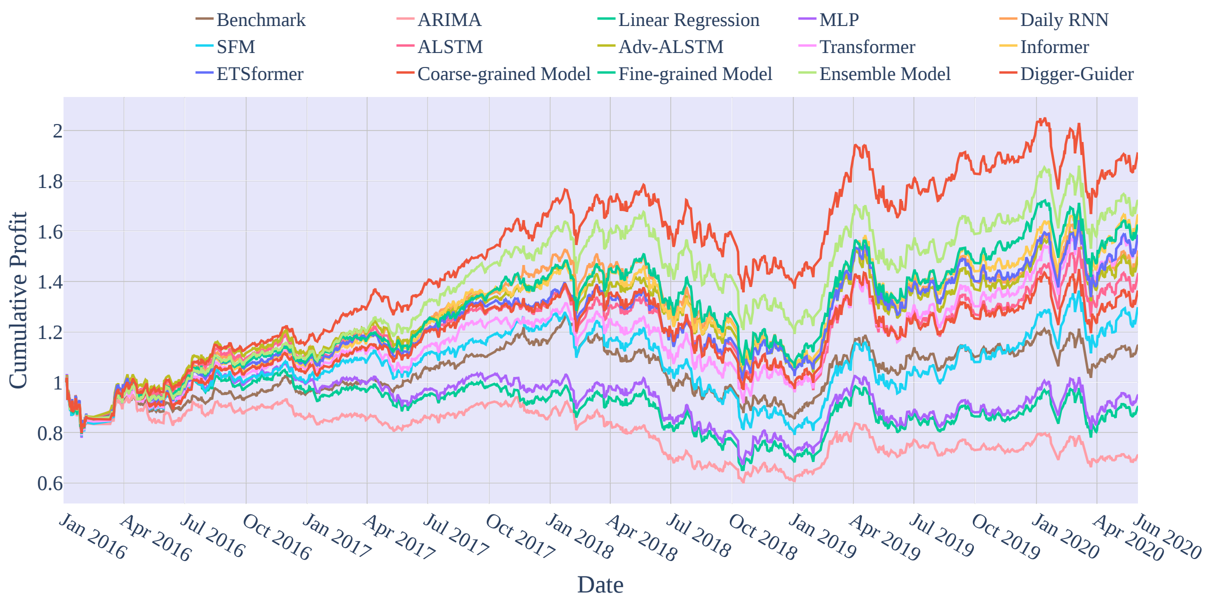


Fig. 4. Cumulative profit with Topk-Drop trading strategy of Digger-Guider model and other baseline methods.

5.2.2 Market Trading Simulation

To verify our method's profitability in real financial investing scenarios, we conduct a market trading simulation. We employ the *Topk-Drop Strategy* [5], [37], which is a simple but popular trading strategy. Initially, investors invest in the *Top K* stocks with the highest predicted ranking score. On each trading day, the *Drop* number of held stocks with the worst prediction score will be sold, and the same number of unheld stocks with the best prediction score will be bought. We conduct back-testing on real stock data with considering practical constraints (i.e., with 0.15% opening fee, 0.25% closing fee and 9.5% price limit threshold). Note that *Topk-Drop* algorithm always sells *Drop* stocks every trading day, which guarantees a fixed turnover rate.

Figure 4 compares the cumulative profit curves corresponding to different approaches. The *Benchmark* in the figure denotes the CSI300 index itself, which reflects the overall performance of the market. From the figure, we find that Digger-Guider achieves the highest cumulative profit over almost the entire testing period despite the market volatility. At the end of the test period, after deducting practical constraints, the *Annualized Return* (AR) of the benchmark (brown line) is 5.14%. The AR of Digger-Guider (red line) is 17.78%, indicating that the trading strategy based on Digger-Guide can achieve an excess return rate of nearly 12.64%.

We conduct further quantitative analysis on trading strategies in Table 2, including rank correlation (*Rank IC* and *Rank IR* [6]), return indicator (*AR*), risk indicators (*Maximum Drawdown, MDD* [38]) and risk-adjusted return (*Information Ratio*, IR [39]). Digger-Guider achieves the best performance over all these quantitative indicators, which indicates that our model can make a steady profit. For instance, the IR of Digger-Guider is the highest, reaching 0.82, while the IR of the benchmark is around 0.27, which indicates that the trading based on Digger-Guider can earn the highest excess of the risk-free rate per unit of volatility.

5.2.3 The Effectiveness of Preventing Over-fitting

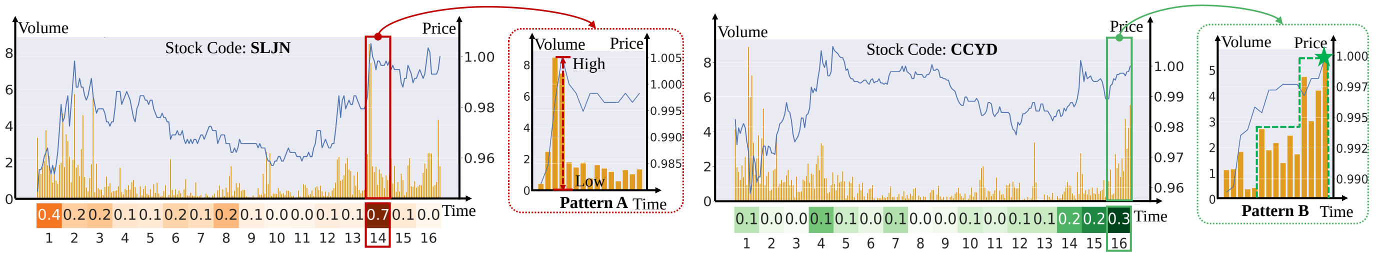
Since our method works as a data-driven regularization to resist noise in high-frequency data (see Section 4), we compare our method with some other common tricks that can prevent over-fitting, including dropout [40], early stop [41] and L2-Norm regularization [42]. Table 3 shows the performance of the Fine-Grained Model and three models combined with popular tricks on the CSI300 dataset. From the table, we observe that with any of the three tricks added, there is no significant improvement of performance on the training and validation set. On the test set, only L2-Norm achieves minor improvements. In contrast, our method significantly outperforms other approaches. The reason is that our model works as a data-driven regularization method. While the other methods roughly restrict the capacity of models. They are irrelevant to the data. Besides, we conduct a ablation study on mutual distillation. We presented the results of Fine-grained Model + Rule-based Guider on Table 3, which means that the Fine-grained Model is only guided from the initial version of Guider. In fact, this is equivalent to $\lambda_1 = 0.5$ in Eqn. (2) and $\lambda_2 = 0$ in Eqn. (5). In this case, the regularization term is data-driven, but the generalization ability of this model is not as good as that of the full version of Digger-Guider. The results of this ablation experiment demonstrate the effectiveness of mutual distillation. By introducing the mutual distillation technique, our Guider model not only regularizes the function class of Digger but also provides robust knowledge to Digger. Therefore, Digger can overcome noise and achieve excellent performance.

5.3 Examples of Discovered Trading Patterns

Through in-depth model analysis, we find some crucial patterns implying price trend recognized by our model from the response values of the convolution kernel of Guider. Here we illustrate two of them as examples. One is denoted as *Pattern A*, which indicates *fluctuation with large volume*. And the other implies *both volume and price rise at the end of the day*,

TABLE 2
Quantitative performance analysis of trading strategies in the market simulation.

Method	Rank IC	Rank IR	AR	IR	MDD
Benchmark	-	-	0.0514	0.2688	-0.3704
ARIMA	-0.0065	-0.1322	-0.0583	-0.2718	-0.4763
Linear Regression	0.0515	0.2850	0.0059	0.0245	-0.4461
MLP	0.0600	0.3636	0.0176	0.0726	-0.4366
Transformer	0.0715	0.5039	0.1373	0.6000	-0.3351
Daily RNN	0.0895	0.6525	0.1269	0.5523	-0.3533
SFM	0.0833	0.5891	0.0982	0.4042	-0.4504
ALSTM	0.0869	0.6205	0.1125	0.4912	-0.3349
Adv-ALSTM	0.0893	0.6438	0.1232	0.5362	-0.3353
Informer	0.0854	0.6180	0.1477	0.6534	-0.3247
ETSformer	0.0857	0.6185	0.1359	0.5998	-0.3074
Coarse-Grained Model	0.0885	0.6428	0.1006	0.4326	-0.3393
Fine-Grained Model	0.0919	0.6567	0.1407	0.6236	-0.3328
Ensemble	0.0961	0.6744	0.1551	0.6903	-0.3210
Digger-Guider	**0.1010**	**0.7376**	**0.1778**	**0.8219**	**-0.2595**



(a) Pattern A: *Fluctuation with large volume*

(b) Pattern B:*Both volume and price rise at the end of the day*

Fig. 5. Illustration of two trading patterns recognized by Digger-Guider from two stocks: SLJN and CCYD.

TABLE 3
RMSE of different methods to prevent over-fitting.

Method	Training	Validation	Test
Fine-Grained Model	**1.3949**	1.4683	2.1812
+ Dropout	1.4138	1.4710	2.2001
+ Early Stop	1.4197	1.4710	2.1815
+ L2-Norm	1.4153	1.4669	2.1322
+ Rule-based Guider	1.4126	1.4653	2.0457
Digger-Guider (Ours)	1.4072	**1.4607**	**1.9669**

which we call *Pattern B*. These patterns provide insights into the interpretability of the model.

We take two cases as examples to illustrate these patterns. As shown in Figure 5, Pattern A and Pattern B are extracted from two stocks: SLJN and CCYD, on May. 24, 2017. The label of SLJN is located at the bottom 10%, and the label of CCYD fails in the top 10% of all stocks. We present the response values of two different CNN kernels in orange and green, respectively. For SLJN, we see that the response value of Guider is the largest on the 14th time bar of the day. SLJN experiences a violent fluctuation in volume and price. Volume surges in two minutes, and then returns to its normal level. This is an apparent trace of Pattern A. By combining Pattern A and other information, Digger-Guider predicts that the price trend of SLJN is likely to be at the bottom of the stock pool. For CCYD, it is obvious that the response value of Guider is largest on the last time bar of the day, and that there is a trend of rising volume and price at the end of the day, which is an apparent trace of Pattern B. By combining Pattern B and other information, our approach predicts that the price trend of CCYD is very likely to be at the top of the stock pool.

We further calculate the proportion of cases that are consistent with the above rules and verify that these two patterns are suggestive over the whole dataset. We conclude that investors can benefit from these comprehensible patterns and make better decisions.

6 Other Analysis

In this section, we conduct analytical experiments, including selecting the granularity of Guider, comparison with intermediate-frequency models, and the effectiveness of mutual distillation.

6.1 Processing Granularity of Guider

Guider is proposed as a simple but robust model. Naturally, the statistical caliber of Guider determines the granularity to exploit high-frequency data. To find a decent caliber, we tested Digger guided by Guider with different granularities. We analyze the model performance in different granularities of Guider using the same frequency data. In addition to the daily frequency, we implement several granularities of high-frequency data on the CSI300 dataset, from top to bottom, from coarse to fine, as summarized in Table 4. The performances of different granularities are shown in Figure

TABLE 4
Different granularity design of Guider

Granularity	Kernel Size	Information Flow
Level 1	$\{16\}$	$15min \rightarrow 4h$
Level 2	$\{8, 2\}$	$15min \rightarrow 2h \rightarrow 4h$
Level 3	$\{4, 4\}$	$15min \rightarrow 1h \rightarrow 4h$
Level 4	$\{4, 2, 2\}$	$15min \rightarrow 1h \rightarrow 2h \rightarrow 4h$
Level 5	$\{2, 2, 2, 2\}$	$15min \rightarrow 30min \rightarrow 1h \rightarrow 2h \rightarrow 4h$

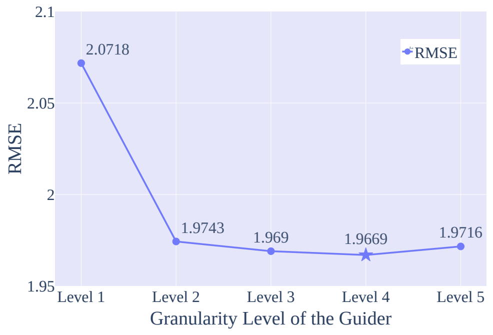


Fig. 6. Performance of Digger-Guider on different granularities.

6. According to the figure, the performance first increases and then decreases as the granularity becomes finer. There is an optimal point among different levels of granularities. The best granularity is located in Level 4. We make the following inferences. 1) If the granularity of Guider is very coarse, Guider will under-fit to the high-frequency information. The poor-performance Guider cannot help Digger. 2) If the granularity of Guider is very fine, it is difficult to resist noise. This means Guider cannot guide Digger as well. Therefore, a careful selection of Guider's granularity is vital.

6.2 Comparison with Intermediate-Frequency Models

Digger-Guider can attain proper model granularity by interacting between fine-grained and coarse-grained models, so one may be curious about whether directly applying an intermediate-frequency model can achieve good results as. We develop several intermediate-frequency models by resampling the high-frequency data to daily (240-minute), 120, 60, 48, 30 and 15-minute data. Then we use these intermediate-frequency data to construct intermediate-frequency models.

Figure 7 shows the performance of intermediate-frequency models and our Digger-Guider on CSI300 dataset. We find that as the data frequency increases (Daily $\rightarrow$ 120min), the performance of intermediate models improves gradually, which indicates that the fine-grained data does contain more effective information. However, as the data frequency further increases (60min $\rightarrow$ 48 min $\rightarrow$ 30min $\rightarrow$ 15min), the performance of intermediate models declines, which discloses that the noise hidden in the finer-grained data begins to dominate. Furthermore, although the 120min model achieves the best performance (RMSE 2.0031), it still has a big gap with Digger-Guider (RMSE 1.9669), which verifies that directly applying intermediate data does not work as well as our method. In summary, Digger-Guider does not just resample the intermediate-frequency data but also balances information utilization and noise tolerance via mutual distillation.

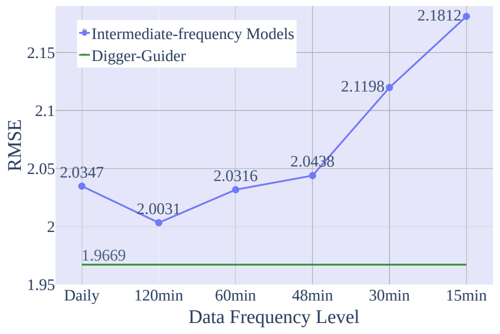


Fig. 7. Comparison with intermediate-frequency models.

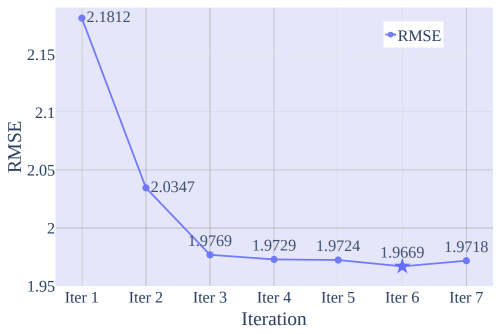


Fig. 8. Performance of Digger-Guider at each iteration.

6.3 Effectiveness of Mutual Distillation

We also study the effectiveness of mutual distillation. Figure 8 shows the performance of Digger-Guider at each iteration. With an increase of the number of iterations (Iter 1 $\rightarrow$ Iter 6), RMSE improves, indicating that the stock factors are refined continually. As the learning process terminates when Digger's performance stops increasing (Iter 7), the gradual improvement of Digger is essentially a process of balancing valuable and noisy information adaptively.

7 Conclusion and Future Work

In this paper, we study how to extract informative stock factors from noisy high-frequency data. We propose a novel learning framework, Digger-Guider. We use Guider to learn robust global features, while Digger is designed to learn rich, detailed signals. We then develop mutual distillation to share the information between Digger and Guider. Extensive experiments on stock markets demonstrate the effectiveness of our framework, with the performance of Digger-Guider significantly improving price trend prediction. We also show that our method can capture the invaluable high-frequency representation of stocks from noisy and volatile data. Our work is one of the first few studies of feature extraction from high-frequency price and volume data. In the future, we plan to further study how to leverage finer-granularity data and how to fuse multi-frequency stock factors to improve stock trend prediction.

In the future, we plan to further study how to leverage high-frequency data to improve stock trend prediction.

8 Limitation

We conclude the limitations of the study as follows: 1) The proposed Digger-Guider framework is evaluated on stock market datasets, and its effectiveness on other types of financial data or in other domains is not explored. 2) The framework requires a large amount of high-frequency data for training, which may not be available or feasible for some applications. 3) Although we have repeated the experiments multiple times and conducted fair comparisons with baselines to ensure the reliability and robustness of the results, it should be noted that the performance of the framework may still depend on the choice of hyperparameters and the specific implementation details.

References

[1] N. Barberis and R. H. Thaler, "A survey of behavioral finance," *Handbook of The Economics of Finance*, pp. 1053–1128, 2002.

[2] R. J. Shiller, "From efficient markets theory to behavioral finance," *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 83–104, 2003.

[3] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," *IJCAI*, 2019.

[4] C. Li, D. Song, and D. Tao, "Multi-task recurrent neural networks and higher-order markov random fields for stock price movement prediction: Multi-task rnn and higer-order mrfs for stock price classification," in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 1141–1151.

[5] C. Chen, L. Zhao, J. Bian, C. Xing, and T.-Y. Liu, "Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction," in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 2376–2384.

[6] Z. Li, D. Yang, L. Zhao, J. Bian, T. Qin, and T.-Y. Liu, "Individualized indicator for all: Stock-wise technical indicator optimization with stock embedding," in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 894–902.

[7] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD*, 2017, pp. 2141–2149.

[8] R. D. Edwards, J. Magee, and W. C. Bassetti, "Technical analysis of stock trends," *CRC press*, 2018.

[9] X. Zhu, S. Gong *et al.*, "Knowledge distillation by on-the-fly native ensemble," in *Advances in neural information processing systems (NeurIPS)*, 2018, pp. 7517–7527.

[10] J. J. Murphy, "Technical analysis of the financial markets: A comprehensive guide to trading methods and applications," *Penguin*, 1999.

[11] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, "Clvsa: A convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3705–3711.

[12] G. Liu, Y. Mao, Q. Sun, H. Huang, W. Gao, X. Li, J. Shen, R. Li, and X. Wang, "Multi-scale two-way deep neural network for stock trend prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 7 2020, pp. 4555–4561, special Track on AI in FinTech.

[13] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale gaussian transformer for stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 7 2020, pp. 4640–4646, special Track on AI in FinTech.

[14] F. Black, "Noise," *Journal of Finance*, vol. 41, no. 3, pp. 529–543, 1986.

[15] J. Zou, Q. Zhao, Y. Jiao, H. Cao, Y. Liu, Q. Yan, E. Abbasnejad, L. Liu, and J. Q. Shi, "Stock market prediction via deep learning techniques: A survey," *arXiv preprint arXiv:2212.12717*, 2022.

[16] A. Thakkar and K. Chaudhari, "Fusion in stock market prediction: a decade survey on the necessity, recent developments, and potential future directions," *Information Fusion*, vol. 65, pp. 95–107, 2021.

[17] A. Thakkar, D. Patel, and P. Shah, "Pearson correlation coefficient-based performance enhancement of vanilla neural network for stock trend prediction," *Neural Computing and Applications*, vol. 33, pp. 16 985–17 000, 2021.

[18] A. Thakkar and K. Chaudhari, "A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions," *Expert Systems with Applications*, vol. 177, p. 114800, 2021.

[19] L. Cao, "Ai in finance: challenges, techniques, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–38, 2022.

[20] A. Thakkar and K. Chaudhari, "Information fusion-based genetic algorithm with long short-term memory for stock price and trend prediction," *Applied Soft Computing*, vol. 128, p. 109428, 2022.

[21] J. Wang, Z. Wang, J. Li, and J. Wu, "Multilevel wavelet decomposition network for interpretable time series analysis," in *Proceedings of the 24th ACM SIGKDD*, 2018, pp. 2437–2446.

[22] Y. Xu and S. B. Cohen, "Stock movement prediction from tweets and historical prices," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1970–1979.

[23] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," vol. 35, no. 12, pp. 11 106–11 115, 2021.

[24] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi, "Etsformer: Exponential smoothing transformers for time-series forecasting," 2022. [Online]. Available: https://arxiv.org/abs/2202.01381

[25] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD*, 2006, pp. 535–541.

[26] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[27] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NIPS)*, 2012, pp. 1097–1105.

[30] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 4696–4705.

[31] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3903–3911.

[32] R. G. Brown, *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

[33] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *Journal of Neural Network Computing*, vol. 2, no. 2, pp. 40–48, 1990.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[35] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *IJCAI*, pp. 2627–2633, 2017.

[36] D. G. Bonett and T. A. Wright, "Sample size requirements for estimating pearson, kendall and spearman correlations," *Psychometrika*, vol. 65, no. 1, pp. 23–28, 2000.

[37] X. Yang, W. Liu, D. Zhou, J. Bian, and T.-Y. Liu, "Qlib: An ai-oriented quantitative investment platform," *arXiv preprint arXiv:2009.11189*, 2020.

[38] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," *Risk Magazine*, vol. 17, no. 10, pp. 99–102, 2004.

[39] T. H. Goodwin, "The information ratio," *Financial Analysts Journal*, vol. 54, no. 4, pp. 34–43, 1998.

[40] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," pp. 402–408, 2000.

[42] C. Ren, D. Dai, and H. Yan, "Robust classification using l 2,1 -norm based regression model," *Pattern Recognition*, vol. 45, no. 7, pp. 2708–2718, 2012.

Yang Liu is a Research SDE at Microsoft. He received his Master's degree from University of Science and Technology of China (USTC). His research interests include AI in Financial Technology, Data Mining and Quantitative Trading. He was the recipient of KDD CUP' 19 PaddlePaddle Special Award.

Chang Xu is a Senior Researcher at Microsoft Research. Before that, she got her Ph.D. from the Joint Ph.D. Program of Microsoft Research Asia and Nankai University in 2019, and a Bachelor's degree in 2014. Her research interests include AI in Financial Technology, Reinforcement Learning, Natural Language Processing, and Computer Vision. She has published her research in many journals and conference proceedings, e.g., the IEEE Transactions on Computers, ACM MM, WWW, AAAI, IJCAI, ICME, and CIKM.

Min Hou received her Bachelor's Degree in Engineering from the Hefei University of Technology, Hefei, China. She is currently pursuing her Ph.D. at the School of Data Science, University of Science and Technology of China, Hefei, China. Her current research interests include data mining, recommender systems and AI for finance. She has published papers in refereed conference proceedings, such as AAAI, IJCAI, ICDM, WSDM.

Weiqing Liu is a Principal Research Manager at Microsoft Research. He holds a Ph.D. degree in the Department of Computer Science from the University of Science and Technology of China. His research interests focus on data mining and machine learning. He is actively transferring research to significant real-world applications, especially to finance scenarios. His work has led to tens of research papers in prestigious conferences, such as KDD, ICML, WWW, WSDM, AAAI and IJCAI.

Jiang Bian is a Senior Principal Research Manager at Microsoft Research. He is leading the machine learning solutions and services group, with the main focus on designing cutting-edge machine learning algorithms into real-world application scenarios, including finance, healthcare, supply-chain and sustainability. Prior to this, he was a Scientist at Yahoo! Labs in the United States, responsible for the content optimization and personalization and Web search modules in Yahoo! Homepage. After that, he jointed a leading content distribution platform in China, i.e., Yidian Inc., and became one of the core members of this startup company, with the major responsibility of developing advanced recommendation models. Dr. Bian has authored tens of research papers in many well-recognized international conferences and has submitted a couple of US patents. He has also served as PC Member for several international conferences and Peer Reviewer for a few well-known journals. Dr. Bian graduated from Peking University in China with a bachelor's degree and then received the Ph.D. degree in computer science at Georgia Institute of Technology in the United States.

Qi Liu received the PhD degree in computer science from University of Science and Technology of China (USTC). He is currently a professor at USTC. His general area of research is data mining and knowledge discovery. He has published prolifically in refereed journals and conference proceedings, e.g., the IEEE Transactions on Knowledge and Data Engineering, the ACM Transactions on Information Systems, the ACM Transactions on Knowledge Discovery from Data, the ACM Transactions on Intelligent Systems and Technology, KDD, IJCAI, AAAI, ICDM, SDM, and CIKM. He has served regularly in the program committees of a number of conferences, and is a reviewer for the leading academic journals in his fields. He is a member of the ACM, the IEEE, and the Alibaba DAMO Academy Young Fellow. His research is also supported by the National Science Fund for Excellent Young Scholars and the Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS).

Tie-Yan Liu received the Ph.D. degree and Bachelor degree both from Tsinghua University. He is an Assistant managing director of Microsoft Research Asia, leading the machine learning research area. His seminal contribution to the field of learning to rank and computational advertising has been widely recognized, and his recent research interests include deep learning, reinforcement learning, and distributed machine learning. In particular, he and his team have proposed a few new machine learning concepts, such as dual learning, learning to teach, and deliberation learning. He is an adjunct/honorary professor at Carnegie Mellon University (CMU), University of Nottingham, and several other universities in China. He has published 200+ papers in refereed conferences and journals, e.g., SIGIR, WWW, ICML, KDD, NeurIPS, IJCAI, AAAI, ACL and so on, with around 20000 citations. He is a fellow of the IEEE, and a distinguished member of the ACM.