Learning Multiple Stock Trading Patterns with Temporal Routing Adaptor and Optimal Transport

Hengxu Lin*†
Sun Yat-sen University
Guangdong, China
v-hengxl@microsoft.com

Dong Zhou*
Microsoft Research Asia
Beijing, China
Zhou.Dong@microsoft.com

Weiqing Liu
Microsoft Research Asia
Beijing, China
Weiqing.Liu@microsoft.com

Jiang Bian
Microsoft Research Asia
Beijing, China
Jiang.Bian@microsoft.com

ABSTRACT

Successful quantitative investment usually relies on precise predictions of the future movement of the stock price. Recently, machine learning based solutions have shown their capacity to give more accurate stock prediction and become indispensable components in modern quantitative investment systems. However, the i.i.d. assumption behind existing methods is inconsistent with the existence of diverse trading patterns[1] in the stock market, which inevitably limits their ability to achieve better stock prediction performance. In this paper, we propose a novel architecture, Temporal Routing Adaptor (TRA), to empower existing stock prediction models with the ability to model multiple stock trading patterns. Essentially, TRA is a lightweight module that consists of a set of independent *predictors* for learning multiple patterns as well as a *router* to dispatch samples to different predictors. Nevertheless, the lack of explicit pattern identifiers makes it quite challenging to train an effective TRA-based model. To tackle this challenge, we further design a learning algorithm based on Optimal Transport (OT) to obtain the optimal sample to predictor assignment and effectively optimize the router with such assignment through an auxiliary loss term. Experiments on the real-world stock ranking task show that compared to the state-of-the-art baselines, e.g., Attention LSTM and Transformer, the proposed method can improve information coefficient (IC) from 0.053 to 0.059 and 0.051 to 0.056 respectively. Our dataset and code used in this work are publicly available[2].

*The first two authors have equal contribution.
†This work was done when the first author was an intern at Microsoft Research Asia.
[1]In this paper, a trading *pattern* means the causal relation between the available information at the current time (i.e., feature) and the stock price movement in the future (i.e., label).
[2]https://github.com/microsoft/qlib

KDD '21, August 14–18, 2021, Virtual Event, Singapore.
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467358

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Applied computing → Forecasting**.

KEYWORDS

computational finance; stock prediction; gated network; conditional computing; optimal transport; multi-domain learning

ACM Reference Format:
Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. 2021. Learning Multiple Stock Trading Patterns with Temporal Routing Adaptor and Optimal Transport. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467358

1 INTRODUCTION

Stock investing is one of the most popular channels for investors to pursue desirable investment goals. A successful investment usually requires precise predictions of the future movements of the stock price. Although the efficient market hypothesis (EMH) [10] states that predicting the stock movement is impossible given the market is efficient enough, many researchers found that the market is, in reality, less efficient than expected and can be potentially predicted. One example of well-observed predictable patterns is mean-reversion [23] meaning that the stock price tends to fall if it is higher than its historical average. Therefore, in some sense, the stock's future price movement can be predicted based on its historical price trend. Lately, machine learning based solutions have been employed in modern quantitative investment systems for stock prediction and prove their capacity to capture more complex patterns with powerful non-linear models from heterogeneous data sources [8, 12, 16, 31–33].

To apply machine learning algorithms for stock prediction, existing approaches usually adopt a supervised learning formulation: use observable information (e.g., price, news) as features $\mathbf{x}_i \in \mathcal{X}$ and stock return ranks or price movement directions in the future as the label $\mathrm{y}_i \in \mathcal{Y}$ per stock per day respectively, then train an estimator parameterized with θ to model the underlying distribution $\mathbb{P}$. Given a collection of N observations[3] $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{N}$ as the training data, parameter θ can be effectively optimized to capture the underlying distribution by maximum likelihood estimation or empirical risk minimization. To pursue superior stock prediction performance, a variety of features extracted from diverse information sources like price [8, 12, 33], fundamental factors [3], social media texts [16, 32], stock relation [4, 5] and many model architectures like LSTM [22], Attention LSTM [12, 24], Transformer [8] have been adopted and show promising gains.

Despite the success of these methods for stock prediction, they rely on the assumption that all stock samples follow an identical distribution. However, as the stock market data is a consequence of trading actions from a large number of participants, their diverse investing strategies will introduce multiple trading patterns that will be reflected in the data [27]. One such evidence is the coexistence of two contradictory phenomenons observed in stock market data: the momentum effect [2, 11] (the stocks that have higher returns in the past will continue to outperform others in the future) and the reversal effect [19] (the stocks that have lower returns in the past may have higher returns in the future). Neglecting multiple patterns in the training samples will inevitably undermine the performance of machine learning models, which has been widely observed and discussed in other related research fields [9, 15, 20, 25, 26]. Therefore, it is necessary to take into account the existence of multiple patterns when designing stock prediction solutions to pursue superior stock prediction performance.

In this work, we consider enhancing stock prediction by learning multiple stock trading patterns. Formally, we assume there are multiple distributions (patterns) in the stock market data $\mathbb{P} = \sum_k \nu_k \mathbb{P}_k$ with ν_k is the relative share of the k-th distribution, and all training and test samples come from one of these distributions[4]. In fact, if we have explicit pattern identifiers for both training and test samples, our problem can be effectively solved by Multi-Domain Learning [9, 15, 25, 26]. However, such identifiers don't pre-exist in stock market data and are hard to determine even for human experts. Even if it is possible to distinguish different patterns for model training by using ground-truth labels, it is not applicable for the test period in which labels will not be available. Therefore, the main challenge to learn multiple trading patterns in stock market data is *how to design a unified solution to discover and distinguish different patterns for both the training stage and the testing stage.*

To this end, we propose a new architecture, Temporal Routing Adaptor (TRA), as an extension module to empower existing stock prediction models with the ability to model multiple stock trading patterns. Essentially, TRA consists of a set of *predictors* to model different patterns and a *router* to predict which pattern a sample belongs to. The router leverages both the latent representation extracted from the backbone model (e.g., last hidden states from LSTM) as well as the temporal prediction errors of different predictors to determine a sample's pattern and assign to a specific predictor with a gating architecture [1, 13, 17, 28]. To further guarantee the discovery of diverse trading patterns, we formulate the optimal sample to predictor assignment problem as an Optimal Transport (OT) problem [7, 30] and use the optimized solution from OT to guide the learning of router through an auxiliary regularization loss term.

To demonstrate the practical value of the proposed framework, we conduct extensive experiments with real-world stock market data. Experiment results show that the proposed method can bring significant gains to existing stock prediction methods. Specifically, our method can improve the information coefficient (IC) from 0.053 to 0.059 when compared to Attention LSTM [12, 24], and from 0.051 to 0.056 when compared to Transformer [8]. Further investigations demonstrate the effectiveness of different components designed in the proposed framework.

The main contributions are summarized as follows:

- To the best of our knowledge, we are the first to consider designing stock prediction methods under the existence of multiple trading patterns and we also give thorough investigations of the existence and impact of multiple patterns.
- We propose Temporal Routing Adaptor (TRA) as an extension module to empower existing stock prediction methods with the capacity of modeling multiple stock trading patterns, and we design an effective learning algorithm based on optimal transport to further guarantee the discovery of multiple patterns.
- We conduct extensive experiments with real-world stock market data and compare with existing state-of-the-art baselines. Experiment results demonstrate the effectiveness of the proposed method.

2 RELATED WORK

Stock Prediction with Deep Neural Networks. Recently, there are many research studies to design deep learning solutions for stock prediction, which mainly fall under two categories, technical analysis (TA) and fundamental analysis (FA). TA methods focus on predicting the future movement of a stock price from market data (mainly price and volume). Among them, [33] proposes a variant of LSTM to discover multiple frequency patterns in stock market data, [24] designs a dual-stage attention-based LSTM for stock prediction, [12] further improves the attention LSTM by leveraging adversarial learning, and [8] also demonstrates Transformer can bring performance gains for mining long-term financial time series. FA methods are often design to capture more diversified alternative data and the most well studied data are news texts. [16] propose a Hybrid Attention Networks (HAN) to predict stock trend based on the sequence of recent related news, [32] designs a stochastic recurrent model (SRM) to address the adaptability of stock markets and [6] designs a framework to leverage knowledge graph to event-based stock prediction. However, all these methods follow the identical distribution assumption and neglect the existence of multiple patterns, which limit their capacity to achieve desirable performance.

Multi-Domain Learning. Multi-Domain Learning (MDL) is the field that addresses the learning challenges when the training data violates the *i.i.d.* assumption and contains multiple domains (patterns) [9, 15, 20, 25, 26]. In MDL, the *domain identifiers* are assumed

[3]Unless otherwise stated, in this paper we use i to denote the index of stock s at time t for simplicity.

[4]Note that if we consider collecting new test samples from an online environment, there may also be new emerging patterns. In this paper, we assume the model can be periodically updated to capture such new patterns and leave the online learning setting where the new pattern only appears during test as future work.

known during both training and test. The most straightforward approach to solve this challenge is separating the whole training set into several subsets according to their domain identifiers, and then training different models for different domains respectively. Recent studies in MDL often take the aforementioned method as the oracle baseline and mainly focus on designing more parameter-efficient network architectures, e.g., residual adaptors [15, 25, 26] or depth-wise separable convolution based gating [14]. However, there are no explicit domain identifiers in the stock market data and thus existing MDL solutions fall to address the challenges in this work.

3 MULTIPLE TRADING PATTERNS

In this section, we will give several empirical evidence of the existence of multiple stock trading patterns from both investment practice and model behavior.

In the real world, most investors will follow specific strategies to purchase and sell stocks. When a large number of individuals adopt the same investing strategy, it will become a pattern in stock market data. Figure 1 shows the annualized excess returns relative to the market of three most known investment strategies [11]: *size* (buy the small-cap stocks), *value* (buy the stocks with the highest book-to-market ratio), and *momentum* (buy the stocks that perform the best in the past 12 months). It can be observed that different strategies will take the lead in different periods (e.g., size before 2017, momentum since 2019), which implies there will be diverse trading patterns in the stock market data.

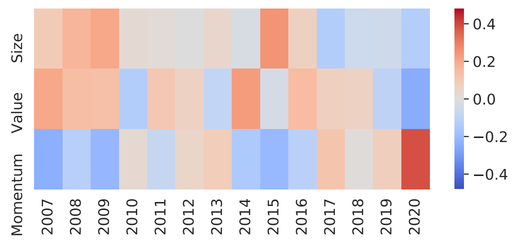

Figure 1: Annualized excess return relative to the market when equally investing in the top quintile stocks by using three investment strategies *size*, *value* and *momentum* in China stock market.

When there are multiple patterns in the data, a single model will be insufficient to capture all of them, especially when some of them are contrary to each other. In the below example, we learn multiple linear models for stock prediction with observations from different periods. We use the market capitalization rank (size), book-to-market ratio rank (value), and total return rank in the past 12 months (momentum) as features ($\mathbf{X}$), as well as the stock return ranks in the next month as the label ($\mathbf{y}$). The learned linear coefficients can be found in Figure 2 and they can be used to characterize the underlying data patterns. For example, the negative coefficient of momentum in 2009 tells us the pattern is that stock returns will be negatively correlated with their past performance. However, in 2013 the momentum coefficient becomes positive which indicates there is an opposite pattern. It will not be possible to model both these two patterns without introducing different sets of parameters to model these two patterns respectively.

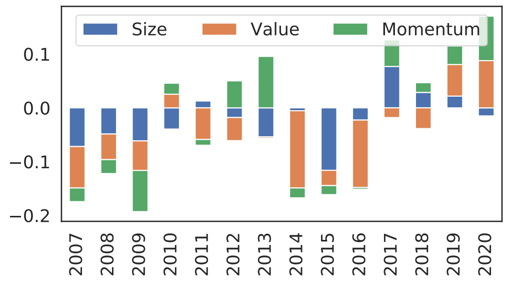

Figure 2: Linear coefficients on three features when learned with observations from different periods. Negative coefficient will be stacked below 0. The height of the bar represents the absolute value of the coefficient.

4 PROBLEM FORMULATION

Based on the empirical observations in Section 3, it becomes obvious that we should take into account multiple trading patterns when designing stock prediction methods. In this section, we will give a formal formulation of the learning task studied in this paper.

Let $\mathcal{X}$ denote the feature space and $\mathcal{Y}$ denote the label space. We assume data are sampled from some mixture of distributions $\mathbb{P} = \sum_k \nu_k \mathbb{P}_k$ on $\mathcal{X} \times \mathcal{Y}$, with subscript k to denote the index of the k-th pattern and ν_k denotes its relative share. Note that both $\mathbb{P}_k$ and ν_k are unknown in the studied task. In practice, we will have a dataset containing N observations $\mathcal{D} = \left\{(\mathbf{x}_i, \mathrm{y}_i)\right\}_{i=1}^{N}$ sampled from the mixture of distributions $\mathbb{P}$, where an observation $\mathbf{x}_i \in \mathcal{X}$ and $\mathrm{y}_i \in \mathcal{Y}$ denote the feature and label of i-th sample. For model training and inference, we assume ν_k has the same proportion between the training dataset $\mathcal{D}^{\text{train}}$ and test dataset $\mathcal{D}^{\text{test}}$. Let θ denote the parameters that we want to learn, then the goal of learning is

$$\min_{\theta} \mathbb{E}_{(\mathbf{x}_i, \mathrm{y}_i) \in \mathcal{D}^{\text{test}}} \ell(\mathbf{x}_i, \mathrm{y}_i; \theta), \tag{1}$$

where ℓ is some measurement function.

In order to optimize θ, a common practice in existing supervised learning settings is altering $\mathcal{D}^{\text{test}}$ with $\mathcal{D}^{\text{train}}$ in Equation 1 to optimize the parameter through empirical risk minimization. However, when there are multiple patterns in $\mathcal{D}^{\text{train}}$, the learned single model with such an objective often performs worse than the multiple models for different patterns [15, 25, 26]. Therefore, it's desired to identify different patterns and learn different sets of parameters for each pattern respectively in order to further enhance the stock prediction performance.

5 TEMPORAL ROUTING ADAPTOR

In this section, we first propose the architecture of *Temporal Routing Adaptor* (TRA) that can be used as an extension module to empower existing stock prediction models with the capacity to learn multiple patterns. Then we design a learning algorithm based on Optimal

Transport (OT) to further guarantee the discovery of multiple patterns. Last, we will give more implementation details for effectively training TRA-based models.

5.1 Architecture

In order to capture multiple stock trading patterns, the model needs to has the ability to provide different parameters for samples that belongs to different patterns. In this work, we propose Temporal Routing Adaptor (TRA) with two components to fulfill this goal: a set of *predictors* to learn different trading patterns, and a *router* to select the appropriate predictor. As illustrated in Figure 3, the *router* will assign samples with the same pattern to the same *predictor*, such that during the training stage the assigned predictor will capture the corresponding pattern from the assigned samples, and during the testing stage the best matched predictor will be used for inference.

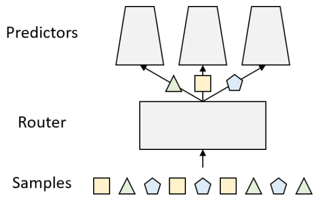

Figure 3: An overview of the proposed *predictors* + *router* framework. The *predictors* are responsible for learning different patterns while the *router* controls how samples are assigned to different predictors.

Let's first consider the design of the *predictors*. The most straightforward approach should be introducing multiple parallel models for different patterns, yet it will introduce $(K-1)\times$ more parameters compared to the original model if we want to model K different patterns. Using such many parameters will make the model more prone to over-fitting as observed in our experiments. Therefore, we only replace the output layer in the traditional stock prediction models with a set of linear predictors. This design can be easily plugged into existing stock prediction networks with negligible added parameters. We use Attention LSTM as the backbone feature extractor to demonstrate this in Figure 4. Another benefit of using this design is that if we keep only one predictor, then TRA will become the classical Attention LSTM and thus our framework won't cause any performance regression.

For the design of the *router*, it depends on what kind of information we want to use to predict the underlying pattern to which a sample belongs. In this work, we utilize two types of information to predict a sample's pattern and assign it to the appropriate predictor: 1) the latent representation from the backbone feature extractor, 2) temporal prediction errors of different predictors. The first type of information should be easy to understand: certain patterns can be directly distinguished from the latent representation $p(\mathbf{x})$. And instead of using raw features, we choose to use latent representation because it is more related to the underlying pattern $p(\mathrm{y}|\mathbf{x})$. Moreover, considering that the latent representation still cannot give enough information on $p(\mathrm{y}|\mathbf{x})$ as the patterns in stock market

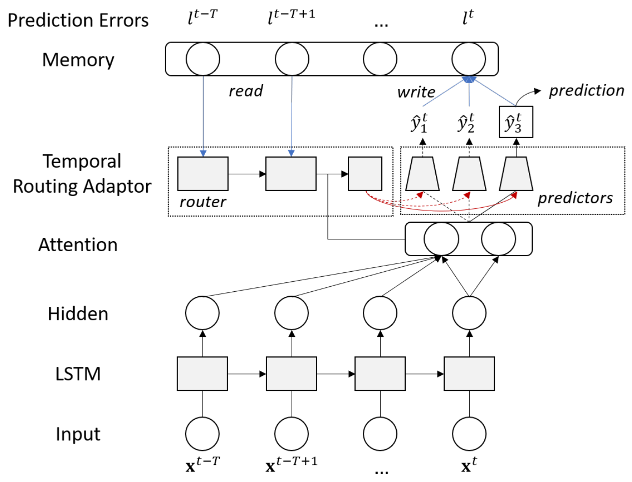

Figure 4: An example to use Temporal Routing Adaptor (TRA) as an extension module on Attention LSTM. The router uses both latent representation from Attention LSTM and temporal prediction errors stored in memory to select the best predictor for the current sample.

data are extremely hard to capture, we leverage the second type of information as a supplement.

The second type of information is inspired by the real-world investment practice: investors will periodically change their investment strategies (analogue to multiple predictors) from one to the other based on the recent performances (analogue to prediction errors) of these strategies. Merrill Lynch investment Clock[5] is such an example that shows certain investment strategies will take the lead following the macroeconomic cycle. Therefore, we can use the temporal prediction errors of different predictors in the past days to predict which predictor should be chosen in the next day for the current sample. To this end, we use a recurrent neural network to summarize the history performance of different predictors into an embedding vector and concatenate this embedding with the latent representation as the input of the router to predict the pattern.

Formally, let $\Theta = \{\theta_1, \theta_2, ..., \theta_K\}$ denote the parameters of K predictors, π denote the parameter of the router, and ψ denote the parameter of the shared feature extractor. For the k-th predictor, we denote the entire model (feature extractor and predictor) to give prediction for the i-th sample as $\hat{\mathrm{y}}_{ik} = \theta_k \circ \psi(\mathbf{x}_i)$. We also use bold symbol $\hat{\mathbf{y}}_i = \Theta \circ \psi(\mathbf{x}_i) \triangleq [\theta_1 \circ \psi(\mathbf{x}_1), \theta_2 \circ \psi(\mathbf{x}_2), ..., \theta_K \circ \psi(\mathbf{x}_K)]^\intercal$ to denote the predictions from all predictors. The prediction errors for the i-th sample can be then calculated as

$$l_i = \ell(\hat{\mathbf{y}}_i, \mathrm{y}), \tag{2}$$

where we use l_i to denote the vectorized prediction errors $[\ell(\hat{\mathrm{y}}_1, \mathrm{y}_1), \ell(\hat{\mathrm{y}}_2, \mathrm{y}_2), ..., \ell(\hat{\mathrm{y}}_K, \mathrm{y}_K)]^\intercal$ of the i-th sample for simplicity, and ℓ is some measurement function related to the stock prediction task.

Note that we can also index the i-th sample by the stock id s and the timestamp t, thus we can aggregate the temporal prediction

[5]https://macro-ops.com/the-investment-clock/

errors for this sample as

$$\mathbf{e}_i \triangleq \mathbf{e}_{i=(s,t)} = \left[l_{(s,t-T)}, l_{(s,t-T+1)}, ..., l_{(s,t-h)}\right]^\top, \quad (3)$$

where T is a maximized lookback window, h is an extra gap to avoid using future information[6]. To speed up model training, we also leverage *memory* to store the prediction errors. More details about the memory implementation can be found in Section 5.3.

Given both latent representation and temporal prediction errors, a straightforward approach for the router to give prediction is by applying a soft attention on different predictors through softmax. Denote the latent representation of sample i extracted from ψ as $\mathbf{h}_i = \psi(\mathbf{x}_i)$, then such softmax-based routing mechanism can be generated by

$$\begin{aligned} \mathbf{a}_i &= \pi(\mathbf{h}_i, \mathbf{e}_i), \\ \mathbf{q}_i &= \frac{\exp(\mathbf{a}_i)}{\text{sum}(\exp(\mathbf{a}_i))}, \\ \hat{\mathrm{p}}_i &= \mathbf{q}_i^\top \hat{\mathbf{y}}_i \end{aligned} \quad (4)$$

where $\mathbf{q}_i$ is the routing attention and $\hat{\mathrm{p}}_i$ is the final prediction of the entire model.

However, it's important for the router to give discrete selection in order to really distinguish different patterns. Instead of using the continuous softmax in Equation 4, the straight approach to achieve the discrete selection is using $argmax(\mathbf{q}_i)$, which however is not differentiable. To achieve differentiable but discrete routing, we utilize the gumbel-softmax trick [18] to enable the differentiability of the router. Essentially, it will add gumbel noise to the logits $\mathbf{a}_i$ and leverage the reparametrization trick to make the computation fully differentiable.

$$\mathbf{q}_i = \frac{\exp((\mathbf{a}_i + \epsilon)/\tau)}{\text{sum}(\exp((\mathbf{a}_i + \epsilon)/\tau))}, \quad (5)$$

where $\epsilon = [\epsilon_1, \epsilon_2, ..., \epsilon_K]$ are i.i.d. sampled drawn from Gumbel(0, 1)[7] distribution and τ is the temperature that controls the sharpness of the output distribution.

5.2 Optimal Transport

While TRA enables the capacity of modeling multiple trading patterns in a single network, it's still challenging to discover the underlying multiple trading patterns. In fact, we observed in our experiments that directly training TRA will end up with some trivial solutions, where all samples are routed to the same predictor and thus fails to capture the diverse patterns. A straightforward approach is to directly add strong regularization to force the router give relatively balanced sample assignment to different predictors. In this work, we propose a better approach to achieve such regularization in the meanwhile encourage the discovery of multiple trading patterns.

Our main idea is to *assign a sample to the predictor that has the smallest loss (objective) while keeping the number of assigned samples to a specific predictor proportional to the relative share of the corresponding pattern (constraint).* Figure 5 gives an illustration of our idea and we can further formalize it as an optimization problem. First we can pack the prediction errors of all samples × all predictors into a loss matrix $\mathbf{L}$, with $\mathrm{L}_{ik} = \ell(\theta_k \circ \psi(\mathbf{x}_i), \mathrm{y}_i)$ is

[6] h should be larger than the horizon of label to avoid using future information.

[7] The Gumbel(0,1) distribution can be sampled by first drawing uniform distribution $z \sim \text{Uniform}(0, 1)$ then applying $-\log(-\log(z))$.

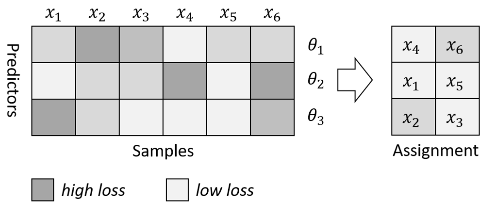

Figure 5: Our goal is assigning samples x_1 to x_6 to three predictors θ_1 to θ_3 with the goal of minimizing the prediction loss while keeping the number of assigned samples balanced.

the prediction loss of sample i when choosing predictor θ_k. Denote the sample assignment matrix for N samples and K predictors as $\mathbf{P} \in \{0, 1\}^{N \times K}$, then we are able to get the assignment $\mathbf{P}$ by solving the following optimal transport problem [7, 30]

$$\begin{aligned} &\min_{\mathbf{P}} \langle \mathbf{P}, \mathbf{L} \rangle, \\ &s.t. \sum_{i=1}^{N} \mathbf{P}_{ik} = v_k * N, \ \forall k = 1...K \\ &\quad \sum_{k=1}^{K} \mathbf{P}_{ik} = 1, \ \forall i = 1...N \\ &\quad \mathrm{P}_{ik} \in \{0, 1\}, \ \forall i = 1...N, k = 1...K, \end{aligned} \quad (6)$$

where $\langle \cdot, \cdot \rangle$ stands for the Frobenius dot-product and v_k is the relative share of the k-th pattern as introduced in Section 4. The first constraint ensures the assigned samples are proportional to the relative shares of the latent patterns, and the second to third constraints ensure the discrete choice of only one predictor. This problem is combinatorial and it will take a great amount of time to get the exact solution with the standard solvers. To overcome this challenge, we leverage the *Sinkhorn's matrix scaling algorithm* [7] to obtain an approximated solution with the benefit of several orders of magnitude speed-ups.

After obtaining the optimal sample assignment matrix $\mathbf{P}$, we further investigate how to use it to assist the training of TRA. Note that the optimized sample assignment $\mathbf{P}$ depends on the prediction loss $\mathbf{L}$ and thus also depends on the unknown label, it cannot be directly used to assign samples to different predictors during the testing stage as labels will not be available. Here we consider to use it to guide the learning of the router through adding an auxiliary regularization term into the objective

$$\min_{\Theta,\pi,\psi} \mathbb{E}_{(\mathbf{x}_i, \mathrm{y}_i) \in \mathcal{D}^{\text{train}}} \left[\ell(\mathbf{x}_i, \mathrm{y}_i; \Theta, \pi, \psi) - \lambda \textstyle\sum_{k=1}^{K} \mathrm{P}_{ik} \log(q_{ik})\right], \quad (7)$$

where q_{ik} is the predicted probability from gumbel-softmax in Equation 5, and λ is a hyper-parameter controlling the regularization strength. The regularization term is defined by cross entropy with $\mathbf{P}$ as the target distribution.

Note that we don't know the exact number of trading patterns K, so it is treated as a hyper-parameter and will be compared by setting different values in our experiments. Besides, the relative shares v_k for different patterns in Problem 6 are also unknown. We address this by first treating different patterns having equal shares (i.e., $v_k = \frac{1}{K}$) in the early stage of training, then we gradually decay λ with decay rate ρ during training so the router can discover more suitable pattern share in the later stage.

5.3 Implementation

As TRA needs temporal prediction errors to determine the assignment of a sample, the straightforward approach to implement this will require a all forward pass of all training samples in each single training step, which will make computation costs too high to bear. To this end, we introduce an external memory $\mathbf{M} \in \mathbb{R}^{N \times K}$ to cache the prediction errors for all N samples with K predictors calculated in previous training steps. Then we follow a two-stage procedure to update the memory: refresh the whole memory before each training epoch and refresh partial memory of mini-batch samples only appeared in the current training steps on the fly. By using this strategy we can achieve $N/\text{batchsize}$ speed-ups.

Algorithm 1 lists the whole process of training TRA, with the optimal transport regularized objective and the two-stage memory updating strategy. Essentially, each training epoch consists of: 1) inference all training samples and store the prediction errors of all predictors into the memory $\mathbf{M}$; 2) sample mini batches to update the model parameters with Equation 7 as well as update the memory for samples in this mini batch. Note that it is important to strictly follow the temporal order to sample batch samples during testing, because TRA needs prediction errors from former timestamps as input information for the router to select the right predictor for the current timestamp.

Algorithm 1: Algorithm for Training TRA.

```
   Input: training data $\mathcal{D}^{\text{train}}$, model parameters $\pi$, $\psi$, $\Theta$ (with
          size $K$), hyper-parameters $\lambda$, $\rho$
   Output: $\psi^*$, $\pi^*$, $\Theta^*$
 1 Randomly initialize $\pi$, $\psi$, $\Theta$;
 2 Initialize memory $\mathbf{M}$;
 3 while not meet the stopping criteria do
       /* step 1: update the memory.                */
 4     for $i \leftarrow 1$ to $N$ do
 5         Update the memory of the $i$-th sample by
             $\mathbf{M}_i = \ell(\Theta \circ \psi(\mathbf{x}_i), \mathbf{y}_i)$;
 6     end
       /* step 2: optimize model parameters.        */
 7     for $\mathcal{D}^{\text{batch}} \leftarrow \mathcal{D}^{\text{train}}$ do
 8         Decay regularization parameter $\lambda = \lambda * \rho$;
 9         Get the loss matrix $\mathbf{L}$ for current batch $\mathcal{D}^{\text{batch}}$;
10         Solve the optimal transport problem in Equation 6
             via Sinkhorn's algorithm;
11         Calculate the regularized loss with Equation 7 and
             update $\Theta$, $\pi$, $\psi$ via gradient descent algorithms;
12         Update memory $\mathbf{M}$ for samples in this batch;
13     end
14 end
```

6 EXPERIMENTS

To demonstrate the effectiveness of the proposed method, we further conduct extensive experiments on the real-world stock market data. In the following subsections, we will first introduce our experimental settings, including task and dataset description, baseline methods, evaluation metrics and hyper-parameter settings. Then we will give the experiment results of the compared methods. Last, we further conduct comprehensive experiments to demonstrate the effectiveness of the proposed method.

6.1 Settings

6.1.1 Task. In this work, we study the stock ranking prediction task with labels defined as the percentiles of cross-sectional stock returns at the end of next month. Using percentiles instead of raw returns as the learning targets has the following advantages: 1) the learning process will be less affected by extraordinary return outliers; 2) percentiles are uniform distributed and not affected by the market fluctuation (which will make the data non-stationary); 3) the model trained to predict cross-section stock return percentiles can be easily used by most portfolio construction strategies like long-short portfolio. As percentiles are continuous, the stock ranking prediction will be studies as a regression task in this work.

6.1.2 Dataset. We construct our dataset based on China stock market data[8]. Specifically, we collect publicly available data for CSI800 stocks[9] from baostock.com. Then we extract 16 features that have been proved informative in China stock market, including market capitalization, price-to-EPS, price-to-book value, price-to-sales, price-to-net cash flow, asset turnover ratio, net profit margin, receivables turnover ratio, EPS growth, asset growth, equity growth, 12 month momentum, 1 month reversal, close to 12 month highest, close to 12 month lowest, and maximum return in last month. All features are transformed into their ranks to match the stock ranking task within each trading day.

Finally, we follow the temporal order (avoid the data leakage problem) to split data into training (from 10/30/2007 to 05/27/2016), validation (from 09/26/2016 to 05/29/2018) and test (from 09/21/2018 to 06/30/2020). Note that the extra splitting gaps (e.g., 05/29/2018 to 09/21/2018) are intentionally introduced to avoid the leakage of both feature and label, because we will use the features in last 60 trading days for time series models and stock returns ranks in the next 21 trading days (1 month) as the learning targets.

6.1.3 Baselines. We compare our proposed method with the following baselines:

- **Linear** is a linear regression model.
- **LightGBM** [21] is non-linear model based on gradient boosted trees.
- **MLP** is a non-linear model based on neural networks.

[8]The China stock is less mature such that the publicly available information can still be used for stock prediction. For the developed market, the public information are no longer enough to predict the market and the proprietary data sources are demanded in order to learn stock prediction models, which however are hard to obtain for the research community.
[9]CSI800 consists of the 800 largest and most liquid China stocks. More information can be found at http://www.csindex.com.cn/en/indices/index-detail/000906.

- **SFM** [33] is a redesigned recurrent neural network by decomposing the hidden states of LSTM into multiple frequency components to model multi-frequency trading patterns.
- **ALSTM** [12, 24] adds an external attention layer into the LSTM architecture to attentively aggregate information from all hidden states in previous timestamps.
- **Trans.** [8] adopts the Transformer [29] architecture that has been used by [8] for stock prediction.

In addition to the above baselines, we also consider the following heuristic approach introduced in Section 3 to learn multiple trading patterns

- **ALSTM+TS/Trans.+TS** simply assigns samples to different predictors according to which time periods they belong to. As test samples belong to a new time period and thus cannot be assigned to the learned models, we simply average predictions from all the learned models.

6.1.4 Evaluation Metrics. In addition to evaluating the prediction errors of stock ranking by regression metrics like Mean Square Error (MSE) and Mean Absolute Error (MAE), the dominate ranking metric used in finance is the information coefficients (IC), which can be calculated as

$$\mathrm{IC} = \frac{1}{N} \frac{(\hat{\mathbf{y}} - mean(\hat{\mathbf{y}}))^\intercal (\mathbf{y} - mean(\mathbf{y}))}{std(\hat{\mathbf{y}}) * std(\mathbf{y})}, \tag{8}$$

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are the predicted rankings and actual rankings respectively. In practice, IC is calculated for different trading days and their average will be reported. To show the stability of IC, we also report the information ratio of IC, i.e., ICIR, which is calculated by dividing the average by the standard deviation of IC.

In addition, we also evaluate the ranking performance of the learned model by constructing a long-short portfolio by longing stocks of the top decile and shorting stocks of the bottom decile[10]. We will measure the portfolio's performance by annualized return (AR), annualized volatility (AV), annualized Sharpe Ratio (SR) and maximum drawdown (MDD). The details for portfolio construction and metric definitions can be found in in Appendix A.

6.1.5 Parameter Settings. To compare the performance of different baselines on our dataset, we first search architecture parameters and hyper-parameters for **LightGBM**, **MLP**, **SFM**, **ALSTM** and **Trans.** and select the best parameter based on the performance of validation set. The architecture parameters and hyper-parameter configurations for the search space can be found in Appendix B. For our methods, we use the same architecture parameter as the backbone models **ALSTM** and **Trans.** for fair comparison and only search the learning rate and decay rate ρ for regularization strength λ used in Equation 7.

6.2 Main Results

Table 1 summarized the evaluation performances of all compared methods. From the exhibited results, we have the following observations:

- The proposed method can be used to further improve the performance of previous state-of-the-art baselines (**ALSTM** and **Trans.**). Specifically, compared to **ALSTM**, our method **ALSTM+TRA** can reduce the prediction errors measured by MSE from 0.158 to 0.157 and improve the ranking metric measured by IC from 0.053 to 0.059. Also compared to **Trans.**, our method **Trans.+TRA** can also improve IC from 0.051 to 0.056. These results demonstrate the efficacy of the proposed method to enhance stock prediction.
- The two heuristic methods **ALSTM+TS** and **Trans.+TS** that model multiple patterns by splitting the dataset into multiple periods even harm the model's performance. We postulate the reason is that the learned individual models can only capture the pattern of a specific period and simply averaging predictions from these models during testing will mixes up different patterns and thus performs poorly. This shows that selecting the right model for inference is also critical to ensure the performance. As a comparison, the router in the proposed TRA framework has the ability to fulfill this goal.
- From the portfolio metrics, it can be observed that our method can also achieve desirable performance. Compared to **Trans.**, our method achieves higher annualized return (ours 16.1% vs **Trans.** 14.5%) as well as higher Sharpe ratio (ours 1.133 vs **Trans.** 1.028). These results further demonstrate that the proposed method can be used in real-world investment to achieve higher risk-return rewards.

[10]In China stock market, shorting stocks can be achieved by securities margin trading.

6.3 Incremental Analysis

In this section, we want to answer the following research questions through incremental analysis:

- **RQ1** Can TRA learn multiple trading patterns and select the correct one for prediction?
- **RQ2** Are both latent representation and temporal prediction errors useful for the router?
- **RQ3** How does optimal transport help the training of TRA?
- **RQ4** Is TRA sensitive to the number of K predictors used?

6.3.1 RQ1. We first analyse the learned multiple predictors with $K = 3$ by showing their prediction loss of a randomly selected stock in Figure 6. It can be observed that the learned predictors have different performances at the same time, which demonstrates TRA does capture the diverse trading patterns. We also plot the selections given by the router in the right side and it can be observed that the router successfully picks the right model for prediction (the one with the lowest prediction loss). This evidence further justify the capability of TRA to select the correct model for inference.

6.3.2 RQ2. To understand what types of information can be used by the router to make good assignments of different samples, we compare the following information sources with ALSTM as the backbone model

- **Random** simply uses Gaussian noise as the input information for the router.
- **Latent Representations, LR** are the extracted latent representation from the backbone feature extractor.
- **Temporal Prediction Errors, TPE** are prediction errors of different predictors defined in Equation 3.
- **LR+TPE** combines **LR** and **TPE** as the input for the router.

Table 1: Stock ranking and portfolio performance of the compared methods. For MSE and MAE, we also report the standard deviation of 5 random seeds as $(\cdot)$. $\uparrow$ means the larger the better while $\downarrow$ means the smaller the better.

Method	Ranking Metrics				Portfolio Metrics			
	MSE ($\downarrow$)	MAE ($\downarrow$)	IC ($\uparrow$)	ICIR ($\uparrow$)	AR ($\uparrow$)	AV ($\downarrow$)	SR ($\uparrow$)	MDD ($\downarrow$)
Linear	0.163	0.327	0.020	0.132	-3.2%	16.8%	-0.191	32.1%
LightGBM	0.160(0.000)	0.323(0.000)	0.041	0.292	7.8%	15.5%	0.503	25.7%
MLP	0.160 (0.002)	0.323 (0.003)	0.037	0.273	3.7%	15.3%	0.264	26.2%
SFM	0.159 (0.001)	0.321 (0.001)	0.047	0.381	7.1%	14.3%	0.497	22.9%
ALSTM	0.158 (0.001)	0.320 (0.001)	0.053	0.419	12.3%	**13.7%**	0.897	**20.2%**
Trans.	0.158 (0.001)	0.322 (0.001)	0.051	0.400	14.5%	14.2%	1.028	22.5%
ALSTM+TS	0.160 (0.002)	0.321 (0.002)	0.039	0.291	6.7%	14.6%	0.480	22.3%
Trans.+TS	0.160 (0.004)	0.324 (0.005)	0.037	0.278	10.4%	14.7%	0.722	23.7%
ALSTM+TRA (Ours)	**0.157 (0.000)**	**0.318 (0.000)**	**0.059**	**0.460**	12.4%	14.0%	0.885	20.4%
Trans.+TRA (Ours)	**0.157 (0.000)**	0.320 (0.000)	0.056	0.442	**16.1%**	14.2%	**1.133**	23.1%

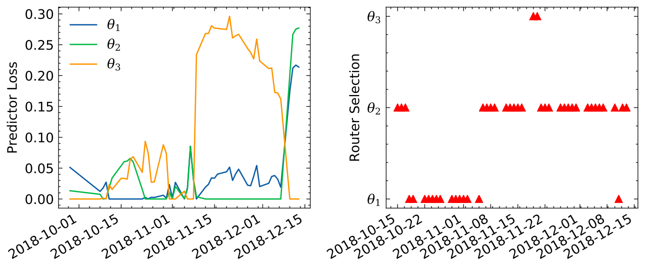

Figure 6: Loss of different predictors (left) and the router's selection (right). Loss is normalized by subtracting the minimum value of three.

The model performances when using the above different types of information are summarized in Table 2. We can see that compared to **Random**, both **LR** and **TPE** consistently improve TRA's performance and the best performance can be achieved when both **LR** and **TPE** are used in TRA.

Table 2: TRA's performance when used with different information sources.

Information	MSE ($\downarrow$)	MAE ($\downarrow$)	IC ($\uparrow$)	ICIR ($\uparrow$)
Random	0.159 (0.001)	0.321 (0.002)	0.048	0.362
LR	0.158 (0.001)	0.320 (0.001)	0.053	0.409
TPE	0.158 (0.001)	0.321 (0.001)	0.049	0.381
LR+TPE	**0.157 (0.000)**	**0.318 (0.000)**	**0.059**	**0.460**

6.3.3 RQ3. We now investigate the role of optimal transport (OT) used in our method. In our framework, we formalize the quest of optimal sample to predictor assignment task as an OT problem in Equation 6, and use the solution from OT to guide the learning of the router through a regularization term in Equation 7. We demonstrate the value of OT by comparing it with the naive objective without the regularization term. As illustrated in Figure 7, after removing the OT regularization, the router tends to give some trivial assignment, where almost all samples are assigned to the same predictor and the remaining predictors are left unused. With OT, the assigned samples are more balanced thus all predictors can be well trained to capture the diverse trading patterns.

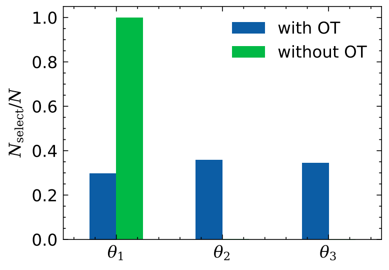

Figure 7: The relative shares of samples assigned to different predictors trained with or without optimal transport (OT).

6.3.4 RQ4. Recall that TRA relies on a set of K predictors to model multiple patterns where K represents the number of latent patterns. As shown in Figure 8, we compare TRA's performance measured by both IC and MSE when using different Ks. It can be observed that compared to $K = 1$ (i.e., only use the backbone model), introducing more predictors will consistently lead to better model performance. Specifically, adding four predictors ($K = 5$) can bring huge performance gains, while adding more predictors can achieve slightly better performance. This suggests that a moderate selection of K should give desirable performance.

7 CONCLUSION

In this paper, we show how the stock prediction performance can be further improved by learning multiple trading patterns in the stock market data. Specifically, We propose Temporal Routing Adaptor (TRA), which is a lightweight extension module that can be used to empower existing stock prediction networks with the ability of modeling multiple trading patterns. In order to encourage the discovery of the diverse trading patterns, we solve the sample to predictor assignment task as an Optimal Transport (OT) problem and use the solution from OT to guide the learning of the router. Experiments results demonstrate the effectiveness of the proposed

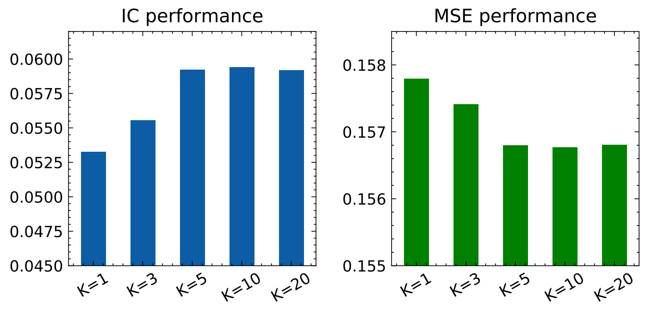

Figure 8: TRA's performance when using different number of predictors.

method for learning multiple patterns and enhancing the stock prediction performance. Future work will take into account the new patterns emerging in the online environment.

REFERENCES

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. 2020. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3931–3940.

[2] Clifford S Asness, Tobias J Moskowitz, and Lasse Heje Pedersen. 2013. Value and momentum everywhere. *The Journal of Finance* 68, 3 (2013), 929–985.

[3] Lakshay Chauhan, John Alberg, and Zachary Lipton. 2020. Uncertainty-Aware Lookahead Factor Models for Quantitative Investing. In *International Conference on Machine Learning*. PMLR, 1489–1499.

[4] Chi Chen, Li Zhao, Jiang Bian, Chunxiao Xing, and Tie-Yan Liu. 2019. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2376–2384.

[5] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. 2018. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1655–1658.

[6] Dawei Cheng, Fangzhou Yang, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. 2020. Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2221–2230.

[7] Marco Cuturi. 2013. Sinkhorn distances: lightspeed computation of optimal transport.. In *NIPS*, Vol. 2. 4.

[8] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. 2020. Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 4640–4646.

[9] Mark Dredze, Alex Kulesza, and Koby Crammer. 2010. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning* 79, 1-2 (2010), 123–149.

[10] Eugene F Fama. 1965. The behavior of stock-market prices. *The journal of Business* 38, 1 (1965), 34–105.

[11] Eugene F Fama and Kenneth R French. 2012. Size, value, and momentum in international stock returns. *Journal of financial economics* 105, 3 (2012), 457–472.

[12] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. 2019. Enhancing Stock Movement Prediction with Adversarial Training. *IJCAI* (2019).

[13] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. 2020. Learning to branch for multi-task learning. In *International Conference on Machine Learning*. PMLR, 3854–3863.

[14] Yunhui Guo, Yandong Li, Liqiang Wang, and Tajana Rosing. 2019. Depthwise Convolution Is All You Need for Learning Multiple Visual Domains. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), 8368–8375. https://doi.org/10.1609/aaai.v33i01.33018368 arXiv:1902.00927

[15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[16] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 261–269.

[17] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G. Edward Suh. 2019. Channel Gating Neural Networks. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., 1886–1896.

[18] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[19] Narasimhan Jegadeesh. 1990. Evidence of predictable behavior of security returns. *The Journal of finance* 45, 3 (1990), 881–898.

[20] Mahesh Joshi, Mark Dredze, William Cohen, and Carolyn Rose. 2012. Multi-domain learning: when do domains matter?. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 1302–1312.

[21] 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. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[22] David MQ Nelson, Adriano CM Pereira, and Renato A de Oliveira. 2017. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1419–1426.

[23] James M Poterba and Lawrence H Summers. 1988. Mean reversion in stock prices: Evidence and implications. *Journal of financial economics* 22, 1 (1988), 27–59.

[24] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971* (2017).

[25] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 506–516.

[26] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8119–8127.

[27] Jay R Ritter. 2003. Behavioral finance. *Pacific-Basin finance journal* 11, 4 (2003), 429–437.

[28] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[30] Cédric Villani. 2008. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media.

[31] Heyuan Wang, Tengjiao Wang, and Yi Li. 2020. Incorporating Expert-Based Investment Opinion Signals in Stock Prediction: A Deep Learning Framework. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (2020).

[32] Yumo Xu and Shay B Cohen. 2018. 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)*. 1970–1979.

[33] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2141–2149.

A PORTFOLIO METRICS

In this paper we consider constructing a long-short portfolio to evaluate the predicted stock ranks from different models. Denote the selected stocks for the target portfolio on day d as Λ_d^+ and Λ_d^- for the long side and short side respectively, and denote the realized return for stock s at day d as $r_{s,d}$, where $r_{s,d} = 0$ if d is not a trade day (e.g., weekends). After carrying out trading simulations day by day, we will have a sequence of portfolio returns $\mathbf{R} = [R_1, R_2, ..., R_D]$ for D days in the evaluation period, with each day's portfolio return R_d calculated by

$$R_d = \frac{1}{|\Lambda_d^+|} \sum_{s \in \Lambda_d^+} r_{s,d} - \frac{1}{|\Lambda_d^-|} \sum_{s \in \Lambda_d^-} r_{s,d}, \quad \forall d = 1, 2, ..., D. \tag{9}$$

Based on the portfolio returns sequence $\mathbf{R}$, there are quite a few metrics concerning different dimensions for how well this dynamic portfolio behaves. Among them, four metrics are used in this paper to evaluate the portfolio performance

- **Annualized Return, AR.** Annualized Return measures the total profits generated by following the model predictions for investment on a yearly basis. It's calculated by scaling daily average portfolio return to the total number of calendar days in one year
$$\text{AR} = mean(\mathbf{R}) \times 365 \times 100\%. \tag{10}$$
- **Annualized Volatility, AV.** Annualized Volatility measures the risk of the portfolio on a yearly basis. It's calculated by first calculating the standard deviation of daily portfolio returns, then scaling the deviation by the square root of the total number of calendar days in one year
$$\text{AV} = std(\mathbf{R}) \times \sqrt{365} \times 100\%. \tag{11}$$
- **Sharpe Ratio, SR.** Sharpe Ratio is a measure of risk-adjusted return. It describes how much excess return we can get for the volatility risk we bear. SR can be calculated as the ratio of AR and AV
$$\text{SR} = \frac{AR}{AV}. \tag{12}$$
- **Maximum Drawdown, MDD.** Maximum drawdown measures the worst performance of the portfolio in a given period. It is calculated as the maximum observed loss from a peak to a trough of a portfolio
$$\text{MDD} = \max_{d \in [1,D]} \left\{ \max_{t \in [1,d]} \sum_{\le t} R_t - \sum_{\le d} R_d \right\} \tag{13}$$

B MODEL ARCHITECTURE AND HYPER-PARAMETERS

Here are the detailed search space for model architecture and hyper-parameters:

- **LightGBM**:
  - *number of estimators*: 500
  - *number of leaves*: [32, 64, 128, 256]
  - *learning rate*: [0.1, 0.05, 0.01]
- **MLP/SFM/ALSTM**:
  - *number of layers*: [1, 2, 4]
  - *hidden size*: [32, 64, 128]
  - *dropout*: [0.0, 0.1, 0.2, 0.5]
  - *learning rate*: [0.001, 0.0005, 0.0002, 0.0001]
  - *batch size*: [512, 1024, 4096]
- **Transformer**:
  - *number of attention heads*: [2, 4, 8]
  - other parameters are the same as MLP etc.