Mastering Stock Markets with Efficient Mixture of Diversified Trading Experts

Shuo Sun
Nanyang Technological University
Singapore
shuo003@e.ntu.edu.sg

Xinrun Wang[*]
Nanyang Technological University
Singapore
xinrun.wang@ntu.edu.sg

Wanqi Xue
Nanyang Technological University
Singapore
wanqi001@e.ntu.edu.sg

Xiaoxuan Lou
Nanyang Technological University
Singapore
xiaoxuan001@e.ntu.edu.sg

Bo An[*]
Nanyang Technological University
Singapore
boan@ntu.edu.sg

ABSTRACT

Quantitative stock investment is a fundamental financial task that highly relies on accurate prediction of market status and profitable investment decision making. Despite recent advances in deep learning (DL) have shown stellar performance on capturing trading opportunities in the stochastic stock market, the performance of existing DL methods is unstable with sensitivity to network initialization and hyperparameter selection. One major limitation of existing works is that investment decisions are made based on one individual neural network predictor with high uncertainty, which is inconsistent with the workflow in real-world trading firms. To tackle this limitation, we propose AlphaMix, a novel three-stage mixture-of-experts (MoE) framework for quantitative investment to mimic the efficient bottom-up hierarchical trading strategy design workflow of successful trading companies. In Stage one, we introduce an efficient ensemble learning method, whose computational and memory costs are significantly lower comparing to traditional ensemble methods, to train multiple groups of trading experts with personalised market understanding and trading styles. In Stage two, we collect diversified investment suggestions through building a pool of trading experts utilizing hyperparameter level and initialization level diversity of neural networks for post hoc ensemble construction. In Stage three, we design three different mechanisms, namely as-needed router, with-replacement selection and integrated expert soup, to dynamically pick experts from the expert pool, which takes the responsibility of a portfolio manager. Through extensive experiments on US and Chinese stock markets, we demonstrate that AlphaMix significantly outperforms many state-of-the-art baselines in terms of 7 popular financial criteria.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Machine learning**; • **Applied computing → Electronic commerce**.

[*] Corresponding authors.

KDD '23, August 6–10, 2023, Long Beach, CA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599424

KEYWORDS

Quantitative investment; computational finance; stock prediction; ensemble learning; mixture-of-experts; deep learning

ACM Reference Format:
Shuo Sun, Xinrun Wang, Wanqi Xue, Xiaoxuan Lou, and Bo An. 2023. Mastering Stock Markets with Efficient Mixture of Diversified Trading Experts. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599424

1 INTRODUCTION

The stock market, a financial ecosystem involving over $90 trillion[1] market capitalization globally in 2020, is one of the most popular channels for investors to pursue desirable financial assets and achieve investment goals. As the famous efficient market hypothesis [10] claims, forecasting stock prices accurately and making highly profitable investment decisions are extremely challenging tasks due to the noisy nature of the financial markets. In the last decade, deep learning (DL) methods have become an appealing direction for stock prediction [51] and quantitative investment [3] owing to its ability to learn insightful market representations in an end-to-end manner. Most existing DL methods adopt various advanced network architectures (e.g., LSTM [28], Attention [34] and Transformer [6]) to extract meaningful features from heterogeneous financial data such as fundamental factors [4], economics news [17], social media [51] and investment behaviors [5].

To apply deep learning models for quantitative investment, the vast majority of existing methods [11, 24, 50, 54] first adopt stock prediction as a supervised learning task. Observable market information (e.g., historical price) is used as the feature vector and the future price movement direction [54] or return rate [50] is applied as the target for classification or regression, respectively. One neural network (NN) predictor is optimized to capture the underlying pattern of financial markets by minimizing the corresponding loss function. Later on, the trained predictor is applied on new test sets for prediction. Finally, investment strategies are constructed by picking stocks with the top predicted price rising probability [54] or highest predicted return rate [50]. However, the performance of previous methods is unstable and sensitive to many factors such

[1] https://data.worldbank.org/indicator/CM.MKT.LCAP.CD/

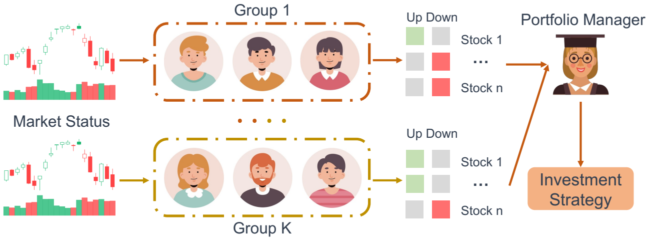

Figure 1: Overview of the bottom-up hierarchical trading strategy design workflow in real-world trading companies.

as random seeds [1]. The potential reason of this phenomenon is that existing methods make investment decisions based on only one individual predictor, whose predictions are usually uncertain [14] and fail to seize fleeting trading opportunities in stock markets. Demonstrative experiments are shown in Section 6.5.

In Figure 1, we plot an overview of trading strategy design workflow in real-world trading firms. There are multiple groups of trading experts with diversified background knowledge (e.g, finance and computer science). Within one group, they collaboratively work together to analyze the market from the same perspective. At the same time, groups work independently with different trading styles and risk tolerance to avoid mutual impact and correlated decisions. Then, a senior portfolio manager summarizes their results and makes the final investment decisions. This bottom-up hierarchical workflow plays a key role in designing robust and profitable trading strategies [8]. Inspired by it, we propose AlphaMix, a novel three-stage mixture-of-experts (MoE) framework for quantitative investment. In Stage one, an efficient ensemble learning method, whose computational and memory costs are significantly lower than traditional ensemble methods, is proposed to train multiple trading experts with personalised market understanding and trading styles. In Stage two, we build a pool of diversified trading experts by utilizing both hyperparameter level and initialization level diversity of neural networks for post hoc ensemble construction. In Stage three, we design three different mechanisms, namely with-replacement selection, integrated expert soup and as-needed router, to dynamically pick experts from the expert pool to take the responsibility of a portfolio manager. As illustrated in Figure 2, AlphaMix achieves the best performance with much smaller model size comparing to three prevalent ensemble methods [22, 45, 52]. The main contributions of this work are two-fold:

- To the best of our knowledge, AlphaMix is the first mixture-of-experts framework for quantitative investment. In AlphaMix, we first efficiently optimize multiple independent groups of trading experts, then we build a pool of diversified models for post hoc ensemble construction, and finally three different mechanisms are introduced to further improve performance by dynamically picking these experts.
- Through experiments on real-world data of two influential stock markets spanning over 5 years, we show that AlphaMix significantly outperforms 11 state-of-the-art baseline methods in terms of 7 financial criteria and demonstrate AlphaMix's practical applicability to quantitative investment with a series of exploratory and ablative studies.

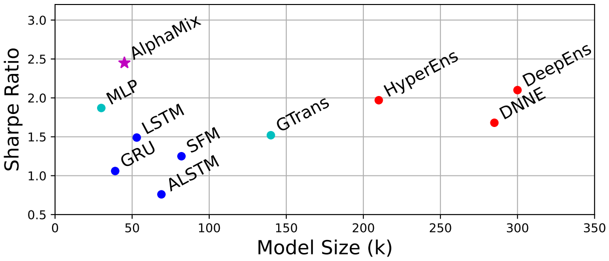

Figure 2: Model size vs. Sharpe ratio dot plot on US stock market. Ensemble methods (red) generally outperform other RNN (dark blue) and Non-RNN (light blue) methods. Our AlphaMix (purple) performs the best (17% improvement) with tiny computation overhead than the base model (MLP).

2 RELATED WORK

2.1 Financial Prediction with Deep Learning

Extensive deep learning methods have been proposed for stock prediction, which can be generally categorised into: 1) recurrent neural networks (RNN), 2) non-recurrent neural networks (NRNN), and 3) models with alternative data sources.

RNN models. RNN-based models are popular for stock prediction since they are specifically designed to capture temporal patterns in sequential data. Nelson et al. [28] show that GRU outperforms many baseline methods. Wang et al. [40] combine attention mechanism with LSTM to model correlated time steps. To improve the robustness of LSTM, Feng et al. [11] apply adversarial training techniques for stock prediction. Zhang et al. [55] propose a State Frequency Memory (SFM) recurrent network with Discrete Fourier Transform (DFT) to discover multi-frequency patterns in stock markets.

NRNN models. There are also many DL approaches that make stock prediction without using RNN-based models. Liu et al. [25] introduce a multi-scale two-way neural network (NN) to predict the stock trend. Ding et al. [6] propose a hierarchical Gaussian transformer-based model for stock movement prediction. Another direction of work employs graph-based DL models to model pairwise relations between stocks. For instance, Feng et al. [12] enhance graph convolutional networks (GCNs) with temporal convolutions for mining inter-stock relations. Sawhney et al. [36] focus on stock industry data and links between company CEOs.

Models with alternative data. To further show the potential of DL methods, the usage of additional data sources is another direction for trading signal discovery. Xu and Cohen [51] propose a variational autoencoder architecture to extract latent information from tweets. Chen et al. [5] enhance trading strategy design with the investment behaviors of professional fund managers. Other data sources such as economics news [17] and earning calls [37] are also used to improve the prediction performance.

However, this line of work focuses on designing one powerful DL model to extract meaningful market features, which usually involve millions of parameters and are hard to train in practice. Our AlphaMix proposes a new direction to make investment decisions with multiple models and significantly outperforms existing SOTA methods with simple multi-layer perceptrons (MLP) backbones.

2.2 Ensemble Learning

Ensemble methods, which combine the output of different models to improve the overall generalization performance, have been studied for decades [15, 16, 21, 31, 43, 45, 46]. DeepEns [22] shows its potential for uncertainty estimation. Although ensemble methods achieve stellar performance when each member makes independent errors [23], the significant increase of computation and memory cost becomes a major bottleneck for the wide deployment of ensemble methods. In recent years, the major direction in ensemble research is how to reduce the cost of training, testing and memory. Snapshot ensemble [18], in which a single model is trained by cyclic learning rates, is proposed to encourage visiting multiple local minima within one run. Wen et al. [43] propose an efficient mini-batch friendly ensemble method called BatchEns with an ensemble weight generation mechanism using Hadamard product. Recent works highlight the benefit of ensemble models with hyperparameter diversity [45] and architecture diversity [15], respectively. Instead of combining the outputs of neural networks, model soup [47] directly average the weights of neural networks and achieves great performance in pretrain settings with no extra inference cost.

Although there is a lot of progress in ensemble learning, most existing ensemble methods for stock markets [30] are only straightforward applications of traditional methods such as bagging and stacking with linear increase of both computation and memory cost. For instance, Xiang and Fu [48] combine different neural networks as sub-models with bagging for stock market prediction. There is still a lack of advanced efficient ensemble learning methods for capturing the fleeting trading opportunities in financial markets. AlphaMix is designed by customizing advanced ensemble techniques [2, 41, 43, 45, 47] into this filed to fill the gap.

2.3 Mixture of Experts

Ever since its introduction more than two decades ago [19], the mixture-of-experts (MoE) approach has been the key component for many works. Eigen et al. [9] extend MoE into deep learning by stacking two layers of mixture of experts with a trainable weighted gating network. Shazeer et al. [38] present a sparsely-gated MoE layer to enable training extremely large number of experts. Ma et al. [26] propose a multi-gate MoE framework to model task relationships in multi-task learning. Xu et al. [49] propose an adaptive Master-Slave regularized model for unexpected revenue prediction enhanced with alternative data. MoE models have achieved stellar performance in many challenging fields such as computer vision [35, 41, 42] and natural language processing [7, 38].

However, even though MoE methods have been successful in many fields, there is a lack of MoE frameworks for quantitative investment in the stochastic financial market. AlphaMix is the first MoE framework to fill this gap with three stages to mimic the efficient hierarchical bottom-up trading strategy design workflow in real-world trading firms.

3 GENERATING DIVERSIFIED EXPERTS

In this section, we first provide a formulation of quantitative investment. We then introduce an efficient ensemble approach to train multiple groups of trading experts with tiny computation and memory overhead. Finally, we conduct model augmentation to build a pool of diversified trading experts, which utilizes two sources of diversity: varied hyperparameters and random initialization.

3.1 Problem Formulation

We formulate quantitative investment as a supervised learning task by generating trading strategies based on stock movement predictions following [54]. Consider a stock pool of size N, where for each stock i on trading day t, there is a corresponding closing price p_t^i and a feature vector x_t^i.

Formalizing stock movement prediction. We define stock movement prediction with label $y^i_{[t,t+\tau]}$ over a period of τ days as a classification task following [51, 54]. Here, we define the label using the closing price of a given stock p_t^i, that can either rise or fall on day $t+\tau$ compared to day t as:

$$y^i_{[t,t+\tau]} = \begin{cases} 1, & p^i_{t+\tau} > p^i_t \\ 0, & p^i_{t+\tau} \leq p^i_t \end{cases} \tag{1}$$

Unless otherwise stated, we use x and y to denote the sequential feature vectors $(x^i_{t-k}, ..., x^i_t)$ and next day stock movement $y^i_{[t,t+1]}$ of stock i on time t in the following of this work for simplicity.

Trading strategy generation. To evaluate the performance of quantitative investment methods, we apply a widely used simple yet robust trading strategy called daily top k buy & hold as in [50, 54]. Specifically, we rebalance our portfolio by evenly allocating money into the top k stocks with the highest probability to increase from the stock pool at the end of each trading day.

3.2 An Efficient Ensemble Approach

To fully utilize the potential of ensemble learning methods in financial markets, there are two major challenges: i) how to train diversified ensemble members by taking personal market status understanding and trading styles into consideration; ii) how to address the significant time and memory cost of traditional ensemble methods for real-world deployment. To address these two challenges, we introduce an efficient ensemble approach to train a group of trading experts simultaneously with limited cost overhead leveraging the idea of [43]. As shown in Figure 3, we first allocate each ensemble member two trainable vectors representing personal market understanding and trading style. Then, we generate an independent rank-one matrix of each ensemble member by multiplying the two personalised vectors. Finally, the ensemble weights are generated by calculating the Hadamard product of the shared weight matrix and the independent rank-one matrix.

Formally, let W be the weights in a neural network layer. Denote the input dimension as m and the output dimension as n, i.e., $W \in \mathbb{R}^{m\times n}$. M trading experts are trained, where each expert has weight matrix $\overline{W_i}$. Specifically, each expert i owns a tuple of trainable vectors r_i and s_i with dimension m and n, which represents the expert's personal market understanding and trading style, respectively. Our algorithm generates a group of ensemble weights $\overline{W_i}$ as follows:

$$\overline{W_i} = W \circ F_i \,, \quad where\ F_i = r_i s_i^T \tag{2}$$

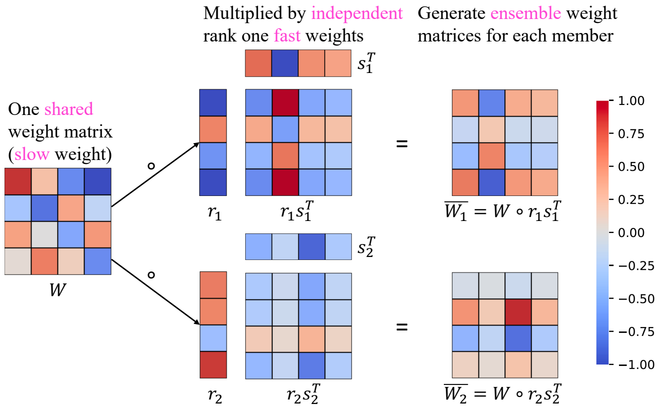
Multiplied by independent rank one fast weights
Generate ensemble weight matrices for each member
One shared weight matrix (slow weight)
s_1^T
s_2^T
∘
∘
=
=
W
r_1
$r_1 s_1^T$
$\overline{W_1} = W \circ r_1 s_1^T$
r_2
$r_2 s_2^T$
$\overline{W_2} = W \circ r_2 s_2^T$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Figure 3: Illustration of the efficient ensemble approach on how to generate the weights for two ensemble members.

For each training sample in the mini-batch, it receives an ensemble weights $\overline{W_i}$ by computing Hadamard product of W (shared slow weights) and a rank-one matrix F_i (personal fast weights). The subscript i represent expert i. The insight here is that one can modulate intermediate features of neural networks to achieve promising performance [32, 43] instead of modulating the whole weight matrices.

Vectorization for Parallelization. To further elaborate the efficiency of this approach, we show that the above ensemble weight generation mechanism can be parallelizable within one device based on [43]. Specifically, one computes a forward pass with respect to multiple ensemble members in parallel by manipulating the matrix computations for a mini-batch [44]. We denote a as the activations of the incoming neurons in a neural network layer. The next layer's activations c are calculated as:

$$\begin{aligned} c_n &= \phi\left(\overline{W}_i^T a_n\right) = \phi\left(\left(W \circ r_i s_i^T\right)^T a_n\right) \\ &= \phi\left(\left(W^T (a_n \circ r_i)\right) \circ s_i\right) \end{aligned} \tag{3}$$

where ϕ represents the activation function and the subscript n denotes the index in the mini-batch. The output is next layer's activations from the i^{th} ensemble member. In order to vectorize these computations, we define matrices R and S whose rows consist of the vectors r_i and s_i for all data samples in the mini-batch. The vectorized form of the above equation is as follows:

$$C = \phi(((A \circ R)W) \circ S) \tag{4}$$

where A and C are the mini-batch input and output of this layer. By applying equation 4, the next layer's activations for each ensemble member can be computed in a mini-batch friendly way. As a results, this ensemble method allows us to take full advantage of parallel accelerators for efficiently training multiple trading experts simultaneously. In practice, we can divide the input mini-batch in M sub-batches and each sub-batch receives ensemble weight $\overline{W}_i$ to match the input and the ensemble weight,.

Individual Ensemble Loss. For the computation of ensemble loss, we feed the feature vector x into the neural network and get predicted stock movement $\hat{y_i}$ of expert i. To combine the loss of multiple experts, one way is to use the aggregated classification logits. We define the collaborated aggregated loss as:

$$L_{collaborative}^{classify}(x, y) = L_{classify}\left(\frac{1}{n}\sum_{i=1}^{n} \hat{y_i}, y\right) \tag{5}$$

where $L_{classify}$ denotes any classification loss (e.g., cross-entropy). However, collaborative loss leads to correlated decisions [24] instead of complementary experts in our preliminary experiments. To discourage correlations, we require each expert to do the job well by itself. Therefore, we apply an effective individual loss as:

$$L_{individual}^{classify}(x, y) = \frac{1}{n}\sum_{i=1}^{n} L_{classify}(\hat{y_i}, y)) \tag{6}$$

Ensembling During Testing. For inference, we first acquire the predictions of each ensemble member and then average them as the final prediction. For the scenarios of batch size B and M ensemble members, we repeat the data of the input mini-batch M times to get an effective batch size $B * M$, which enables all ensemble members to compute the output of the same B input data points in a single forward pass. As a result, it reduces the computation cost by eliminating the need to calculate the output of each ensemble member sequentially [43].

3.3 Hyper-Level Expert Augmentation

In professional trading firms, there is usually an administrative staff who collects and organizes the investment suggestions from different groups of trading experts as supportive information for portfolio managers' final decisions. Motivated by this, we introduce a way to construct an expert pool exploiting two sources of diversity: varied hyperparameters [45] and random initialization [13], which fully utilizes the investment decisions diversity of different groups of trading experts. We proceed in three main steps as summarized in Algorithm 1 following [45]. Specifically, random_search is a function that train models[2], i.e., $f_\theta(\cdot, \lambda)$, with random selected hyperparameters λ (e.g., learning rate, hidden size). topk_ens is a function that pick models with top K performance. In lines 1-2, we run random search of hyperparameters, which leads to a set of N models (i.e., $\mathcal{M}_0$). Then, we generate one "row" of K models using topk_ens to pick K models with better performance. We then tile and stratify that "row" by training the models for K different random initialization, thus $O(K^2)$ models to train in total (lines 4-7). A pictorial view of the expert pool is illustrated in Figure 4. We show that the expert pool of two strong ensemble methods, i.e., DeepEns [22] and HyperEns [45], are subsets of that for AlphaMix, which indicates that AlphaMix has the potential to achieve better performance by utilizing both sources of diversity.

4 ROUTING EXPERTS DYNAMICALLY

The role of a senior portfolio manager is to make the final investment decisions based on the suggestions from different groups of junior trading experts. We try three different mechanisms to act as a portfolio manager by dynamically picking experts from the pool. First, we propose a novel routing mechanism to route the experts on an as-needed basis [41]. Second, we provide one simple yet robust ensemble aggregation option named with-replacement selection [2] that builds ensembles where the contribution of each

[2] each $f_\theta(\cdot, \lambda)$ is a model trained with the efficient ensemble approach in Section 3.2 that represents the investment decision of one group of experts.

Algorithm 1: Construct A Diversified Expert Pool

```
$\mathcal{M}_0 = \{f_{\theta_j}(\cdot, \lambda_j)\}_{j=1}^{N} \leftarrow$ random_search(N);
$\varepsilon_0 \leftarrow$ topk_ens($\mathcal{M}$, $K$) and $\varepsilon_{strat} = \{\}$;
foreach $f_\theta(\cdot, \lambda) \in \varepsilon_0$.unique() do
    foreach $k \in \{1, ..., K\}$ do
        $\theta' \longleftarrow$ random initialization;
        $f_{\theta_k}(\cdot, \lambda) \longleftarrow$ train $f_{\theta'}(\cdot, \lambda)$ ;
        $\varepsilon_{strat} = \varepsilon_{strat} \cup \{f_{\theta_k}(\cdot, \lambda)\}$
    end
end
return $\varepsilon_{strat}$
```

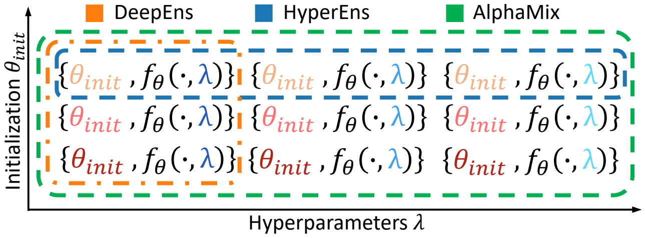
DeepEns
HyperEns
AlphaMix
Initialization θ_{init}
$\{\theta_{init}, f_\theta(\cdot, \lambda)\}$
Hyperparameters λ

Figure 4: Pictorial view of the expert pool. Our AlphaMix can search in the whole "block" (green), while Deepens and Hyperens are built on one column (orange) or row (blue).

ensemble member is weighted. Further, we customize the idea of model soup [47] from pre-train settings, which directly mixes the weights of neural networks, to incorporate different trading experts' knowledge without any extra training or inference cost. See Section 6.2 for detailed performance comparison of the three mechanisms.

4.1 Training As-Needed Routers

For a portfolio manager, it is a smart way to listen to suggestions from junior traders sequentially and make final investment decisions once they believe necessarily enough suggestions are already considered. This helps them avoid hesitation due to too many suggestions and make timely profitable decisions. Motivated by this, we train $n-1$ routers with parameters $h_{\theta_1}, ..., h_{\theta_{n-1}}$ to dynamically deploy these (arbitrarily ordered) experts sequentially on an as-needed basis as shown in Figure 5 inspired by [41]. When the mean logits of the first k-th expert provide the correct prediction ($\hat{y} = y$), which indicates the first k experts are enough, the router should ideally switch off the $k+1$ expert with $y_{on} = 0$. Otherwise, the router should turn on the $k+1$ expert to explore more possibility with $y_{on} = 1$. In practice, we build a binary classifier with two fully connected layers to learn each router. Each of the $n-1$ routers for n experts has a shared component to reduce feature dimensions and an individual component to make decisions. Specifically, we normalize the market feature x and reduce the embedding dimension (8 in our experiments) by a fully connected layer W_1, which is shared with all routers, followed by LeakyReLU, and then concatenate it with the mean logits $\hat{y}_{mean}^{k} = \frac{1}{k}\sum_{i=1}^{k} \hat{y}_i$ of the first k experts. Furthermore, we project it to a scalar by $W_2^{(k)}$ which is independent between routers, and finally apply Sigmoid function $S(x) = \frac{1}{1+e^{-x}}$ to get a continuous activation value $v(x) \in [0, 1]$:

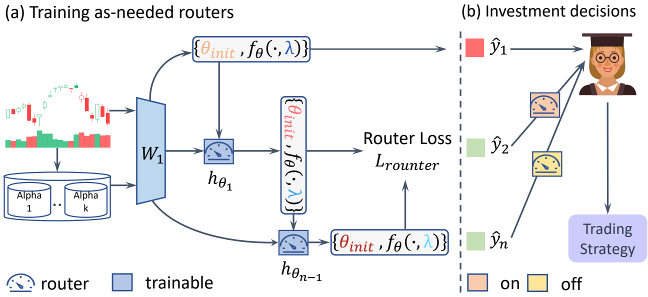

Figure 5: Illustration of AlphaMix with as-needed routers.

$$v(x) = S\left(W_2^{(k)} \begin{bmatrix} LeakyRelu(W_1 \frac{x}{\|x\|}) \\ \hat{y}_{mean}^{k} \end{bmatrix}\right) \tag{7}$$

The routers are optimized with a weighted variant of binary cross-entropy loss:

$$L_{router} = -\omega y_{on} log(v(x)) - (1 - y_{on}) log(1 - v(x)) \tag{8}$$

where ω controls the easiness to switch on the routers. We find $\omega = 1.7$ to be a good trade-off between performance and computational cost for both datasets. At the test time, we simply threshold the activation with 0.5 to decide the on/off of the current router.

4.2 With-Replacement Experts Selection

In real-world trading, a portfolio manager may choose to give more weights on suggestions of some traders based on his/her preferences. Correspondingly, we adopt a simple yet efficient mechanism called with-replacement model selection [2] as shown in Algorithm 2 to enable weighted contribution of each member. Given a set of models (a pool of trading experts), we build an ensemble greedily until the target ensemble size is met by selecting the model leading to the optimal improvement of profits with replacement. This selection strategy enables to construct ensembles where the contribution of each member is weighted. To properly deal with the situation that there might be multiple times the same model is selected, we use .unique() here to address this issue.

Algorithm 2: With-Replacement Selection

```
Input: ensemble $\varepsilon = \{\}$, ensemble size $K$, validation profit
       Pro($\cdot$), Pro$_{best} = +\infty$, model set $\mathcal{M} = \{f_{\theta_1}, ..., f_{\theta_n}\}$
while $\varepsilon$.unique() $\leq$ K do
    $f_{\theta^*} = \text{argmax}_{f_{\theta_i} \in \mathcal{M}} \text{Pro}(\varepsilon \bigcup \{f_{\theta_i}\})$;
    if Pro($\varepsilon \cup \{f_{\theta^*}\}$) < Pro$_{best}$ then
        $\varepsilon = \varepsilon \cup \{f_{\theta^*}\}$, Pro$_{best}$ = Pro($\varepsilon$);
    else
        return $\varepsilon$;
    end
end
return $\varepsilon$
```

4.3 Integrated Model Soup

Instead of making investment decision directly based on suggestions of other trading experts, another working paradigm for portfolio managers is to integrate the market understanding and analysis of others and derive investment decisions based on their own experience. Inspired by it, we customize the idea of model soup [47] into financial markets, which directly mixs the network weights of different experts, i.e., their knowledge and market understanding. Unlike the above two mechanisms that mix the output of different experts, it is perhaps surprising that averaging the weights of independently trained models achieves high performance since the loss landscape of neural network training is nonconvex with many solutions in different loss basins [47]. The insight here is that fine-tuned models optimized independently with the same initialization or hyperparameters may lie in the same basin of the error landscape [29]. As shown in Algorithm 3, the greedy soup sequentially adds each model as a potential ingredient in the soup, and only pick the model in the soup if performance on the validation set improves. We sort the models in decreasing order of validation set performance to ensure that the greedy soup can be no worse than the best individual model. In the end, we average the weights of all ingredient networks in the soup with function average() and construct one network without any extra inference cost.

Algorithm 3: Greedy Expert Soup

```
Input: Model set $\mathcal{M} = \{f_{\theta_1}, ..., f_{\theta_n}\}$ (sorted in decreasing
       order of validation profit $\text{Pro}(f_{\theta_i})$),
       soup ingredients $\varphi = \{\}$
for $i \leftarrow 1$ to $k$ do
    if $\text{Pro}(\text{average}(\varphi \bigcup \{f_{\theta_i}\})) \geq \text{Pro}(\text{average}(\varphi))$ then
        $\varphi = \varphi \bigcup \{f_{\theta_i}\}$
    end
end
return $\text{average}(\varphi)$
```

5 EXPERIMENT SETUP

5.1 Datasets

To conduct a comprehensive evaluation of AlphaMix, we evaluate it on two real-world datasets from US and Chinese stock markets spanning over five years as representatives of developed and developing financial markets, respectively. We summarize statistics of the two datasets in Table 1 and further elaborate on them as follows:

ACL18 [51] is a widely used public dataset collected using Yahoo Finance[3], which contains historical stock prices of 88 US stocks with top capital size from 9 different industry categories: Basic Materials, Consumer Goods, Healthcare, Services, Utilities, Conglomerates, Financial, Industrial Goods and Technology, in the Nasdaq exchange.

SZ50 is a dataset with 4-year (2016-2020) historical prices of 47[4] Chinese stocks collected from Yahoo Finance[3]. The stock pool contains component stocks of SSE50 Index, which represents top 50 companies by "float-adjusted" capitalization in Shanghai exchange.

Dataset	Market	Freq	Stock	Days	From	To
ACL18	US	1d	88	1258	12/09/01	17/09/01
SZ50	China	1h	47	1036	16/06/01	20/09/01

Table 1: Dataset statistics detailing market, data frequency, number of stocks, trading days and chronological period[5].

For dataset split, we use data of the last year for test, penultimate year for validation and the remaining for training on both datasets. It is worth mentioning that the test period of ACL18 (16/09/01 to 17/09/01) is a stable period of economic growth while the test period of SZ50 (19/09/01 to 20/09/01) goes through an unstable period due to the panic decline during the COVID-19 pandemic. Experimental results in Table 3 show the great performance of AlphaMix under different market status.

5.2 Features

We generate 11 temporal features as shown in Table 2 to describe the stock markets following [11, 54]. z_{open}, z_{high} and z_{low} represent the relative values of the open, high, low prices compared with the close price at current time step, respectively. z_{close} and z_{adj_close} represent the relative values of the closing and adjusted closing prices compared with time step $t-1$. z_{dk} represents a long-term moving average of the adjusted close prices during the last k time steps compared to the current close price.

Features	Calculation Formula
$z_{open}, z_{high}, z_{low}$ z_{close}, z_{adj_close}	e.g., $z_{open} = open_t/close_t - 1$ e.g., $z_{close} = close_t/close_{t-1} - 1$
$z_{d\_5}, z_{d\_10}, z_{d\_15}$ $z_{d\_20}, z_{d\_25}, z_{d\_30}$	e.g., $z_{d\_5} = \frac{\sum_{i=0}^{4} adj\_close_{t-i}/5}{adj\_close_t} - 1$

Table 2: Formulas of features to describe the stock markets.

5.3 Evaluation Metrics

We evaluate AlphaMix on 7 different financial metrics including 1 profit criterion, 3 risk criteria and 3 risk-adjusted profit criteria:

- **Total Return (TR)** is the overall return rate of the whole trading period. It is defined as $TR = \frac{n_t - n_1}{n_1}$, where n_t is the final net value and n_1 is the initial net value.
- **Volatility (VOL)** is the variance of the daily return defined as $\sigma[\mathbf{ret}]$ to measure the risk level of trading strategies, where $\mathbf{ret} = (ret_1, ret_2, ..., ret_t)$ is a vector of daily return and $\sigma[.]$ is the variance.
- **Downside Deviation (DD)** is the variance of negative return of the whole trading period.
- **Maximum Drawdown (MDD)** measures the largest loss from any peak to show the worst case.
- **Sharpe Ratio (SR)** considers the amount of extra return that a trader receives per unit of increase in risk. It is defined axs: $SR = E[\mathbf{ret}]/\sigma[\mathbf{ret}]$, where $E[.]$ is the expected value.
- **Calmar Ratio (CR)** is defined as $CR = \frac{E[\mathbf{ret}]}{MDD}$, which is the expected return divided by the maximum drawdown.

[3] Yahoo Finance: https://github.com/yahoo-finance

[4] Three stocks are filtered out from the whole stock pool of size 50 since they are not listed in 2016.

[5] The dates are in the formats YY/MM/DD, where 12/09/01 indicates September 1st of year 2012.

- **Sortino Ratio (SoR)** applies downside deviation as the risk measure. It is defined as: $SoR = \frac{E[\mathbf{ret}]}{DD}$.

5.4 Training Setup

We conduct all experiments on a Tesla V100 GPU. To build the expert pool of AlphaMix, we apply grid search to train experts with batch size in list $[32, 64, 128, 256]$, hidden size in range $[32, 64, 128]$, and learning rate $\alpha \in [5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}]$. In addition, we explore the number of experts in range 2 to 32. Adam is used as the optimizer and we train AlphaMix for 10 epochs on all datasets. It takes about 1.5 and 1.1 hours to train and test on ACL18 and SZ50 datasets, respectively. As for other baselines, there are two conditions: i) there are authors' official or open-source library [53] implementations, we apply the same hyperparameters for a fair comparison[6]. ii) if there are no publicly available implementations, we reimplement the algorithms and try our best to maintain consistency based on the original papers.

5.5 Baselines

To provide a comprehensive comparison of AlphaMix, we select 11 state-of-the-art and representative stock prediction methods of 4 different types consisting of recurrent neural network (RNN), non-recurrent neural network (NRNN), boosting decision tree (BDT) and ensemble learning (ENS) methods. Descriptions of baselines are as follows:

Recurrent Neural Network (RNN)

- **ALSTM [34]** adds an external attention layer into LSTM to attentively aggregate information from all hidden states.
- **LSTM [28]** uses a two-layer vanilla LSTM network with hidden size 32 to predict stock movement.
- **SFM [55]** is a state frequency memory recurrent network, which decomposes prices into signals of different frequencies via Discrete Fourier Transform.
- **GRU [39]** is a new version of recurrent networks with gated recurrent units. We apply a two-layer GRU with hidden size 64 for stock movement.

Non-Recurrent Neural Network (NRNN)

- **GTrans [6]** is a novel hierarchical Gaussian transformer-based model for stock prediction.
- **MLP [27]** applies multi-layer perceptron for stock movement prediction with hidden size 128.

Boosting Decision Tree (BDT)

- **LightGBM (LGB) [20]** is an efficient implementation of gradient boosting decision tree with gradient-based one-side sampling and exclusive feature bundling.
- **CatBoost (CatB) [33]** is a gradient boosting toolkit with ordered boosting and categorical feature processing.

Ensemble Learning Methods (ENS)

- **DNNE [52]** is a deep neural network ensemble method for stock market prediction based on bagging.
- **DeepEns [22]** is a simple yet efficient ensemble method with remarkable accuracy and robust uncertainty estimates.
- **HyperEns [45]** builds ensembles that involve a random search over different hyperparameters.

6 RESULTS AND ANALYSIS

6.1 Comparison with Baselines

We compare AlphaMix[7] with 11 state-of-the-art baselines in terms of 7 financial criteria in Table 3. We observe that AlphaMix consistently generates significantly higher performance than other baselines on one profit and three risk-adjusted profit metrics across both US and China markets. In the US market, AlphaMix outperforms the second best by 12%, 17%, 58%, 5% in terms of TR, SR, CR, and SoR. As for the China market, AlphaMix outperforms the second best by 30%, 27%, 116%, 17% in terms of TR, SR, CR, and SoR. For risk metrics, AlphaMix performs the best in MDD, which shows its resistance to drawdown. For VOL and DD, AlphaMix achieves 1st and 2nd places on the US market that demonstrates its stellar performance on risk control. As for the China market, AlphaMix comes up with higher VOL and DD due to the trade-off between fluctuation and return in developing stock markets. In general, ensemble learning methods perform better than single deep learning methods (RNN and NRNN), as they combine multiple submodels for more robust and profitable investment decisions. We postulate that both RNN and NRNN methods have the potential to capture the internal patterns of financial data since both RNN and NRNN methods achieve decent performance on the two datasets. Boosting decision tree models tend to overfit to the noise in training data and generalize poorly in test data with the lowest profitability. In addition, we observe that DeepEns is the most powerful baseline with at least the second best performance on 7 out of 14 metrics. MLP achieves robust performance on both markets and CatBoost performs particularly well for risk control in the China market.

6.2 Ablation

We conduct ablation studies on AlphaMix's performance benefits from different ensemble construction mechanisms. There are four variants of AlphaMix: i) AlphaMix-V is a vanilla version that simply averages the prediction of each group of expert; ii) AlphaMix-R trains extra as-needed routers (Sec 4.1); iii) AlphaMix-W uses the heuristic with-replacement selection (Sec 4.2) and iv) AlphaMix-S applies model soup (Sec 4.3). We note that AlphaMix-V and AlphaMix-S can be considered as an enhanced version of [43, 47] and outperform the original version in our preliminary experiments. In this subsection, we compare them with the three ensemble baselines in Table 4. First, we observe that AlphaMix outperforms the three ensemble baselines in most cases (red indicates worse than baselines). The AlphaMix-V is one simple yet efficient choice on both markets. With extra training of as-needed routers, AlphaMix-R performs the best. For AlphaMix-W, it outperforms AlphaMix-V in terms of profit, but the MDD is higher. For AlphaMix-S with model soup, it does not perform as well as it is in computer vision [47] due to the low signal-to-noise ratio of stock markets. In summary, we recommend AlphaMix-V or AlphaMix-R for users, who prefer simple yet robust methods or best performance, respectively.

[6] This condition fits for most baselines except GTrans and DNNE.

[7] We report the experiment results of AlphaMix with as-needed routers in this Section if not particularly pointed out.

			Profit	Risk-Adjusted Profit			Risk Metrics		
Market	Type	Model	TR(%)↑	SR↑	CR↑	SoR↑	VOL(%)↓	DD(%)↓	MDD(%)↓
ACL18	RNN	ALSTM	9.9±8.1	0.76±0.61	1.76±1.91	1.14±0.99	0.82±0.03	0.56±0.04	6.88±1.43
		LSTM	19.5±4.3	1.49±0.34	3.31±1.15	2.30±0.59	0.82±0.02	0.54±0.04	6.21±1.26
		SFM	15.1±3.0	1.25±0.23	1.86± 0.48	1.94± 0.32	0.76±0.02	0.49±0.04	8.40± 1.58
		GRU	14.1±7.9	1.06±0.61	2.27±1.92	1.52±0.96	0.84±0.02	0.61±0.05	7.96± 2.59
	NRNN	GTrans	19.8±9.1	1.52 ±0.67	3.45 ±2.37	2.42±1.16	0.81±0.06	0.52±0.06	7.08 ±2.57
		MLP	22.1±6.3	1.87±0.57	3.69±1.77	2.52±0.88	0.75±0.03	0.57±0.07	6.82±2.15
	BDT	LGB	4.8	0.4	0.70	0.53	0.81	0.57	6.76
		CatB	9.3±3.3	0.70±0.25	1.31±0.55	1.00±0.37	0.83±0.03	0.59±0.03	7.33±0.91
	ENS	DNNE	23.3±9.4	1.68±0.72	3.76±2.52	2.73±1.24	0.90±0.14	0.56±0.09	7.99±3.37
		DeepEns	22.4	2.10	4.79	3.42	0.69	0.42	5.53
		HyperEns	23.3±3.2	1.96±0.28	3.39±0.78	2.67±0.42	0.74±0.02	0.54±0.02	6.88±0.97
	Ours	AlphaMix	26.1	2.45	7.59	3.59	0.67	0.45	3.43
% Improvement over SOTA			12%	17%	58%	5%	3%	-	61%
SZ50	RNN	ALSTM	23.2±7.0	1.07±0.26	1.59±0.48	1.34±0.33	1.38±0.09	1.10±0.08	14.74±1.89
		LSTM	20.7±7.8	1.01±0.33	1.57±0.52	1.28±0.42	1.30±0.07	1.03±0.05	13.02±0.85
		SFM	17.3±2.3	0.79±0.08	1.16±0.15	1.04±0.11	1.40±0.17	1.07±0.15	14.95±2.17
		GRU	36.5±27.9	1.26±0.76	2.00±1.35	1.74±1.14	1.66±0.39	1.22±0.22	16.81±2.96
	NRNN	GTrans	16.8±22.2	0.59±0.77	0.85±1.19	0.80±1.08	1.90±0.28	1.44±0.21	25.41±6.97
		MLP	43.5±7.3	1.82±0.15	2.87±0.30	2.46±0.24	1.53±0.15	1.13±0.09	15.36±3.23
	BDT	LGB	15.3	0.67	0.85	0.85	1.48	1.16	17.91
		CatB	21.0±1.8	1.06±0.08	1.72±0.23	1.43±0.12	1.27±0.11	0.95±0.10	12.28±0.72
	ENS	DNNE	40.8±8.1	1.66±0.35	3.25±0.90	2.27±0.56	1.58±0.05	1.17±0.08	12.82±1.43
		DeepEns	48.2	1.79	3.13	2.39	1.73	1.29	15.57
		HyperEns	44.3±7.9	1.83±0.31	2.92±0.56	2.51±0.46	1.51±0.06	1.10±0.08	15.32±1.68
	Ours	AlphaMix	62.6	2.34	7.04	2.95	1.72	1.37	8.89
% Improvement over SOTA			30%	27%	116%	17%	-	-	38%

Table 3: Performance comparison (mean and standard deviation of 10 individual runs) on ACL18 (US) and SZ50 (China) stock markets with 11 baselines including recurrent network (RNN), non-recurrent network (NRNN), boosting decision tree (BDT) and ensemble (ENS) learning methods. LightGBM (LGB), DeepEns and AlphaMix are three *deterministic* methods without the performance standard deviation. Results in pink, green and blue show best, second best and third best results on each dataset.

	ACL18 (US)			SZ50 (China)		
Models	TR(%)↑	SR↑	MDD↓	TR(%)↑	SR↑	MDD↓
DNNE	23.3	1.68	7.99	40.8	1.66	12.82
DeepEns	22.4	2.10	5.53	48.2	1.79	15.57
HyperEns	23.3	1.97	6.88	44.3	1.83	15.32
AlphaMix-V	24.5	2.39	3.95	53.7	2.08	11.19
AlphaMix-R	26.1	2.45	**3.43**	**62.6**	**2.34**	**8.89**
AlphaMix-W	**26.7**	**2.54**	5.83	61.0	2.24	12.10
AlphaMix-S	26.0	2.31	6.64	52.7	2.04	13.80

Table 4: Ablation studies over the effectiveness of different routing mechanisms of AlphaMix on US and China markets. Red indicates worse than either DNNE, DeepEns or HyperEns. Bold shows the best results.

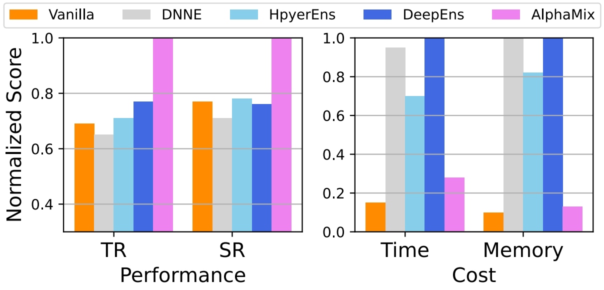

Figure 6: AlphaMix and several other methods on China market. AlphaMix achieves the best trade-off among performance (↑) and time & memory (↓) costs.

6.3 Computational Cost

We show AlphaMix is an efficient ensemble method that addresses the extensive computation cost overhead of traditional ensemble methods. Figure 6 displays results on China market in term of performance (TR & SR) and cost (time & memory). Although DeepEns and HyperEns slightly outperform the vanilla single model, their computation cost is much higher. For AlphaMix, it significantly outperforms all three other methods (over 20% improvement) with much less computation overhead. The poor performance of DNNE further shows that straightforward applications of traditional ensemble methods are not enough in financial markets.

6.4 Diversity Analysis

To evaluate diversity, we use the predictive disagreement metric from [13]. This metric is based on the average of the pairwise comparisons of the predictions across the ensemble members. For a given pair of members, it is zero when they are making identical predictions, and one when all their predictions differ. We report the diversity of AlphaMix-V and two baselines in Table 5. The diversity of AlphaMix is significantly better than other two baselines as it utilizes two sources of diversity: hyperparameter (H) level and initialization (I) level.

6.5 Reduce Uncertainty with AlphaMix

We conduct experiments to show a single neural network is sensitive to random seeds with high uncertainty and AlphaMix can tackle

	Diversity	Ens Size	ACL18 (US)	SZ50 (China)
DeepEns	I	4	0.4194	0.2832
HyperEns	H	4	0.3454	0.3992
AlphaMix	I & H	4	**0.5947**	**0.6124**

Table 5: Diversity comparison of the predictive disagreement. AlphaMix performs the best by utilizing two sources of diversity: hyperparameter (H) level and initialization (I) level.

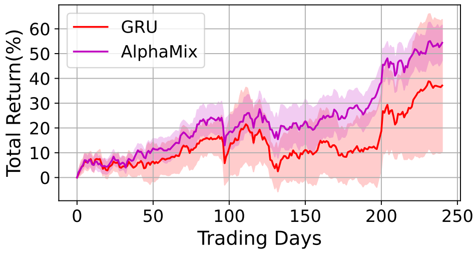

Figure 7: Trading days vs. total return curve. AlphaMix achieves higher return with much lower standard deviation.

this issue with mixture-of-experts. We train a popular two-layer GRU [39] model and our AlphaMix for stock movement prediction and generate investment decisions with the classic top-4 buy & hold trading strategy with 10 independent runs. The trading days vs. total return curve (mean and standard deviation) of the two models is reported in Figure 7. AlphaMix achieves higher return with much lower standard deviation, which makes investment decisions from AlphaMix more reliable. To analysis the potential reason of this phenomenon, we plot the confidence score (largest logits after softmax) for stock movement prediction (binary classification) in Figure 8. The confidence score of GRU is quite low (between 0.5 and 0.6 for most samples) that indicates one single predictor is not enough. We observe that AlphaMix achieves much higher confidence score with mixture-of-experts. It is reasonable that higher confidence score leads to lower performance variance.

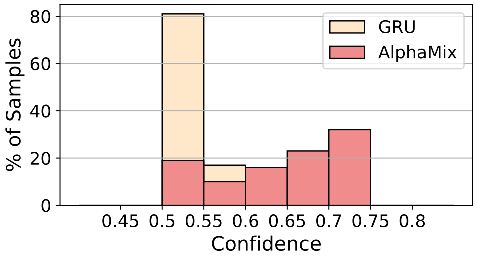

Figure 8: Confidence histogram to show AlphaMix has higher decision certainty comparing to GRU.

6.6 Parameter Analysis: Probing Sensitivity

Number of selected top k stocks. We analyze AlphaMix's performance (TR & SR) variation with the number of selected top stocks on ACL18 and SZ50 in Figure 9. We find that AlphaMix performs consistently well in terms of TR and SR on both datasets, which shows AlphaMix's robustness and suitability to strategies with different risk taking appetites. We pick $k = 4$ in practice.

Number of experts e. We analyze AlphaMix's performance (TR & SR) variation with the number of experts e on ACL18 and SZ50 in Figure 10. We observe that AlphaMix performs the best when e is relatively small. Then, the performance drops a lot with the increase of expert number. In addition, when the number of experts goes to very large (e.g., 32). The performance increases again. This indicates that it is better to make investment decision based on the discussion of a few elite or a large population of experts. Decisions form medium size experts may lead to worse decisions.

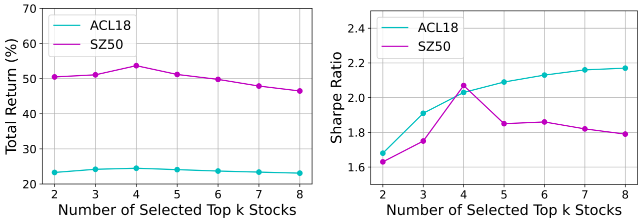

Figure 9: The impact of the number of top k stocks on AlphaMix in terms of TR and SR on ACL18 and SZ50.

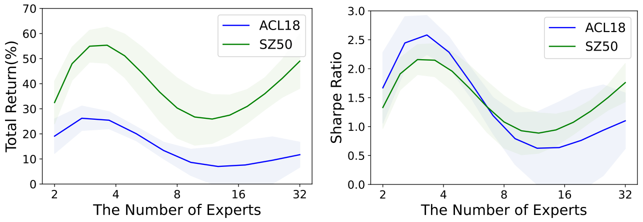

Figure 10: The impact of the number of experts on AlphaMix in terms of TR and SR on ACL18 and SZ50.

7 CONCLUSION

In this paper, we propose AlphaMix, a novel three-stage mixture-of-experts framework, to mimic the efficient bottom-up workflow in real-world trading firms for quantitative investment. Specifically, we introduce an efficient ensemble learning method to train multiple trading experts with personalised market understanding and trading styles. Later on, we conduct model augmentation to construct a pool of diversified trading experts. Finally, we leverage three different mechanisms, namely as-needed router, integrated soup and with-replacement selection, to further improve performance by dynamically routing experts from the expert pool. Extensive experiments on both China and US stock markets demonstrate that AlphaMix significantly outperforms many strong baselines. Ablation studies show the effectiveness of the proposed components.

8 ACKNOWLEDGMENTS

This project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. 2015. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications* 42, 20 (2015), 7046–7056.

[2] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of the International Conference on Machine learning*.

[3] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira. 2016. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications* 55 (2016), 194–211.

[4] Lakshay Chauhan, John Alberg, and Zachary Lipton. 2020. Uncertainty-aware lookahead factor models for quantitative investing. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 1489–1499.

[5] 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.

[6] 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 29th International Joint Conference on Artificial Intelligence (IJCAI)*. 4640–4646.

[7] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2021. GLaM: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905* (2021).

[8] Bernard Dumas. 1978. The theory of the trading firm revisited. *The Journal of Finance* 33, 3 (1978), 1019–1030.

[9] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314* (2013).

[10] Eugene F Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance* 25, 2 (1970), 383–417.

[11] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. 2018. Enhancing stock movement prediction with adversarial training. *arXiv preprint arXiv:1810.09936* (2018).

[12] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–30.

[13] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (2019).

[14] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. 2021. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342* (2021).

[15] Raphael Gontijo-Lopes, Yann Dauphin, and Ekin Dogus Cubuk. 2022. No One Representation to Rule Them All: Overlapping Features of Training Methods. In *International Conference on Learning Representations*.

[16] Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, 10 (1990), 993–1001.

[17] 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 11th ACM International Conference on Web Search and Data Mining (WSDM)*. 261–269.

[18] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109* (2017).

[19] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation* 3, 1 (1991), 79–87.

[20] 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. In *33th Conference on Neural Information Processing Systems (NeurIPS)*.

[21] Anders Krogh and Jesper Vedelsby. 1994. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems* 7 (1994).

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems* 30 (2017).

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[24] 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 & Data Mining*. 1017–1026.

[25] Guang Liu, Yuzhao Mao, Qi Sun, Hailong Huang, Weiguo Gao, Xuan Li, Jianping Shen, Ruifan Li, and Xiaojie Wang. 2020. Multi-scale two-way deep neural network for stock trend prediction.. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*. 4555–4561.

[26] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[27] Mahdi Pakdaman Naeini, Hamidreza Taremian, and Homa Baradaran Hashemi. 2010. Stock market value prediction using neural networks. In *2010 International Conference on Computer Information Systems and Industrial Management Applications*. 132–136.

[28] 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)*. 1419–1426.

[29] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? *Advances in Neural Information Processing Systems* 33 (2020), 512–523.

[30] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. 2020. A comprehensive evaluation of ensemble learning for stock-market prediction. *Journal of Big Data* 7, 1 (2020), 1–40.

[31] David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* 11 (1999), 169–198.

[32] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[33] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: Unbiased boosting with categorical features. In *34th Conference on Neural Information Processing Systems (NeurIPS)*.

[34] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 2627–2633.

[35] Carlos Riquelme Ruiz, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. In *37th Conference on Neural Information Processing Systems (NeurIPS)*.

[36] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. 2020. Deep attentive learning for stock movement prediction from social media text and company correlations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8415–8426.

[37] Ramit Sawhney, Piyush Khanna, Arshiya Aggarwal, Taru Jain, Puneet Mathur, and Rajiv Shah. 2020. VolTAGE: Volatility forecasting via text-audio fusion with graph convolution networks for earnings calls. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8001–8013.

[38] 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).

[39] Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng, and Jianjun Xu. 2018. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science* 131 (2018), 895–903.

[40] Jia Wang, Tong Sun, Benyuan Liu, Yu Cao, and Hongwei Zhu. 2019. 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 (IJCAI)*. 3705–3711.

[41] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. 2020. Long-tailed recognition by routing diverse distribution-aware experts. In *9th International Conference on Learning Representations (ICLR)*.

[42] Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. 2020. Deep mixture of experts via shallow embedding. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*. 552–562.

[43] Yeming Wen, Dustin Tran, and Jimmy Ba. 2019. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*.

[44] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. 2018. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. In *International Conference on Learning Representations*.

[45] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. 2020. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems* (2020), 6514–6527.

[46] Tim Whitaker and Darrell Whitley. 2022. Prune and tune ensembles: low-cost ensemble learning with sparse independent subnetworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8638–8646.

[47] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482* (2022).

[48] Cheng Xiang and WM Fu. 2006. Predicting the stock market using multiple models. In *2006 9th International Conference on Control, Automation, Robotics and Vision*. 1–6.

[49] Jin Xu, Jingbo Zhou, Yongpo Jia, Jian Li, and Xiong Hui. 2020. An adaptive master-slave regularized model for unexpected revenue prediction enhanced with alternative data. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 601–612.

[50] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. Rest: Relational event-driven stock trend forecasting. In *Proceedings of the Web Conference 2021*. 1–10.

[51] 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 (ACL)*. 1970–1979.

[52] Bing Yang, Zi-Jia Gong, and Wenqi Yang. 2017. Stock market index prediction using deep neural network ensemble. In *2017 36th Chinese Control Conference (CCC)*. 3882–3887.

[53] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. 2020. Qlib: An AI-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189* (2020).

[54] Jaemin Yoo, Yejun Soun, Yong-chan Park, and U Kang. 2021. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2037–2045.

[55] 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 & Data Mining*. 2141–2149.