2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

When Coordinated Knowledge Distillation Meets Mixture of Expert Inference: Insights from Portfolio Optimization

Jianzeng Song[1], Senjie Xia[1], Yong Zhang[1], Jie Wei[2] and Jianfei Yin[1]*

[1] Shenzhen University, Canghai Campus Nanshan District, Shenzhen, China
[2] Shenzhen SDG Information Co., Ltd., 18F, Block B, Tefa Infoport, No. 2 Kefeng Rd, Nanshan District, Shenzhen, China
{songjianzeng2023, xiasenjie2023, zhang-yong2023}@email.szu.edu.cn, weijie@sdgi.com.cn, yjf@szu.edu.cn

Abstract. To achieve stable profits in uncertain financial environments characterized by pervasive noise signals, unavoidable transaction costs and zero-sum dynamics, it is crucial to construct optimized portfolios based on comprehensive data processing. However, existing methods often overlook the importance of learning hedge financial knowledge from data and leveraging mixture-of-expert (MoE) inferences to maximize agent profitability. To address this issue, we propose the Coordinated Knowledge Distillation and Inference Framework (CKDIF). CKDIF introduces a three-dimensional discrete coordinate system to train deep reinforcement learning agents with hedge trading behaviors, enabling the effective distillation of underlying micro-financial knowledge directly from noisy financial data. Furthermore, CKDIF constructs a novel ensemble of MoE networks by harnessing these pretrained agents and uses the ensemble to make final portfolio selection across any asset dimension. Notably, with transaction costs set at a realistic rate of 0.1%, CKDIF outperforms eight representative algorithms on five out of six real-world financial datasets. It achieves an average cumulative wealth and Calmar ratio that are 1.66 and 3.70 times higher, respectively, compared to the buy-and-hold strategy. These results underscore the potency of coordinated knowledge distillation and MoE inference in enhancing agent performance in competitive environments.

Keywords: Coordinated Knowledge Distillation, Mixture of Experts, Portfolio Optimization.

1 Introduction

Achieving stable returns in real-world financial markets is challenging due to the complex nature of optimization under financial uncertainty [1]. Unlike well-studied domains like natural language processing (NLP) and computer vision (CV), financial data present unique difficulties for developing automated portfolio optimization agents. The main challenges are: (i) Pervasive noise signals—such as sentiment-driven fluctuations,

* Corresponding author

oversold/overbought conditions, and black swan events—distort financial data, reducing the effectiveness of traditional mean-variance portfolio optimization, especially in short-term strategies [2]; (ii) Unavoidable transaction costs incurred during each trading action can significantly erode profits, especially in market conditions like bear markets. This poses a challenge to the trial-and-error learning paradigm commonly used in deep reinforcement learning (DRL) [3]; (iii) Zero-sum competition causes strategies to lose effectiveness over time, making it essential to maintain multiple coexisting strategies for consistent returns [4, 5]. However, many approaches converge to a single strategy, neglecting the importance of diversification.

To address these issues, we emphasize the extraction and use of hedge financial knowledge, which represents the collective behavior of DRL agents exhibiting hedge trading strategies [6]. This expands traditional diversification and allows inference-based adaptation to evolving markets. However, existing knowledge distillation methods face several limitations: (i) information loss that reduces diversity in trend learning; (ii) lack of hedge-oriented behaviors in agent strategies [7–9]; and (iii) difficulty incorporating essential financial knowledge—such as sparse portfolio construction [10] and transaction cost control [3]—due to challenges in unifying these objectives during training.

To overcome these limitations, we propose the Coordinated Knowledge Distillation and Inference Framework (CKDIF)[1]. CKDIF effectively learns and utilizes hedge trading knowledge for robust portfolio optimization. Our key contributions include:

- A novel three-dimensional discrete coordinate system for training DRL agents to effectively capture hedge trading knowledge from noisy data.
- An ensemble of Mixture-of-Expert (MoE) networks built from these pretrained agents for robust and generalized inference across diverse asset dimensions.
- A unified framework incorporating domain-specific knowledge—hedge trading strategies, portfolio sparsity, and transaction cost control—into the distillation and inference process.

To evaluate CKDIF, we tested it on six real-world financial datasets with a realistic 0.1% transaction cost. CKDIF outperformed eight baseline methods on five datasets, achieving cumulative wealth and a Calmar ratio that are 1.66 times and 3.70 times higher, respectively, than the buy-and-hold strategy. These results highlight the effectiveness of coordinated knowledge distillation and MoE inference in boosting agent performance in competitive environments.

The remainder of this paper is structured as follows: Section 2 reviews related work in portfolio optimization. Section 3 outlines the CKDIF workflow. Section 4 details the coordinated knowledge distillation process used to obtain pretrained agents. Section 5 describes the inference mechanism on these agents. Section 6 reports extensive experimental results. Finally, Section 7 concludes the paper.

[1] The code is available at https://github.com/ccckkkyyy666/CKDIF

2 Related Work

This section reviews related work on knowledge distillation and ensemble agent methods for portfolio optimization.

2.1 Knowledge Distillation

Knowledge distillation originated from the need to compress knowledge from large-sized models into smaller ones [7-9]. As a learning method, it traditionally involves two roles: the teacher and the student models. In the financial domain, one of the main motivations for utilizing knowledge distillation is its effectiveness in handling noisy financial data. Notably, Tsantekidis et al. [9] employed a diverse collection of teacher models to handle transactions in various currencies, allowing the student network to distill common insights. Similarly, Chen et al. [7] developed a student-teacher framework where multiple agents distill market information to train a student DRL model, demonstrating profitable strategies and flexible asset allocation. Additionally, Moustakidis et al. [8] proposed an online distillation method for DRL agents that transfers both output and intermediate-layer knowledge from a teacher ensemble to a student model, improving training stability and performance in noisy financial environments. However, these methods have certain limitations: (i) Student models indirectly learn financial knowledge from teacher models, which may result in information loss during knowledge propagation and reduced behavior diversity in learning; (ii) The pool of students lacks training in hedge trading behaviors; (iii) Crucial portfolio-specific knowledge, such as sparse portfolio construction and transaction cost control, remains unlearned from the teachers.

2.2 Ensemble of Agents

The utilization of multiple trading agents can enhance portfolio diversification while enabling online inference capabilities [7, 11, 12]. For instance, Chen et al. [7] introduced a role-aware multi-agent algorithm that categorizes trading agents into distinct groups, providing diverse observation sets and reward functions to simulate real-world investment behaviors. Similarly, Shavandi and Khedmati [11] trained agents using deep Q-Networks (DQN) across diverse time frames. However, relying solely on price information from different time intervals limits the exploitation of hedging opportunities and diverse price information. Another study by Yang et al. [12] employed the Sharpe ratio to automatically select the best-performing agent from an ensemble of proximal policy optimization (PPO), advantage actor-critic (A2C), and deep deterministic policy gradient (DDPG) algorithms. Nevertheless, a common issue with these ensemble methods is the lack of behavior diversity in constructing the agent pool, as the agents often share similar trading styles due to using the same optimization objective during training. This can lead to a situation where all agents exhibit the same “follow-the-winner” trading behavior.

3 The Workflow of CKDIF

The CKDIF workflow (Fig. 1) outlines data flows across four key steps, each using a distinct dataset marked by color. It consists of two stages: Stage 1 (Steps 1–2) distills knowledge from training data into agents as a living knowledge base, while Stage 2 (Steps 3–4) trains the MoE network and performs inference on this base.

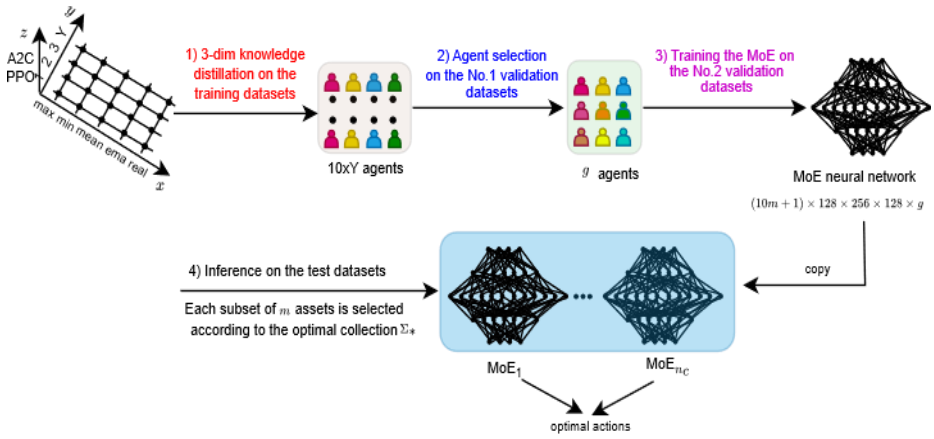


Fig. 1. The workflow of CKDIF.

3.1 Datasets Allocation

The datasets, detailed in Table 1, used in the CKDIF workflow are obtained from Yahoo Finance, encompassing trading data from global markets, including the Dow Jones Industrial Average Stock Index (DOW), Shanghai and Shenzhen Indexes (HS), Hong Kong Index (HK), Financial Times Stock Exchange Index (FTSE), New York Stock Exchange Index (NYSE), as well as global cryptocurrency market data (CRYPTO). These datasets reflect a range of market conditions, including black swan events and regime shifts. For instance, the CRYPTO market experienced a major shift when Bitcoin dropped from $45,308 to $41,454, an 8.51% decline, triggering nearly $600 million in liquidations. These events test a portfolio strategy's ability to adjust to extreme market conditions and confirm its generalization to out-of-sample events.

The training datasets, using asset set Ω^{train} over n_{train} periods, are used to train agents. No.1 validation datasets, with asset set Ω^1 over n_1 periods, are employed to select agents exhibiting hedge trading behaviors. Both sets contain 29 assets, matching the neural network's capacity m. The No.2 validation datasets are used to train the MoE network with asset set Ω^2 over n_2 periods.

Table 1. The allocation of datasets.

Market	Training Datasets		No.1 Validation Datasets		No.2 Validation Datasets		Test Datasets	
	Periods	#Assets	Periods	#Assets	Periods	#Assets	Periods	#Assets
DOW	05/01/2010-31/12/2013	29	05/01/2014-05/01/2015	29	06/01/2015-06/01/2016	29	07/01/2016-06/01/2017	29
HS	04/01/2011-31/12/2014	29	06/01/2015-06/01/2016	29	07/01/2016-06/01/2017	63	09/01/2017-08/01/2018	63
CRYPTO	10/11/2017-10/11/2021	29	11/11/2021-10/11/2022	29	11/11/2022-11/11/2023	33	12/11/2023-29/04/2024	33
HK	05/01/2015-31/12/2018	29	03/01/2019-03/01/2020	29	06/01/2020-04/01/2021	73	05/01/2021-05/01/2022	73
NYSE	02/01/2013-30/12/2016	29	02/01/2017-02/01/2018	29	03/01/2018-03/01/2019	62	04/01/2019-03/01/2020	62
FTSE	04/01/2017-31/12/2020	29	04/01/2021-04/01/2022	29	05/01/2022-05/01/2023	75	06/01/2023-05/01/2024	75

4 Coordinated Knowledge Distillation

We employ a 3D coordinate system to train a pool of agents $\boldsymbol{\mathcal{A}}^{(x,y,z)}$ with the goal of acquiring highly effective hedge financial knowledge. Each agent, denoted as $\boldsymbol{\mathcal{A}}^{(x,y,z)}$, can be viewed as a function $\boldsymbol{\mathcal{A}}^{(x,y,z)}$: $\mathbb{R}^{10m+1} \to \mathbb{R}^m$, mapping inputs from the real space $\mathbb{R}^{10m+1}$ to trading actions in the action space $\mathbb{R}^m$, where m represents the maximum number of assets that can be processed by the neural network implementing these agents.

4.1 Agent State Space

For each agent $\boldsymbol{\mathcal{A}}^{(x,y,z)}$, its state $\boldsymbol{s}_t^x \in \mathbb{R}^{10m+1}$ is defined as follows:

$$\boldsymbol{s}_t^x = \mathrm{vec}(cash_t, \boldsymbol{p}_t^x, \boldsymbol{h}_t, \mathbf{ind}_t). \tag{1}$$

The symbol $cash_t \in \mathbb{R}_+$ represents the remaining cash at period t. The notation $\boldsymbol{p}_t^x \in \mathbb{R}^m$, as defined in Table 2, refers to the price-trend features utilized by agents $\boldsymbol{\mathcal{A}}^{(x,y,z)}$ whose first index is x. The vector $\boldsymbol{h}_t \in \mathbb{R}_+^m$ represents the holdings of m assets by an agent, and the vector $\mathbf{ind}_t \in \mathbb{R}^{m\times 8}$ contains eight technical indicators for each asset, as described in Table 3. The operator vec vertically stacks the input vectors $cash_t$, $\boldsymbol{p}_t^x$, $\boldsymbol{h}_t$, and $\mathbf{ind}_t$ to form a state $\boldsymbol{s}_t^x$. Consequently, the dimension of the state $\boldsymbol{s}_t^x$ is determined to be $10m + 1$.

The input price vector $\boldsymbol{p}_t \in \mathbb{R}_+^m$ in Table 2 represents the close prices of m assets, while the parameter $w \in \mathbb{R}_+$ denotes the time window used to calculate these features, and the parameter $\beta \in (0, 1)$ represents the weight used in computing the feature vector $\boldsymbol{p}_t^{ema}$ through the exponential moving average formula.

Table 2. Definition of price-trend features $\boldsymbol{p}_t^x$.

	x				
	max	min	$mean$	ema	$real$
$\boldsymbol{p}_t^x$	$\max_{0\le k\le w-1} \boldsymbol{p}_{t-k}$	$\min_{0\le k\le w-1} \boldsymbol{p}_{t-k}$	$\frac{1}{w}\sum_{k=0}^{w-1} \boldsymbol{p}_{t-k}$	$\sum_{k=0}^{w-1} \beta(1-\beta)^k \boldsymbol{p}_{t-k}$	$\boldsymbol{p}_t$

Table 3. Technical indicators used in the vector $\mathbf{ind}_t$.

Indicator	Description
macd	Moving average convergence divergence
boll_ub	Bollinger bands upper band
boll_lb	Bollinger bands lower band
rsi_30	Relative strength index for 30 periods
cci_30	Commodity channel index for 30 periods
dx_30	Directional movement index for 30 periods
ma_30	Simple moving average of closing prices for 30-periods
cci_30	Simple moving average of closing prices for 60-periods

4.2 Agent Action Space and Training Objective

All agents $\boldsymbol{\mathcal{A}}^{(x,y,z)}$ operate within an action space that encompasses all allowable trading actions $\boldsymbol{a}_{t+1}^{y} \in \mathbb{R}^m$ for m assets in a given state $\boldsymbol{s}_t^x$ at period t. These actions include buying ($a_{t+1,i}^{y} > 0$), selling ($a_{t+1,i}^{y} < 0$), or holding ($a_{t+1,i}^{y} = 0$) assets. For instance, $\boldsymbol{a}_{t+1}^{y} = (10, -10, 0, ...)$ indicates buying 10 shares of the first asset, selling 10 shares of the second asset, and holding the remaining shares of the third asset.

It is essential that agents' actions follow a hedge behavior structure while maximizing individual profits. Therefore, we propose the following optimization objective for training each agent $\boldsymbol{\mathcal{A}}^{(x,y,z)}$:

$$\min -\eta L_1^y + L_2, \tag{2}$$

where $\eta \in \mathbb{R}_+$ is a preset constant. The first item L_1^y is defined as follows:

$$L_1^y = \sum_{\tau \in [0, n_{train}]} \sum_{j=1}^{y-1} \mathrm{KL}(\boldsymbol{a}_\tau^j || \boldsymbol{a}_\tau^y). \tag{3}$$

Eq. (3) sums the KL divergences between actions $\boldsymbol{a}_\tau^y$ from $\boldsymbol{\mathcal{A}}^{(x,y,z)}$ and actions $\boldsymbol{a}_\tau^j$ from $\boldsymbol{\mathcal{A}}^{(x',j,z')}$, where the trading style index j < y. The second term in Eq. (2) is the PPO or A2C loss, aiming to maximize rewards via advantage estimation. Together, these terms guide agents to distill hedge financial knowledge in a greedy fashion. Agents $\boldsymbol{\mathcal{A}}^{(x,y,z)}$ are trained using the library stable-baselines3[2]. Training complexity is $O(\boldsymbol{\mathcal{N}}_a \times \frac{\boldsymbol{\mathcal{R}}_a}{\boldsymbol{\mathcal{B}}_a} \xi)$, where $\boldsymbol{\mathcal{N}}_a$ iterations $\boldsymbol{\mathcal{R}}_a$ samples of observation–reward pairs, and each network update (cost ξ) uses a batch of $\boldsymbol{\mathcal{B}}_a$ samples.

4.3 Training and Selection of Agents via the 3D Coordinate System

Our agent pool, denoted as $\boldsymbol{\mathcal{A}}^{(x,y,z)}$, is trained within a 3D coordinate system (Fig. 1). The first index, $x \in \{max, min, mean, ema, real\}$, defines the set of price features

[2] https://github.com/DLR-RM/stable-baselines3/blob/master/stable_baselines3/common/on_policy_algorithm.py

used for training. The second, $y \in \{1, 2, ..., Y\}$, specifies the desired trading style. The third, $z = \{PPO, A2C\}$, indicates the training algorithm: Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C). We employ both PPO, known for robust exploration, and A2C, recognized for responsive policy updates, to cultivate a diverse set of agents tailored to various financial strategies and market dynamics. With 5 price features, Y trading styles and 2 algorithms, we train a total of $10Y$ agents $\boldsymbol{\mathcal{A}}^{(x,y,z)}$.

To select agents exhibiting hedge behaviors, we identify the top and bottom performers within each of the 6 markets in the No.1 validation dataset. This process yields a maximum of 12 selected agents for the subsequent inference phase. However, the actual number g of unique selected agents may be less than 12 due to potential repetitions, as illustrated in Appendix A where g=9.

5 Inference over Agents

This section details the process of generating final trading decisions by performing inference on the trained DRL agents. This is achieved by constructing a Mixture-of-Experts (MoE) neural network and further enhancing it through an ensemble of multiple MoE instances to improve robustness and performance.

5.1 MoE Neural Network

The MoE neural network can be conceptualized as a function MoE: $\mathbb{R}^{10m+1} \rightarrow \mathbb{R}^m$ and is defined as follows:

$$\underset{128\times1}{\boldsymbol{u}_1} = \underset{128\times(10m+1)}{\boldsymbol{W}_1}\ \underset{(10m+1)\times1}{\boldsymbol{s}_t^{real}} + \underset{128\times1}{\boldsymbol{b}_1}, \tag{4a}$$

$$\boldsymbol{u}_1 = \mathrm{Dropout}(\gamma_e) \circ \mathrm{LeakyRelu}(\boldsymbol{u}_1), \tag{4b}$$

$$\underset{256\times1}{\boldsymbol{u}_2} = \underset{256\times128}{\boldsymbol{W}_2}\ \underset{128\times1}{\boldsymbol{u}_1} + \underset{256\times1}{\boldsymbol{b}_2}, \tag{4c}$$

$$\boldsymbol{u}_2 = \mathrm{Dropout}(\gamma_e) \circ \mathrm{LeakyRelu}(\boldsymbol{u}_2), \tag{4d}$$

$$\underset{128\times1}{\boldsymbol{u}_3} = \underset{128\times256}{\boldsymbol{W}_3}\ \underset{256\times1}{\boldsymbol{u}_2} + \underset{128\times1}{\boldsymbol{b}_3}, \tag{4e}$$

$$\boldsymbol{u}_3 = \mathrm{Dropout}(\gamma_e) \circ \mathrm{LeakyRelu}(\boldsymbol{u}_3), \tag{4f}$$

$$\underset{g\times1}{\boldsymbol{u}_4} = \underset{g\times128}{\boldsymbol{W}_4}\ \underset{128\times1}{\boldsymbol{u}_3} + \underset{g\times1}{\boldsymbol{b}_4}, \tag{4g}$$

$$\boldsymbol{u}_4 = \mathrm{Softmax} \circ \mathrm{Topk}(\boldsymbol{u}_4, k = 2), \tag{4h}$$

$$\underset{m\times1}{\boldsymbol{c}_t} = \Big[\dots \underset{m\times g}{\boldsymbol{\mathcal{A}}^{(x_i,y_i,z_i)}}(\boldsymbol{s}_t^{real}) \dots\Big] \underset{g\times1}{\boldsymbol{u}_4}. \tag{4i}$$

For the network design, we adopt a repeated three-layer block composed of fully connected layers with Dropout and LeakyRelu activation. The hidden dimension is first expanded to 256 to explore richer feature interactions, then reduced to 128 to filter out redundant components and retain salient signals. LeakyRelu is chosen to maintain stable gradient flow and mitigate dead neuron issues. In the final layer, the output vector $\boldsymbol{u}_4$ is used to compute a weighted sum of portfolio vectors proposed by the trained DRL agents. According to Eq. (4i), the domain of $\boldsymbol{c}_t$ matches the agents' action space $\mathbb{R}^m$, making $\boldsymbol{c}_t$ interpretable as traders' trading actions. The MoE neural network is trained

using the following loss function:

$$loss = -\big(cash_t + \boldsymbol{h}_t^{\intercal}\boldsymbol{p}_t - (cash_{t-1} + \boldsymbol{h}_{t-1}^{\intercal}\boldsymbol{p}_{t-1})\big). \quad (5)$$

At each period t, the loss is calculated as the difference in capital between two consecutive periods, with $\boldsymbol{h}_t^{\intercal}\boldsymbol{p}_t$ representing the estimated value of shares $\boldsymbol{h}_t$, obtained from the agent trading environment *MoEEnv*, as described in Step 9 of Algorithm 1. The MoE neural network is trained on samples from the No.2 validation dataset[3] using Algorithm 1, with support from the FinRL framework[4].

Algorithm 1 MoE Neural Network Training Algorithm

Input: Price vectors: $\{\boldsymbol{p}_t: \boldsymbol{p}_t \in \mathbb{R}_+^{|\Omega|}\}_{t=1}^{n_2}$, where Ω is a subset of assets from a No.2 validation dataset; a set of g agents: $\{\boldsymbol{\mathcal{A}}^{(x_i,y_i,z_i)}\}_{i=1}^{g}$; number of iterations $\boldsymbol{\mathcal{N}}_e$, learning rate λ_e, and dropout rate γ_e;
Procedure:
Initialize MoE network weights $\boldsymbol{\mathcal{W}}$, as defined in Eq. (4)
Initialize *optimizer* = *optim.Adam*($\boldsymbol{\mathcal{W}}$, λ_e)
Initialize cumulative wealth ω_0 = 1, 000, 000
MoEEnv = *StockTradingEnv* (ω_0).get_sb_env()
for $i = 1 \rightarrow \boldsymbol{\mathcal{N}}_e$ **do**
  Initialize $cash_0 = \omega_0$, $\boldsymbol{b}_0$ = [0, ..., 0]
  for $t = 1 \rightarrow n_2$ **do**
    compute state $\boldsymbol{s}_t^{real}$ via Eq. (1) and action $\boldsymbol{c}_t$ via Eq. (4)
    obtain $cash_t$, $\boldsymbol{p}_t$, and $\boldsymbol{h}_t$ via invoking *MoEEnv*.step($\boldsymbol{c}_t$)
    compute $loss$ via Eq. (5)
    optimizer.zero_grad(); *loss*.backward(); *optimizer*.step()
Output: network weights $\boldsymbol{\mathcal{W}}$

In Step 4, the variable *MoEEnv* represents an instance of the *StockTradingEnv* class, which encapsulates crucial trading state information, including price vectors $\boldsymbol{p}_t$, remaining cash $cash_t$, cumulative wealth ω_t, and asset holdings $\boldsymbol{h}_t$, among others. The computational complexity of Algorithm 1 can be estimated as $O(\boldsymbol{\mathcal{N}}_e \times n_2\ [m(1280 + g) + g(\xi + 128)])$, where ξ denotes the computational cost associated with inference on an agent's neural network.

5.2 Varied Inference Intervals

Transaction fees significantly affect portfolio profitability for agents. To mitigate this, we propose a heuristic to determine the optimal inference interval d_t^* at period t, based on the following objective:

$$d_t^* = \arg\max_{d \in D} \left\{ \alpha(\omega_t - \omega_{t-d})/d + (1-\alpha)\sqrt{\textstyle\sum_{k=1}^{d}(\Delta\omega_{t-k} - \overline{\Delta\omega_t})^2/d} \right\}. \quad (6)$$

Here, $\alpha \in [0, 1]$ is a preset interval coefficient, and $D \in 2^{\mathbb{N}}$ is the set of candidate inference intervals. ω_t denotes cumulative portfolio wealth, with change $\Delta\omega_t = \omega_t - \omega_{t-1}$.

[3] The training datasets for the MoE neural network can be accessed at https://github.com/ccckkkyyy666/CKDIF/blob/main/Data/data/forMoE.csv.

[4] https://github.com/AI4Finance-Foundation/FinRL

Objective (6) balances the growth ratio of ω_t and the variance of recent increase ratios over window d. Details on selecting D and α are provided in Appendix B.

5.3 CKDIF Inference Algorithm

This section introduces an ensemble of MoE networks to make final trading decisions on the test datasets Ω^{test}. To handle potential mismatches between the number of assets in Ω^{test} and the maximum asset capacity m of a single MoE network (as defined in Eq. (4)), this ensemble, denoted as $\{\mathrm{MoE}_i\}$, is constructed as follows:

- We sample a collection of asset subsets denoted as $\Sigma = \{\Omega_1, ..., \Omega_{n_c} : |\Omega_i| = m, \Omega_i \subseteq \Omega^{test}\}$. Here, n_c is calculated as $n_c=\lceil\frac{|\Omega^{test}|}{m}\rceil$, where Ω^{test} represents the asset set in the given test dataset.
- Construct $\{\mathrm{MoE}_i\}$ to use the sampled collection Σ to generate the final trading actions $\boldsymbol{c}'_t \in \mathbb{R}^m$ according to the following rule:

$$\boldsymbol{c}'_t = \mathrm{Sparse}\big(\mathrm{MoE}_j\big(\boldsymbol{s}_t^{real}; \Omega_j\big); \Omega_j\big), \tag{7}$$

where j =arg $\max_{i\in[1,n_c],\, \Omega_i\in\Sigma} \mathrm{MoE}_i\big(\boldsymbol{s}_t^{real}; \Omega_i\big).cash$.

The notation $(\boldsymbol{s}_t^{real}; \Omega_i)$ indicates that the state $\boldsymbol{s}_t^{real}$ must be constructed using only the features corresponding to the asset subset Ω_i, such as the price-trend vector $\boldsymbol{p}_t^{real}$ and the holdings $\boldsymbol{h}_t$ for assets in Ω_i. The term $\mathrm{MoE}_i\big(\boldsymbol{s}_t^{real}; \Omega_i\big).cash$ represents the remaining cash balance in the trading environment of MoE_i after executing the action $\mathrm{MoE}_i\big(\boldsymbol{s}_t^{real}; \Omega_i\big)$. The function Sparse: $\mathbb{R}^m \times 2^{\mathbb{N}} \rightarrow \mathbb{R}^m$ is designed to transform an action $\boldsymbol{c}_t \in \mathbb{R}^m$ into a sparse action $\boldsymbol{c}'_t \in \mathbb{R}^m$ according to the following rule:

$$\mathrm{Sparse}\big(c_{t,i}; \Omega_j\big) = \begin{cases} -h_{t,i}, \text{if } i \neq k \wedge h_{t,i} > 0, \\ \big(cash_t + \sum_{l\neq k} p_{t,l} h_{t,l}\big)/p_{t,i}, \text{if } i = k, \end{cases} \tag{8}$$

where k is defined as k = arg $max_i c_{t,i}$, and $h_{t,i} \in \mathbb{R}_+$, for i =1, ..., m, represents the number of shares of each asset in Ω_j that the action $\boldsymbol{c}_t$ operates upon.

Finally, we introduce the CKDIF inference algorithm (Algorithm 2) for generating final trading actions. CKDIF adaptively updates the optimal inference interval d^* at each period t, triggered when t is a multiple of the previous d^* (Step 10). Its computational complexity is approximated as $O(\frac{n}{min\ D} \times \lceil \frac{|\Omega|}{\mathrm{m}} \rceil \times [m(1280 + g) + g(\xi + 128)])$, where ξ is the cost of inferring from a single agent's network.

To prepare the optimal asset subset collection Σ_* for invoking Algorithm 2, multiple candidates $\Sigma_i = \{\Omega_1^i, \dots, \Omega_{n_c}^i\}$ are sampled from validation set Ω^2, where $|\Omega_j^i| = m$ and $n_c = \lceil\frac{|\Omega^2|}{m}\rceil$. The optimal Σ_* is chosen as $\mathrm{argmax}_{\Sigma_i} \mathrm{CIA}(\Sigma_i, D, \alpha)$, with CIA denoting Algorithm 2.

Algorithm 2 CIA: CKDIF Inference Algorithm

Input: Price vectors: $\{\boldsymbol{p}_t : \boldsymbol{p}_t \in \mathbb{R}_+^{|\Omega|}\}_{t=1}^{n}$, where Ω is an asset set; collection of asset subsets: $\Sigma = \{\Omega_1, \ldots, \Omega_{n_c} : |\Omega_i| = m, \Omega_i \subseteq \Omega\}$, where $n_c = \left\lceil \frac{|\Omega|}{m} \right\rceil$; n_c MoE neural networks and $n_c \times g$ agents; candidate inference intervals D, and interval coefficient α;

Procedure:

Initialize the cumulative wealth $\omega_0 = 1,000,000$, inference interval $d^* = D[0]$
EnsembleEnv = *StockTradingEnv*(ω_0).get_sb_env()
for $i = 1 \rightarrow n_c$ **do**
    MoE_i = *StockTradingEnv*(ω_0).get_sb_env()
for $t = 1 \rightarrow n$ **do**
    compute state $\boldsymbol{s}_t^{real}$ via Eq. (1), and compute final action $\boldsymbol{c}_t'$ via Eq. (7) using Σ and $\boldsymbol{s}_t^{real}$
    if $t <$ max(D) **then**
        obtain ω_t via invoking *EnsembleEnv*.step($\boldsymbol{c}_t'$); continue
    if t % $d^* == 0$ **then**
        obtain ω_t via invoking *EnsembleEnv*.step($\boldsymbol{c}_t'$); update interval d^* via Eq. (6)
Output: the final cumulative wealth ω_n

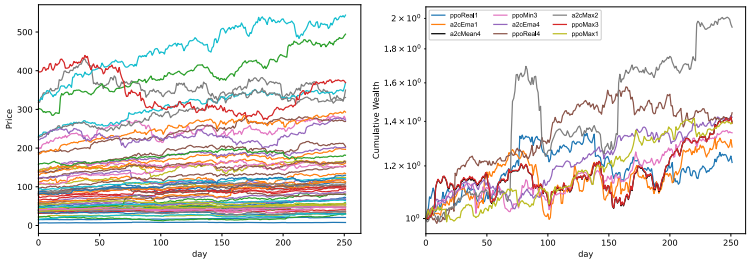
6 Experiment Results

We conducted experiments on the test datasets (Table 1) using a machine with an AMD Ryzen 7 5800H, GeForce RTX 3060 GPU, and 16GB RAM.

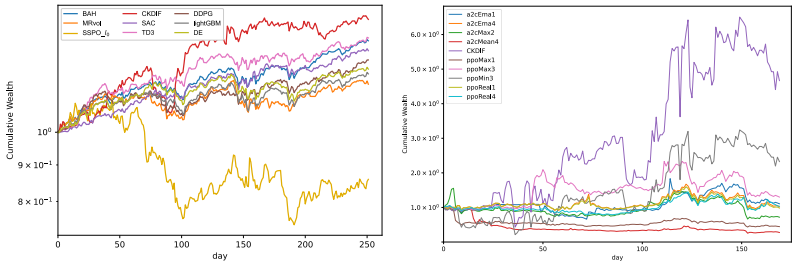
6.1 Effect of Knowledge Distillation and MoE Inference

To assess the effect of knowledge distillation, we evaluated nine agents (Appendix A) on the datasets in Table 1. Fig. 2a shows the price series of 62 assets over 250 periods in the NYSE dataset, where hedge patterns are subtle and hard to identify. In contrast, Fig. 2b illustrates the diverse trading behaviors learned by the nine agents. For example, agent a2cMax2 exhibits distinct trading dynamics from a2cEma1 during days 75–90, and from ppoReal4 during days 150–250, highlighting strategy diversity.

The effectiveness of MoE inference is demonstrated in Fig. 2c, where CKDIF, leveraging MoE-based inference across the trained agent ensemble, significantly outperforms the other eight methods on the NYSE dataset. A similar performance advantage is shown in Fig. 2d for the CRYPTO dataset, where no individual agent exceeds CKDIF's performance. These results confirm that MoE inference improves both generalization and robustness beyond what single-agent approaches can achieve.



(a) Asset prices in NYSE dataset (b) Agent performance on NYSE dataset



(c) Performance comparison on NYSE dataset (d) Performance comparison on CRYPTO dataset

Fig. 2. Effect of knowledge distillation and MoE inference.

6.2 Inference Performance Comparison

Table 4 presents a performance comparison against a diverse set of baselines: the traditional buy-and-hold (BAH) strategy, the mean-reversion multi-expert strategy MRvol [13], the sparse strategy SSPO_l_0 [14], the tree-based model LightGBM [18], and ensemble techniques like the Double Ensemble (DE) [19]. Our evaluation also includes enhanced implementations of DRL algorithms (SAC[15], TD3[16], and DDPG[17]) using the Ray RLlib framework[5].

Cumulative Wealth. Cumulative wealth (CW) evaluates long-term performance as the ratio of final to initial wealth $\mathrm{CW} \coloneqq \frac{\omega_n}{\omega_0} = \prod_{t=1}^{n} \left(\frac{\boldsymbol{p}_t}{\boldsymbol{p}_{t-1}}\right)^{\intercal} \boldsymbol{b}_t$, where $\boldsymbol{b}_t \in \mathbb{R}_+^m$ is a portfolio vector with $\boldsymbol{b}_t^{\mathrm{T}}\mathbf{1} = 1$. Table 4 shows that CKDIF consistently outperforms other algorithms in CW, ranking first on five of six datasets. Notably, on the CRYPTO dataset, its CW of 4.67 more than doubles MRvol's 1.56. This strong performance highlights the effectiveness of coordinated knowledge distillation and MoE inference.

Sharpe Ratio. The Sharpe ratio (SR) evaluates risk-adjusted portfolio returns, defined as $\mathrm{SR} \coloneqq \frac{\mathbb{E}[R-R_f]}{\sigma[R-R_f]}$. CKDIF achieves the highest SRs on the CRYPTO and HK datasets and ranks second on FTSE, just 0.04 behind the top method. However, since SR treats all return variance $\sigma[R - R_f]$ as risk, it may overestimate risk during periods of consistent growth.

Maximum Drawdown. Maximum Drawdown (MDD) measures the largest loss from peak to trough before recovery. While CKDIF shows higher MDDs than the best-performing algorithms in this metric, this is due to its higher trading frequency. Notably, CKDIF achieves significantly greater Cumulative Wealth (CW). For instance, on the CRYPTO dataset, MRvol has the lowest MDD but a CW of only 1.56—about one-third of CKDIF's 4.67—highlighting the trade-off between MDD and CW.

[5] https://github.com/ray-project/ray/tree/master/rllib/algorithms

Calmar Ratio. Calmar Ratio (CAR) measures investment performance by dividing annualized return (AR) by maximum drawdown (MDD). The AR is computed as: $\mathrm{AR} := (1 + \mathrm{CR})^{\frac{252}{n}} - 1$, where $\mathrm{CR} := \frac{\omega_n}{\omega_0} - 1$ represents the cumulative return. CKDIF outperforms on CAR, achieving top CARs on HS and CRYPTO datasets, second on FTSE, and third on HK.

Table 4. Inference performance comparison (best scores in bold).

Datasets	Metrics	BAH	CKDIF	MRvol	SSPO_l_0	SAC	TD3	DDPG	LightGBM	DE
DOW	CW	1.21±0.00	**1.37±0.00**	1.25±0.00	1.10±0.00	1.14±0.00	1.32±0.00	1.22±0.00	1.17±0.00	1.14±0.02
	SR	1.52±0.00	1.50±0.00	1.44±0.00	0.40±0.00	1.29±0.00	**2.15±0.00**	1.62±0.00	-0.01±0.00	-0.02±0.01
	MDD (%)	**5.15±0.00**	14.42±0.00	7.49±0.00	13.62±0.00	6.83±0.00	6.18±0.00	5.59±0.00	25.40±0.00	26.25±0.38
	CAR	4.01±0.00	2.55±0.00	3.28±0.00	0.70±0.00	2.10±0.00	**5.16±0.00**	3.60±0.00	0.14±0.01	0.23±0.06
HS	CW	1.23±0.00	**1.69±0.00**	1.13±0.00	0.79±0.00	1.45±0.00	1.17±0.00	1.26±0.00	1.16±0.00	1.23±0.02
	SR	2.28±0.00	1.97±0.00	1.04±0.00	-0.80±0.00	**2.73±0.00**	1.42±0.00	2.16±0.00	0.04±0.00	0.06±0.01
	MDD (%)	**6.28±0.00**	10.65±0.00	8.76±0.00	38.70±0.00	8.26±0.00	7.03±0.00	6.60±0.00	16.78±0.00	15.54±0.65
	CAR	3.85±0.00	**6.72±0.00**	1.51±0.00	-0.55±0.00	5.60±0.00	2.56±0.00	4.12±0.00	-0.63±0.00	-1.05±0.14
CRYPTO	CW	1.17±0.00	**4.67±0.00**	1.56±0.00	1.02±0.00	0.73±0.00	0.89±0.00	0.76±0.00	1.36±0.00	1.24±0.10
	SR	0.41±0.00	**2.1±0.00**	1.02±0.00	0.01±0.00	-0.20±0.00	0.02±0.00	-0.16±0.00	0.01±0.00	-0.01±0.02
	MDD (%)	30.35±0.00	75.42±0.00	**21.63±0.00**	75.60±0.00	53.96±0.00	34.57±0.00	51.29±0.00	59.14±0.00	57.27±3.81
	CAR	0.85±0.00	**11.69±0.00**	4.30±0.00	0.03±0.00	-0.68±0.00	-0.45±0.00	-0.65±0.00	-0.11±0.00	0.11±0.18
HK	CW	0.96±0.00	**1.12±0.00**	0.77±0.00	0.67±0.00	0.94±0.00	0.97±0.00	1.06±0.00	0.93±0.01	0.97±0.01
	SR	-0.22±0.00	**0.47±0.00**	-1.02±0.00	-0.52±0.00	-0.17±0.00	-0.18±0.00	0.35±0.00	-0.06±0.00	-0.05±0.00
	MDD (%)	17.14±0.00	34.63±0.00	36.91±0.00	61.38±0.00	17.30±0.00	**16.14±0.00**	18.17±0.00	24.74±0.69	21.11±1.64
	CAR	-0.23±0.00	0.34±0.00	-0.62±0.00	-0.54±0.00	-0.33±0.00	-0.17±0.00	0.34±0.00	**0.77±0.00**	0.70±0.06
NYSE	CW	1.34±0.00	**1.44±0.00**	1.17±0.00	0.86±0.00	1.30±0.00	1.36±0.00	1.26±0.00	1.21±0.00	1.22±0.01
	SR	**2.55±0.00**	1.95±0.00	1.11±0.00	-0.52±0.00	2.41±0.00	2.30±0.00	1.95±0.00	-0.01±0.00	0.00±0.01
	MDD (%)	5.91±0.00	13.79±0.00	9.75±0.00	33.50±0.00	**5.06±0.00**	5.19±0.00	7.69±0.00	14.93±0.00	14.46±1.12
	CAR	5.80±0.00	3.18±0.00	1.72±0.00	-0.42±0.00	5.95±0.00	**6.84±0.00**	3.41±0.00	0.08±0.01	0.01±0.08
FTSE	CW	1.11±0.00	1.36±0.00	**1.40±0.00**	0.36±0.00	1.08±0.00	1.15±0.00	0.89±0.00	1.08±0.03	1.00±0.02
	SR	0.72±0.00	1.31±0.00	**1.61±0.00**	-1.67±0.00	0.53±0.00	1.19±0.00	-0.38±0.00	-0.09±0.01	-0.12±0.01
	MDD (%)	10.98±0.00	19.91±0.00	12.63±0.00	75.29±0.00	19.97±0.00	**9.41±0.00**	33.99±0.00	29.93±1.98	36.94±2.30
	CAR	0.98±0.00	1.79±0.00	**3.14±0.00**	-0.85±0.00	0.41±0.00	1.63±0.00	-0.31±0.00	0.76±0.04	0.83±0.01

7 Conclusion

This paper introduces the Coordinated Knowledge Distillation and Inference Framework (CKDIF), which integrates knowledge distillation with mixture-of-expert inference to achieve stable profits in non-stationary, zero-sum financial markets. CKDIF demonstrates superior performance compared to eight benchmark algorithms in portfolio optimization and brings the pretrain–finetune–inference paradigm from NLP into the financial domain. Future research will explore extending hedge knowledge distillation within the context of established financial theories, such as Markowitz's mean-variance model.

References

1. Xu, J., Li, B.: Uncertain utility portfolio optimization based on two different criteria and improved whale optimization algorithm. Expert Systems with Applications 268, 126281 (2025)
2. Lai, Z.R., Yang, H.: A survey on gaps between mean-variance approach and exponential growth rate approach for portfolio optimization. ACM Computing Surveys (CSUR) 55(2), 1–36 (2022)

3. Zhang, Y., Zhao, P., Wu, Q., Li, B., Huang, J., Tan, M.: Cost-sensitive portfolio selection via deep reinforcement learning. IEEE Transactions on Knowledge and Data Engineering 34(1), 236–248 (2020)
4. Xie, A., Harrison, J., Finn, C.: Deep reinforcement learning amidst continual structured risk-stationarity. In: International Conference on Machine Learning. pp. 11393–11403. PMLR (2021)
5. Li, Y., Wang, P., Chen, H.: Can reinforcement learning solve asymmetric combinatorial-continuous zero-sum games? arXiv preprint arXiv:2502.01252 (2025)
6. Yin, J., Zhong, A., Xiao, X., Wang, R., Huang, J.Z.: An asset subset-constrained minimax optimization framework for online portfolio selection. Expert Systems with Applications 254, 124299 (2024)
7. Chen, M.Y., Chen, C.T., Huang, S.H.: Knowledge distillation for portfolio management using multi-agent reinforcement learning. Advanced Engineering Informatics 57, 102096 (2023)
8. Moustakidis, V., Passalis, N., Tefas, A.: Online probabilistic knowledge distillation on cryptocurrency trading using deep reinforcement learning. Pattern Recognition Letters 186, 243–249 (2024)
9. Tsantekidis, A., Passalis, N., Tefas, A.: Diversity-driven knowledge distillation for financial trading using deep reinforcement learning. Neural Networks 140, 193–202 (2021)
10. Wang, H., Zhang, W., He, Y., Cao, W.: L0-norm based short-term sparse portfolio optimization algorithm based on alternating direction method of multipliers. Available at SSRN 4115395 (2023)
11. Shavandi, A., Khedmati, M.: A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets. Expert Systems with Applications 208, 118124 (2022)
12. Yang, H., Liu, X.Y., Zhong, S., Walid, A.: Deep reinforcement learning for automated stock trading: An ensemble strategy. In: Proceedings of the first ACM international conference on AI in finance. pp. 1–8 (2020)
13. Lin, H., Zhang, Y., Yang, X.: Online portfolio selection of integrating expert strategies based on mean reversion and trading volume. Expert Systems with Applications 238, 121472 (2024)
14. Wang, H., Zhang, W., He, Y., Cao, W.: L0-norm based short-term sparse portfolio optimization algorithm based on alternating direction method of multipliers. Available at SSRN 4115395 (2023)
15. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. International conference on machine learning. pp. 1861–1870. PMLR (2018)
16. Kabbani, T., Duman, E.: Deep reinforcement learning approach for trading automation in the stock market. IEEE Access 10, 93564–93574 (2022)
17. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
18. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems 30 (2017)
19. Zhang, C., Li, Y., Chen, X., Jin, Y., Tang, P., Li, J.: DoubleEnsemble: A new ensemble method based on sample reweighting and feature selection for financial data analysis. In: 2020 IEEE International Conference on Data Mining (ICDM). pp. 781–790. IEEE (2020)

A Agent Selection for MoE Training and Inference

To choose the optimal hedging agents for MoE, we evaluated all agents on the No.1 validation dataset, select the top (green) and bottom (red) performers from each dataset to form the pool of agents for MoE training and inference. As shown in Table 5, the results indicate the selection of $g = 9$ agents, distinguished by at least one colored mark in their corresponding index rows.

Table 5. Agents' performance on the No.1 validation datasets.

Agent	DOW	HS	CRYPTO	HK	NYSE	FTSE
(x=*real*, y=1, z=*PPO*)	0.5232	0.3210	0.1899	0.6599	0.7042	1.2104
(x=*real*, y=2, z=*PPO*)	0.8602	1.4432	0.1377	0.9310	0.6712	1.1029
(x=*real*, y=3, z=*PPO*)	0.6527	0.7428	0.1382	0.6750	0.7000	1.2057
(x=*real*, y=4, z=*PPO*)	0.9868	0.8587	0.0529	1.9492	1.1230	0.7996
(x=*max*, y=1, z=*PPO*)	0.7421	0.5162	0.1354	0.6557	0.6547	0.6684
(x=*max*, y=2, z=*PPO*)	0.5836	0.4920	0.1492	0.8948	0.8232	1.1687
(x=*max*, y=3, z=*PPO*)	1.3249	1.0814	0.1540	1.0976	1.1633	1.8432
(x=*max*, y=4, z=*PPO*)	0.5817	0.6027	0.2063	0.4505	0.7100	1.1924
(x=*min*, y=1, z=*PPO*)	0.6303	0.6352	0.1121	0.7658	0.6259	1.2899
(x=*min*, y=2, z=*PPO*)	1.0763	0.4762	0.3435	0.6649	1.1113	1.1175
(x=*min*, y=3, z=*PPO*)	0.6414	0.2991	0.2976	0.9579	0.7289	0.9617
(x=*min*, y=4, z=*PPO*)	0.6359	0.7204	0.9762	0.7592	0.7525	1.0440
(x=*mean*, y=1, z=*PPO*)	0.8661	0.5902	0.1165	1.1165	0.7892	0.7479
(x=*mean*, y=2, z=*PPO*)	0.7370	0.4645	0.0607	1.0266	0.6646	1.0094
(x=*mean*, y=3, z=*PPO*)	1.1324	0.9966	0.1803	1.3481	0.9609	1.1784
(x=*mean*, y=4, z=*PPO*)	0.8856	0.6454	0.1350	0.6869	0.8572	0.9404
(x=*ema*, y=1, z=*PPO*)	0.6869	1.2475	0.1448	0.6007	0.8046	1.1714
(x=*ema*, y=2, z=*PPO*)	0.9746	0.4739	0.2794	0.7422	1.2960	1.1333
(x=*ema*, y=3, z=*PPO*)	1.1507	0.8396	0.2437	0.5925	0.7859	0.8581
(x=*ema*, y=4, z=*PPO*)	0.9061	0.3174	0.1908	0.5800	0.5818	0.9623
(x=*real*, y=1, z=*A2C*)	0.5998	0.3682	0.3252	0.7487	0.7420	1.2678
(x=*real*, y=2, z=*A2C*)	0.8623	0.9266	0.1327	0.9178	0.5915	1.0924
(x=*real*, y=3, z=*A2C*)	0.7116	1.0866	0.0974	0.4838	0.8692	1.6652
(x=*real*, y=4, z=*A2C*)	0.8317	0.3095	0.1452	0.7582	1.0800	1.0125
(x=*max*, y=1, z=*A2C*)	0.6982	0.3788	0.2160	0.9156	0.4958	0.8679
(x=*max*, y=2, z=*A2C*)	0.7107	0.4791	0.1413	0.4495	0.5569	0.9021
(x=*max*, y=3, z=*A2C*)	1.2263	0.8937	0.0764	0.8408	1.0608	0.9904
(x=*max*, y=4, z=*A2C*)	0.8477	0.5697	0.2136	0.6340	0.6778	1.1404
(x=*min*, y=1, z=*A2C*)	0.6237	0.8784	0.2215	0.9227	0.6700	1.6029
(x=*min*, y=2, z=*A2C*)	0.7596	0.3328	0.1109	1.1281	0.7848	1.1301
(x=*min*, y=3, z=*A2C*)	0.6025	0.3214	0.1128	0.4992	0.7707	1.1402
(x=*min*, y=4, z=*A2C*)	0.7773	1.0657	0.1774	0.7748	0.8251	1.0897
(x=*mean*, y=1, z=*A2C*)	0.7169	0.3598	0.2372	0.4693	1.0868	1.0832
(x=*mean*, y=2, z=*A2C*)	0.7483	0.6707	0.1935	0.8090	0.6761	1.2517
(x=*mean*, y=3, z=*A2C*)	0.6745	0.8021	0.1231	0.8384	0.7100	1.0356
(x=*mean*, y=4, z=*A2C*)	0.7601	1.9874	0.4716	0.8998	0.6967	1.0662
(x=*ema*, y=1, z=*A2C*)	1.4212	1.0063	0.0960	0.6285	1.5695	1.7013
(x=*ema*, y=2, z=*A2C*)	0.6911	0.8570	0.1090	0.9358	0.5363	0.9000
(x=*ema*, y=3, z=*A2C*)	0.6618	0.5491	0.2265	0.7673	0.7857	0.7087
(x=*ema*, y=4, z=*A2C*)	0.8065	0.3185	0.0256	0.6968	0.4573	1.2848

B Parameter Settings of D and α

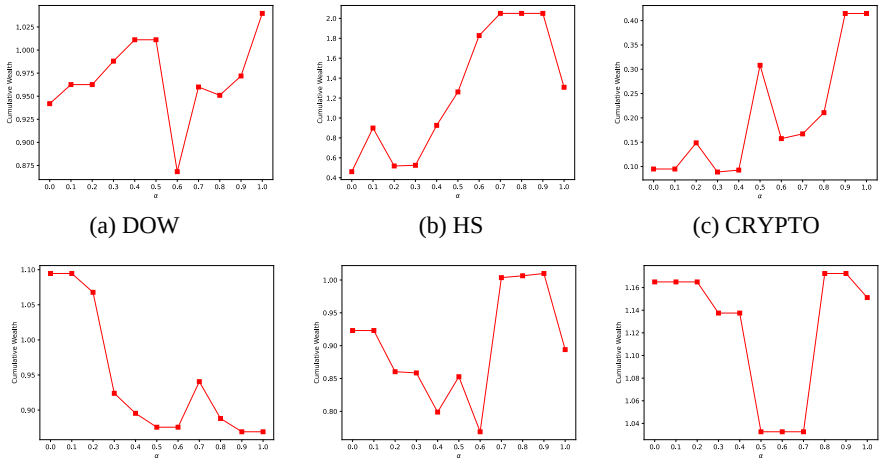
To identify the members d of the inference intervals set D in Eq. (6), we evaluate the performance of a single MoE neural network using various d settings on the No.1 validation datasets. Based on the results in Table 6, we select $D = \{6, 9, 10\}$. This choice balances inference time in the CKDIF inference algorithm while maintaining satisfactory performance.

To determine the optimal setting for the interval coefficient α in Eq. (6), we apply

the CKDIF inference algorithm (Algorithm 2) to the No.1 validation datasets. With fixed inference intervals $D = \{6, 9, 10\}$, we iterate through different values of $\alpha \in [0, 1]$ with a step size of 0.1. The cumulative wealth obtained is depicted in Fig. 3. After evaluating the algorithm's performance across all markets in the No.1 validation datasets, we have chosen $\alpha = 0.45$ as the default value.

Table 6. Exploration of different inference intervals d (best scores in bold).

d	DOW	HS	CRYPTO	HK	NYSE	FTSE
1	0.9550	0.4562	0.3222	0.4954	0.6526	1.0631
2	0.9361	0.5031	0.1279	0.5631	0.8573	1.1681
3	1.0204	0.8358	0.2442	0.8555	0.9181	1.0095
4	0.9810	0.8347	0.0672	0.6855	0.8750	1.1164
5	1.0521	0.3054	**0.3795**	0.8030	0.9497	1.0450
6	0.9666	0.8160	0.2692	**1.0751**	**1.1183**	1.0551
7	1.0706	1.0532	0.0699	0.8845	0.8490	1.1300
8	1.1963	0.7422	0.1654	0.9328	1.0727	1.2171
9	**1.2094**	1.2438	0.1412	0.8147	1.0911	**1.2426**
10	1.1630	1.4161	0.2102	0.9917	0.9910	0.9987



(a) DOW (b) HS (c) CRYPTO

(d) HK (e) NYSE (f) FTSE

Fig. 3. Exploration of different settings of the interval coefficient α.