Federated Learning for Data Trading Portfolio Allocation With Autonomous Economic Agents

Lei Zhao, *Member, IEEE*, Lin Cai, *Fellow, IEEE*, and Wu-Sheng Lu, *Life Fellow, IEEE*

***Abstract*— In the rapidly advancing ubiquitous intelligence society, the role of data as a valuable resource has become paramount. As a result, there is a growing need for the development of autonomous economic agents (AEAs) capable of intelligently and autonomously trading data. These AEAs are responsible for acquiring, processing, and selling data to entities such as software companies. To ensure optimal profitability, an intelligent AEA must carefully allocate its portfolio, relying on accurate return estimation and well-designed models. However, a significant challenge arises due to the sensitive and confidential nature of data trading. Each AEA possesses only limited local information, which may not be sufficient for training a robust and effective portfolio allocation model. To address this limitation, we propose a novel data trading market where AEAs exclusively possess local market information. To overcome the information constraint, AEAs employ federated learning (FL) that allows multiple AEAs to jointly train a model capable of generating promising portfolio allocations for multiple data products. To account for the dynamic and ever-changing revenue returns, we introduce an integration of the histogram of oriented gradients (HoGs) with the discrete wavelet transformation (DWT). This innovative combination serves to redefine the representation of local market information to effectively handle the inherent nonstationarity of revenue patterns associated with data products. Furthermore, we leverage the transform domain of local model drifts in the global model update process, effectively reducing the communication burden and significantly improving training efficiency. Through simulations, we provide compelling evidence that our proposed schemes deliver superior performance across multiple evaluation metrics, including test loss, cumulative return, portfolio risk, and Sharpe ratio.**

***Index Terms*— Autonomous economic agents (AEAs), data trading, discrete wavelet transformation (DWT), federated learning (FL), histogram of oriented gradient (HoG), transform domain.**

I. Introduction

With the rapid advancement of artificial intelligence (AI) techniques and the exponential increase in processing power, our society is moving toward a future of ubiquitous intelligence, where data plays a crucial role in fueling AI applications [1], [2]. To ensure fair and efficient utilization of data resources, the establishment of data trading markets is imperative [3]. However, the ownership and revenue generation of data resources are highly dynamic, making it exceedingly difficult, if not impossible, for owners to make trading decisions [4]. Therefore, the expectation is that intelligent devices will autonomously process information and make profitable trading decisions [5], [6].

Self-governing trading technologies are at the forefront of the future data-centric society [7], [8]. The vision is for intelligent devices to autonomously process information and make profitable decisions, ultimately contributing to the development of a new intelligent economy [6]. Within a local data trading market, these intelligent devices, known as autonomous economic agents (AEAs), have the ability to monetize the data products they possess [9], [10]. AEAs acquire real-time raw data from data owners by paying a fixed fee and then process this data to create different types of data products, which they can sell to various companies to support their applications. The revenue generated by AEAs is based on the quality and utility of the data products they offer to these companies.

AEAs play a crucial role as intelligent and autonomous data suppliers for model companies, enabling the development of advanced applications. However, the research on data trading with AEAs is still in its early stages, presenting significant challenges. These challenges include defining the data trading market and, more importantly, effectively training AEAs to make intelligent and autonomous trading decisions. One challenge is the evaluation of data product revenue, considering their varying utilities across different applications and over time. This requires a robust approach to accurately assess the value of data products. Confidentiality is another concern in data trading. An individual AEA may have limited historical information to evaluate data, which is insufficient for training a reliable portfolio allocation model. Furthermore, real-world data market information is often nonstationary, with a low signal-to-noise ratio (SNR), due to dynamic demand for data products from various companies.

This article aims to address the aforementioned challenges and makes the following contributions. First, we propose an intelligent economic framework that integrates many AEAs to automatically fetch and process local data from individual owners and gain profit by selling the generated data products to others. Second, we design a learning objective for each AEA to train it to make promising trading decisions intelligently and autonomously in real-time. The raw observation of the market information by each AEA is its own trading experience in a specific region with possibly a different time scale. However, the knowledge of its own trading experiences may not be

Manuscript received 10 January 2023; revised 25 June 2023; accepted 8 November 2023. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and Compute Canada. *(Corresponding author: Lin Cai.)*

The authors are with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: cai@ece.uvic.ca).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2023.3332315.

Digital Object Identifier 10.1109/TNNLS.2023.3332315

sufficient to train the model that can perform data trading intelligently and autonomously. To achieve this goal, we design an efficient method to generate a learning objective for intelligent and autonomous data trading which can readily be adapted into highly nonstationary revenue data. Third, we introduce federated learning (FL) to provide a suitable framework for portfolio allocation model training. With the FL framework, AEAs can jointly train a promising model without revealing private trading information. To tackle the challenges that the market information from different AEAs may have a low SNR, high redundancy, and possibly strong feature correlation [11], we integrate the histogram of oriented gradients (HoGs) and discrete wavelet transformation (DWT) into the federated stochastic variance reduced gradient (FSVRG) method. The developed techniques are robust in handling and effective in extracting critical features of the nonstationary low-SNR revenue data. Furthermore, we apply the transform domain of the local model drifts in global model update at each federated training round which can greatly reduce the communication burden among AEAs and improve training efficiency.

The rest of this article is organized as follows. Section II presents the background and related works. Section III describes the objective design for AEA. Then, we introduce the federated optimization framework in Section IV. To improve the efficiency of the federated training procedure among AEAs, transform domain, and feature extraction-enabled FSVRG methods are proposed in Section V. Simulation results are presented in Section VI to illustrate the performance with different metrics, followed by concluding remarks and further research issues in Section VII.

II. Background and Related Works

The recent literature has underscored the importance of creating a data trading market, which stems from the notion of granting individuals ownership rights over their personal data resources [12], [13]. This ownership enables individuals to monetize their personal data, fostering a more equitable data ecosystem [14], [15]. For instance, autonomous vehicles can autonomously trade their sensor data based on the owner's interests, eliminating the need for frequent human interactions in the trading process [16]. The effectiveness of data trading relies on providing owners with sufficient incentives to exercise their ownership rights while ensuring a simple and streamlined trading procedure.

However, the dynamic nature of revenue returns in data trading poses significant challenges, making trading decisions complex and intricate [17], [18]. To address this challenge, we propose the creation of a data trading market facilitated by AEAs. These AEAs assume the responsibility of handling the complexities associated with exercising ownership rights over individual data resources. In this envisioned market, individual owners no longer need to directly contend with the highly dynamic revenue returns. Instead, the risks and uncertainties are assumed by the AEAs, which pay a constant fee to the individual owners. By shifting the burden of risk onto the AEAs and providing convenience and certainty to the owners, our proposed data trading market encourages and empowers owners to exercise their ownership rights more willingly. This framework promotes greater data liquidity, and ultimately, a more equitable and efficient data economy [19].

Our work stands out from the existing research that applied machine learning to financial applications with two major differences. First, most literature in this field typically analyzes the entire time span, combining training and testing data, where the financial data is nonconfidential [18], [20]. In contrast, our work involves local datasets derived from the private experiences of AEAs, making the data confidential. In addition, the experiences for each data product are segmented as AEAs engage in trading activities across different regions and time periods. This unique characteristic necessitates a different approach to handling and analyzing the data. Second, the existing machine-learning applications in the financial domain primarily focus on forecasting future returns, volumes, volatilities, and other essential quantities [21], [22]. However, our work diverges from this trend as we train AEAs based on separate local datasets to effectively generate portfolio allocations for multiple data products. Rather than solely focusing on forecasting specific financial metrics, our objective is to equip AEAs with the capability to automatically identify near-optimal portfolio allocations with multistep-ahead predictions. This approach broadens the scope of machine-learning applications in the financial domain, addressing the specific challenges and requirements of data trading and portfolio allocation in a dynamic and confidential environment.

In the FL training framework, individual machines perform multiple local model updates before communicating with the central server [23] to address communication cost and delay [24]. The popular federated averaging (FedAvg) algorithm, introduced by [25], achieves good empirical results by averaging local SGD updates assuming identically distributed local datasets. This approach relies on the assumption that the local gradient provides an unbiased estimate of the global gradient. Data heterogeneity significantly degrades the performance of FedAvg since the local stochastic gradient can no longer be considered an unbiased estimator of the global gradient in the presence of local data heterogeneity [26].

To address the challenges posed by data heterogeneity in FL, one approach is to utilize the FedProx algorithm [27], which limits the distance between the local model and the global model by adding a proximal term to the local objective functions. However, tuning the hyperparameter controlling the proximal term is crucial for achieving good performance. Another promising direction to tackle data heterogeneity in FL is the application of variance reduction techniques [28]. The SCAFFOLD algorithm proposed by [29] introduces control variates for the server and individual machines where the local updates are corrected by incorporating the difference between these variates. A more stable approach is FSVRG [30], derived from the stochastic variance reduced gradient (SVRG) [31], [32], which can be seen as an improved version of the distributed optimization algorithm DANE [33].

In our work, we develop a framework for a data trading market and enable intelligent and autonomous data trading among AEAs. Our federated training objective considers optimal trading decisions for multiple data products. To improve the training process, we integrate HoGs and DWT into the FSVRG method, known as HoG-enabled

TABLE I
PROS AND CONS OF FL ALGORITHMS

Methods	Pros	Cons
FedAvg	Simple to implement	Difficult to handle heterogeneous local data sets
FedProx	Slightly modify FedAvg and easy to implement	Need to optimize the hyper-parameter
FSVRG	Stable performance with heterogeneous local data sets	High communication and computation overhead
SCAFFOLD	Simplify FSVRG	Local updating could be unstable
HFSVRG	Robust and Fast Convergence	Relative weaker performance compared with TDHW-FSVRG
TDHW-FSVRG	Improved HFSVRG	Relative slower in convergence compared with HFSVRG

FSVRG (HFSVRG), and transform domain and HoG-wavelet enabled FSVRG (TDHW-FSVRG), where TDHW-FSVRG also employ transform-domain analysis of local model drifts during global model updates, reducing communication burdens and enhancing training efficiency among AEAs. The advantages and disadvantages of these FL algorithms are summarized in Table I.

III. OBJECTIVE IN DESIGN LOCAL DATASETS FOR AEAS

In this section, we begin by explaining the local data trading market with AEAs. We then outline the two main goals that AEAs aim to achieve: maximizing the return rate and controlling the risk of revenue returns.

A. Local Data Trading Market With AEAs

The AEA interacts with individual data owners by offering them a fixed fee in exchange for specific types and quantities of information. It then processes this data to create a variety of data products. These data products are traded with companies to enhance various applications. The pricing of these data products is determined through collaboration between the AEA and the companies, considering the value they bring to the services. To meet the dynamic market demands, the AEA must make intelligent decisions regarding data acquisition and processing, aiming to maximize long-term profitability.

In the data trading market, we assume that there are K AEAs and n different types of data products being traded. While the data product types are the same for all AEAs, each AEA has its unique trading experiences. The trading experience of the kth AEA encompasses the revenue returns of n data products over t_k time slots. We can represent the private sequential revenue returns of multiple data products for the kth AEA using the set $\{\beta^k_{i,j}\}_{i=1,\ldots,n;\, j=1,\ldots,t_k}$. Here, $\beta^k_{i,j}$ denotes the revenue return of data product i in the kth AEA at the jth time slot.

B. Rate of Return in Design of Local Datasets

The main target of AEAs is to achieve a higher rate of returns, which represents the increasing rate of the revenue return brought by trading the corresponding data products. We define a_{ij} as the ratio of the revenue return of data product i at time slot j and that of time slot $j-1$ as $a_{ij} = \beta^k_{i,j}/\beta^k_{i,j-1}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, t_k$. The rate of returns is computed from the recorded trading experience of the kth AEA. We use b_{ij} to denote the rate of return of data product i at time slot j, which is evaluated by

$$b_{ij} = \frac{\left(\beta^k_{i,j} - \beta^k_{i,j-1}\right)}{\beta^k_{i,j-1}} = a_{ij} - 1 \tag{1}$$

for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, t_k$.

The kth AEA generates the rate of return data for n different data products based on the observed price data. The trading experience of the kth AEA encompasses n types of data products over t_k time slots, resulting in t_k rate of returns for each data product. To generate the samples used in the local training procedure, the kth AEA follows a moving window scheme along its observed time slots. Two windows, denoted as $\{\boldsymbol{X}^k_p\}_{p=1,\ldots,P_k}$ and $\{\boldsymbol{B}^k_p\}_{p=1,\ldots,P_k}$, are designed for this purpose. Each sample $\boldsymbol{X}^k_p$ captures the rate of return data for n data products over l time slots, resulting in a window size of $n \times l$. Similarly, the window used to generate $\{\boldsymbol{B}^k_p\}_{p=1,\ldots,P_k}$ has a size of $n \times m$, incorporating future rate of return data for m time slots of n data products to guide portfolio prediction. A gap denoted by ν is introduced between these two moving windows to account for information from the immediate future that cannot be used for investment decisions. The number of local samples for the kth AEA can be calculated as $P_k = t_k - l - m - \nu + 1$.

During the trading period of the kth AEA, which lasts for t_k time slots, the demand for different data products varies greatly. The kth AEA must adjust its resource fetching and processing strategy frequently based on recent revenue returns to meet the changing demand. Each trading decision has a validity period of m to ensure a stable supply of data products for a certain time, supporting application development. Within the validity period of the pth decision, the rates of return for n data products are represented by the block matrix $\boldsymbol{B}^k_p \in R^{n\times m}$. The kth AEA's local dataset, derived from its trading experience, is denoted by $\{(\boldsymbol{X}^k_p, \boldsymbol{B}^k_p)\}_{p=1,\ldots,P_k}$. For each sample pair, the kth AEA makes trading decisions based on the rate of return features observed in $\boldsymbol{X}^k_p$. The goal of these decisions is to maximize the revenue return over m time slots and n data products ahead of the observation, utilizing the rate of returns provided in $\boldsymbol{B}^k_p$.

In the pth trading decision-making procedure of the kth AEA, we denote the portfolio value over n data products as α_0 at the starting time of $\boldsymbol{B}^k_p$, that is, the initial budget to invest in the period of m time slots. The kth AEA maintains a portfolio allocation weight variables $\boldsymbol{\theta}_{p,k} \in R^n$ for $\boldsymbol{B}^k_p$ to allocate the initial investment budget α_0 into n data products and α^i_0 is the amount of budget invested into the data product i at the beginning of the first time slot where $\alpha^i_0 = \alpha_0\theta^i_{p,k}$, $i = 1, 2, \ldots, n$. At the end of time slot $j-1$, the total value of this portfolio is $\alpha_{j-1} = \sum_{i=1}^n \alpha^i_{j-1}$. The value of data product i at the end of time slot j is updated as $\alpha^i_j = \alpha_{j-1}a_{ij}\theta^i_{p,k}$. With consistency, we set $a_{ij} = 0$ when there is no budget invested into data product i.

At the beginning of each time slot, the kth AEA calculates the current total portfolio value, which represents the revenue rate of each data product. This total value serves as the

investment budget for the next time slot, determining the allocation of resources for fetching and processing. At the end of time slot $(j-1)$, the kth AEA evaluates the total portfolio value, denoted as α_{j-1}, and reallocates this value across n data products using the portfolio allocation weight $\boldsymbol{\theta_{p,k}}$. The investment in each data product at the beginning of time slot j is derived from the total portfolio value α_{j-1} obtained at the end of time slot $(j-1)$. The overall portfolio value at the end of each time slot changes based on the varying rate of returns. The total value of the portfolio at the end of time slot j, denoted as α_j, accounts for the reinvestment of the total portfolio value, incorporating any gains or losses from previous periods, which can be obtained by

$$\alpha_j = \alpha_0 \prod_{i=1}^{j} \left(1 + \boldsymbol{b_i^T \theta_{p,k}}\right), \quad \text{for } j = 1, 2, \ldots, m \tag{2}$$

where $\boldsymbol{b_i}$ is the ith column of $\boldsymbol{B_p^k}$. We define the weighted mean of the rate of returns over the n data products in time slot j as $r_j = \boldsymbol{b_j^T \theta_{p,k}}$ for $j = 1, 2, \ldots, m$. When $r_i \ll 1$, $i = 1, 2, \ldots, j$, and the number of time slots j is small, the total value of the portfolio at the end of time slot j, as shown in (2) can be approximated by

$$\alpha_j \approx \alpha_0 \left(1 + \sum_{i=1}^{j} r_i\right), \quad \text{for } j = 1, 2, \ldots, m. \tag{3}$$

Obviously, with the input sample $\boldsymbol{X_p^k}$, the first objective of the kth AEA is to maximize the cumulative portfolio value over $\boldsymbol{B_p^k}$, that is, the cumulative portfolio value at the last time slot α_m. The objective can be transformed to maximize the average of the weighted mean of the rate of returns for n data products over m time slots, that is, $\bar{r}$, which can be evaluated by $\bar{r} = \boldsymbol{\mu^T \theta_{p,k}}$ where $\boldsymbol{\mu}$ denotes the average rate of return vector over m time slots defined as

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{j=1}^{m} \boldsymbol{b_j}. \tag{4}$$

The cumulative portfolio value at the mth time slot with respect to $\boldsymbol{B_p^k}$ can be evaluated by $\alpha_m \approx \alpha_0(1 + m\bar{r})$. To maximize α_m can be transformed to maximize $\bar{r}$ since the initial investment budget α_0 can be regarded as a constant.

C. Risk of Revenue Return in Design of the Local Dataset

In data trading, risk is a significant factor alongside return. The fluctuation in revenue returns represents the risk associated with investing in specific data products. Consequently, when AEAs seek to determine the optimal portfolio allocation weights, they must strike a balance between maximizing the final portfolio value and minimizing investment risk.

The inherent risk of each data product can be described as the variance of the rate of returns for each data product over m time slots, that is, the variation of the rate of return $\boldsymbol{b_j}$ for $j = 1, 2, \ldots, m$ from the average rate of return vector $\boldsymbol{\mu}$. The investment risk refers to the variation of the weighted mean rate of returns r_j for $j = 1, 2, \ldots, m$ from their average $\bar{r}$ over m time slots. We use Euclidean distance to measure the overall investment risk

$$\sum_{j=1}^{m} (r_j - \bar{r})^2 = \|\boldsymbol{d}\|_2^2 \tag{5}$$

where $d_j = (\boldsymbol{b_j} - \boldsymbol{\mu})^T \boldsymbol{\theta_{p,k}}$ is the jth element in the variation vector $\boldsymbol{d}$. It is obvious that the risk of the rate of return is directly connected with the matrix defined as $[(\boldsymbol{b_1} - \boldsymbol{\mu}), \ldots, (\boldsymbol{b_m} - \boldsymbol{\mu})]$. Then, the investment risk can be evaluated by a matrix

$$\boldsymbol{C} = \frac{1}{m} \sum_{j=1}^{m} (\boldsymbol{b_j} - \boldsymbol{\mu})(\boldsymbol{b_j} - \boldsymbol{\mu})^T \tag{6}$$

where the element c_{kl} is related to the covariance of the rate of return for data product k and data product l over m time slots.

TABLE II
SUMMARY OF NOTATIONS IN THE SYSTEM MODEL AND PERFORMANCE METRICS

Notation	Definition
K	Number of AEAs
n	Number of different types data products
t_k	The trading experience time span for the k-th AEA
$\beta_{i,j}^k$	The revenue return of data product i at time slot j of the k-th AEA
a_{ij}	The ratio of the revenue return of data product i at time slot j and that of time slot $j-1$
b_{ij}	the rate of return of data product i at time slot j
m	The valid time period for one trading decision
$\boldsymbol{X_p^k}$	The p-th rate of return sample matrix in the k-th AEA
$\boldsymbol{B_p^k}$	The rate of return block used to design label for $\boldsymbol{X_p^k}$
α_0	The initial investment budget
$\boldsymbol{\theta_{p,k}}$	The portfolio allocation weight variables for $\boldsymbol{B_p^k}$
α_j	The total portfolio value at the end of time slot j
r_j	The weighted mean of the rate of returns over the n data products in time slot j
$\bar{r}$	The average of the weighted mean of the rate of returns for n data products over m time slots
$\boldsymbol{\mu}$	The average rate of return vector over m time slots
$\boldsymbol{C}$	The covariance matrix of the rate of return over n data products
$\tilde{\boldsymbol{\theta}}_{\boldsymbol{p,k}}$	The predicted portfolio allocation for $\boldsymbol{X_p^k}$
$\boldsymbol{\theta^*_{p,k}}$	The designed label for $\boldsymbol{X_p^k}$
P_k	The number of local samples with the k-th AEA

IV. Federated Optimization Framework

Due to the privacy issue of the application development information in companies, the local trading experience data from AEAs cannot be gathered together to train one global model. Therefore, we propose a federated optimization framework to enable individual AEAs to collaboratively train their models which will be more robust compared with their local optimized solution. In Section IV-A, we formulate the local model to predict the portfolio allocation vector and design the supervised label for each sample in detail in Section IV-B. Furthermore, we also introduce the local and global training objectives in Section IV-C. All the notations are summarized in Table II.

A. Local Model Design for AEAs

To generate more promising portfolio allocations, AEAs collaborate with each other to train their models. Each AEA trains its own local model using its private dataset, derived

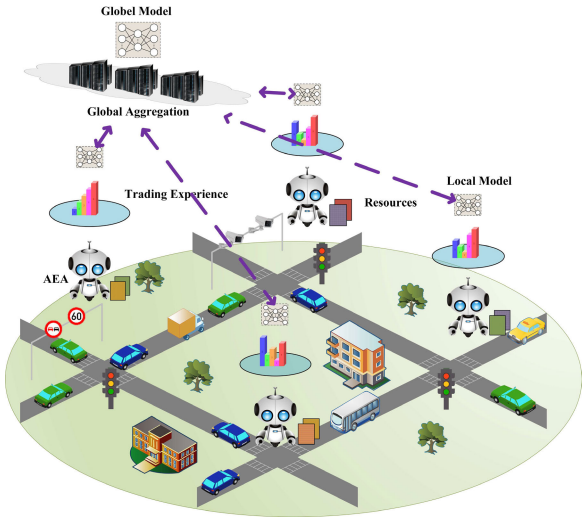


Fig. 1. FL architecture in data trading markets.

from its individual trading experience, and it can share knowledge and learn from other AEAs, as depicted in Fig. 1. Given the current observations of the rate of returns for the n data products in the market, denoted as $\boldsymbol{X}_p^k$, the decision made by the kth AEA is represented by the portfolio allocation vector $\tilde{\boldsymbol{\theta}}_{p,k} \in \mathbb{R}^n$. In other words, the model of the kth AEA should be capable of outputting the portfolio allocation vector to achieve promising revenue returns with respect to $\boldsymbol{B}_p^k$ based on the observation $\boldsymbol{X}_p^k$.

The model design for each AEA takes into account two properties of the portfolio allocation vector, that is, $0 \leq \tilde{\theta}_{p,k}^i \leq 1$ for $i = 1, 2, \ldots, n$, and $\sum_{i=1}^n \tilde{\theta}_{p,k}^i = 1$. The model parameters are decomposed into n submodels, that is, $\{\boldsymbol{w}_i\}_{i=1}^n$. We vectorize the parameters with respect to the kth AEA into one-parameter vector $\hat{\boldsymbol{w}}_k$, where $\hat{\boldsymbol{w}}_k = [\boldsymbol{w}_1^{kT}, \ldots, \boldsymbol{w}_n^{kT}]^T$, where $\boldsymbol{w}_i^k$ represents a submodel used to predict the ith component in the optimal portfolio allocation vector $\tilde{\boldsymbol{\theta}}_{p,k}$, for $p = 1, 2, \ldots, P_k$. The portfolio allocation vector $\tilde{\boldsymbol{\theta}}_{p,k}$ is formulated as follows:

$$\tilde{\boldsymbol{\theta}}_{p,k} = \left\{ \frac{e^{\boldsymbol{w}_i^{kT} \boldsymbol{x}_p^k}}{\sum_{j=1}^n e^{\boldsymbol{w}_j^{kT} \boldsymbol{x}_p^k}} \right\}_{i=1,2,\ldots,n} \tag{7}$$

where $\boldsymbol{x}_p^k$ is the vectorization of $\boldsymbol{X}_p^k$.

B. Optimal Label Design

To perform data trading intelligently and autonomously based on a good model, we need to build up training targets as the optimal portfolio allocations with the multitime slots ahead of data. The optimal portfolio allocation weights for m time slots in $\boldsymbol{B}_p^k$ with sample $\boldsymbol{X}_p^k$ is denoted by $\boldsymbol{\theta}_{p,k}^*$. The decision made by AEA can be regarded as a prediction of $\boldsymbol{\theta}_{p,k}^*$. The training procedure tries to minimize the difference between the decision made by AEA, that is, $\tilde{\boldsymbol{\theta}}_{p,k}$ and the optimal portfolio allocation $\boldsymbol{\theta}_{p,k}^*$ with respect to $\boldsymbol{B}_p^k$. In the supervised learning perspective, the corresponding label can be mapped into the optimal portfolio allocation weights $\boldsymbol{\theta}_{p,k}^*$. Therefore, the P_k samples in the local dataset of the kth AEA can be redesigned as $\{(\boldsymbol{X}_p^k, \boldsymbol{\theta}_{p,k}^*)\}_{p=1,\ldots,P_k}$.

For the given observation of the rate of return block $\boldsymbol{X}_p^k$, the designed label, that is, the optimal portfolio allocation $\boldsymbol{\theta}_{p,k}^*$ for the upcoming market scenario $\boldsymbol{B}_p^k$, should take a tradeoff between the portfolio return and the risk. For each sample pair in the local dataset, we integrate portfolio optimization to design the learning objective for portfolio allocations. We apply the mean-variance minimization based on $\boldsymbol{B}_p^k$ to design the training label for $\boldsymbol{X}_p^k$. The learning objective of the kth AEA can be designed by integrating maximizing the total portfolio value and minimizing the investment risk with all of the current budget invested into n data products. We define $\boldsymbol{A} = \boldsymbol{1}^T$ and $\boldsymbol{b} = [1]$, generating the learning label for observation $\boldsymbol{X}_p^k$ is formulated as the following constrained optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta}_{p,k}} \frac{1}{2} \boldsymbol{\theta}_{p,k}^T \boldsymbol{C} \boldsymbol{\theta}_{p,k} - \lambda \boldsymbol{\theta}_{p,k}^T \boldsymbol{\mu} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{\theta}_{p,k} = \boldsymbol{b} \end{aligned} \tag{8}$$

where λ is a positive parameter indicating the tradeoff between risk and revenue return. For each new market observation pair, the AEA needs to solve the above question to generate the corresponding label which is used to guide the local training of the model. It is critical to provide a more economical approach to generate the label for the given market observation data $\boldsymbol{X}_p^k$.

As designed in (6), $\boldsymbol{C}$ is symmetric and positive semidefinite. Because the constraints are linear equations, it gives us the idea that we can search for the optimal portfolio allocation $\boldsymbol{\theta}_{p,k}^*$ from all the solutions defined by the constraint equations. We can transform the portfolio allocation vector $\boldsymbol{\theta}_{p,k}$ to another vector variable $\hat{\boldsymbol{\varphi}}_{p,k} \in R^{n \times 1}$ and then we can easily reduce several elements in $\hat{\boldsymbol{\varphi}}_{p,k}$ and only focus on a variable $\boldsymbol{\varphi}_{p,k} \in R^{(n-1)\times 1}$ with smaller dimensions. To get the solutions of $\boldsymbol{A}\boldsymbol{\theta}_{p,k} = \boldsymbol{b}$, we need to get the Moore–Penrose pseudo-inverse of $\boldsymbol{A}$ which is denoted by

$$\boldsymbol{A}^+ = \boldsymbol{A}^T (\boldsymbol{A}\boldsymbol{A}^T)^{-1}. \tag{9}$$

All the feasible points defined by the linear equality constraint can be characterized by

$$\boldsymbol{\theta}_{p,k} = \boldsymbol{A}^+ \boldsymbol{b} + (\boldsymbol{I}_n - \boldsymbol{A}^+ \boldsymbol{A}) \hat{\boldsymbol{\varphi}}_{p,k}. \tag{10}$$

Then, based on the SVD of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, we can continue to transform $\hat{\boldsymbol{\varphi}}_{p,k}$ to a lower-dimensional variable $\boldsymbol{\varphi}_{p,k}$, where $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{0} \end{bmatrix}$ and $\boldsymbol{S} = \text{diag}\{\sigma_1\}$. The pseudo-inverse $\boldsymbol{A}^+$ can be denoted by

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^T (\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T)^{-1} \boldsymbol{U}^T = \boldsymbol{V} \begin{bmatrix} \boldsymbol{S}^{-1} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{U}^T.$$

Thus,

$$\boldsymbol{I}_n - \boldsymbol{A}^+ \boldsymbol{A} = \boldsymbol{I}_n - \boldsymbol{V} \begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{V}^T = \sum_{i=2}^n \boldsymbol{v}_i \boldsymbol{v}_i^T = \boldsymbol{V}_r \boldsymbol{V}_r^T$$

where $\boldsymbol{V}_r = [\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_n]$. All feasible regions defined by the linear equality constraints becomes

$$\boldsymbol{\theta}_{p,k} = \boldsymbol{A}^+ \boldsymbol{b} + \boldsymbol{V}_r \boldsymbol{V}_r^T \hat{\boldsymbol{\varphi}}_{p,k} = \boldsymbol{\theta}_s + \boldsymbol{V}_r \boldsymbol{\varphi}_{p,k} \tag{11}$$

where we define $\boldsymbol{\varphi}_{p,k} = \boldsymbol{V}_r^T \hat{\boldsymbol{\varphi}}_{p,k}$ and $\boldsymbol{\theta}_s = \boldsymbol{A}^+ \boldsymbol{b}$. We put $\boldsymbol{\theta}_{p,k} = \boldsymbol{\theta}_s + \boldsymbol{V}_r \boldsymbol{\varphi}_{p,k}$ back to the original objective function,

the problem becomes

$$\min \frac{1}{2}\boldsymbol{\varphi}_{p,k}^{T}\hat{\boldsymbol{H}}\boldsymbol{\varphi}_{p,k}+\boldsymbol{\varphi}_{p,k}^{T}\hat{\boldsymbol{p}}+\hat{k}$$

where $\hat{\boldsymbol{H}}=\boldsymbol{V}_r^T\boldsymbol{C}\boldsymbol{V}_r$, $\hat{\boldsymbol{p}}=\boldsymbol{V}_r^T\boldsymbol{C}\boldsymbol{\theta}_s-\boldsymbol{\lambda}\boldsymbol{V}_r^T\boldsymbol{\mu}$, and $\hat{k}=(1/2)\boldsymbol{\theta}_s^T\boldsymbol{C}\boldsymbol{\theta}_s-\lambda\boldsymbol{\mu}^T\boldsymbol{\theta}_s$. Because $\boldsymbol{V}_r\in R^{n\times(n-1)}$, thus $\hat{\boldsymbol{H}}\in R^{(n-1)\times(n-1)}$, we shrink the corresponding Hessian matrix after transforming $\boldsymbol{\theta}_{p,k}$ to $\boldsymbol{\varphi}_{p,k}$. The problem becomes an unconstrained problem with respect to $\boldsymbol{\varphi}_{p,k}$, to get the optimal solution, we need to compute the gradient and set it to $\mathbf{0}$, that is, $\hat{\boldsymbol{H}}\boldsymbol{\varphi}_{p,k}+\hat{\boldsymbol{p}}=\mathbf{0}$. If $\boldsymbol{C}$ is P.D., then $\hat{\boldsymbol{H}}$ is invertible, then the label for sample $\boldsymbol{X}_p^k$ is designed as the unique global solution, that is,

$$\boldsymbol{\theta}_{p,k}^{*}=\boldsymbol{\theta}_s+\boldsymbol{V}_r\boldsymbol{\varphi}_{p,k}^{*}=\boldsymbol{A}^{+}\boldsymbol{b}+\boldsymbol{V}_r\boldsymbol{\varphi}_{p,k}^{*} \tag{12}$$

where $\boldsymbol{\varphi}_{p,k}^{*}=-\hat{\boldsymbol{H}}^{-1}\hat{\boldsymbol{p}}$.

C. Local and Global Learning Objectives

Based on the designed optimal label for sample $\boldsymbol{X}_p^k$ in the kth AEA as $\boldsymbol{\theta}_{p,k}^{*}$ shown in (12), the local model will be directly trained to generate promising portfolio allocation weights for the current budget to invest in the data resource collection and processing for the next time slot supply to companies. From the local model design, the predicted portfolio allocation vector is defined by $\tilde{\boldsymbol{\theta}}_{p,k}$ as shown in (7). We use least-square to formulate the local objective function for the kth AEA with P_k local samples which are defined as

$$f_k(\boldsymbol{w})=\frac{1}{P_k}\sum_{p=1}^{P_k}\left\|\tilde{\boldsymbol{\theta}}_{p,k}-\boldsymbol{\theta}_{p,k}^{*}\right\|_2^2. \tag{13}$$

With the participation of K AEAs in the federated optimization framework, the initialized model parameter $\boldsymbol{w}^{\mathbf{0}}$ is distributed to the kth AEA for $k=1,2,\ldots,K$ at $t=0$ to be its initialed local model $\hat{\boldsymbol{w}}_k$. Then, each AEA trains its own local model based on the local dataset within a limited time. The local training procedure tunes the local model parameters by minimizing the local objective function in (13) for several updating steps following the guidance of some optimizers. After that, AEAs will transmit their local model parameters and participate in the model parameter aggregation. The aggregated model will be redistributed to all AEAs to perform the local training in the next round. The final goal of federated optimization is to obtain a global model that takes advantage of all the local knowledge and can outperform locally trained models. The final trained global model targets to minimize the global objective function over K AEAs formulated as

$$f(\boldsymbol{w})=\sum_{k=1}^{K}\frac{P_k}{P}f_k(\hat{\boldsymbol{w}}_k),\quad \text{for } P=\sum_{k=1}^{K}P_k. \tag{14}$$

There are K subobjects in the overall objective function $f(\boldsymbol{w})$. The intuitive idea to minimize $f(\boldsymbol{w})$ is to minimize each subobjective function $f_k(\hat{\boldsymbol{w}}_k)$, for $k=1,2,\ldots,K$, and then combine the K separated trained models by averaging to be the solution for the federated optimization problem.

V. Feature Extraction and Transform-Domain Enabled FL

To address the challenge of a low SNR resulting from the high randomness of the data market and achieve rapid convergence, we propose two novel approaches. First, we introduce the HoG to redefine the representations of observed samples by capturing the changes in the rate of returns across different data products over adjacent time slots. This leads to the development of the HFSVRG algorithm. Second, we enhance the representation of the rate of returns by integrating HoG with wavelet transformation and incorporating transform-domain techniques in the global model update. This results in the TDHW-FSVRG algorithm, which significantly improves the training efficiency. These techniques enable us to effectively handle low SNR data and expedite the convergence process in the dynamic data trading market. The notations in the algorithms design are summarized in Table III.

A. HoG-Enabled FSVRG

The gradient information can be regarded as the learned knowledge directly from the corresponding data samples which guides how to tune the model parameters to minimize the objective function. In the local training procedure of the kth AEA with the observed pth local sample $\boldsymbol{X}_p^k$, the local model $\hat{\boldsymbol{w}}_k$ is trained to minimize the difference between the portfolio allocation vector $\tilde{\boldsymbol{\theta}}_p$ and $\boldsymbol{\theta}_p$. The portfolio allocation vector $\tilde{\boldsymbol{\theta}}_p$ is decided by the kth AEA which can be regarded as the estimation of the optimal portfolio allocation vector by the local model $\hat{\boldsymbol{w}}_k$. The optimal portfolio allocation vector $\boldsymbol{\theta}_p$ is an optimal solution to the formulated objective in (8) considering the rate of return block $\boldsymbol{B}_p^k$ ahead of $\boldsymbol{X}_p^k$. As shown in the local objective function (13), the difference between the estimation $\tilde{\boldsymbol{\theta}}_p$ and the optimal solution $\boldsymbol{\theta}_p$ is formulated by $||\tilde{\boldsymbol{\theta}}_p-\boldsymbol{\theta}_p||_2^2$. We apply the gradient information to tune the model parameters $\hat{\boldsymbol{w}}_k$ to achieve a promising portfolio estimation $\tilde{\boldsymbol{\theta}}_p$.

The local gradient is highly depended on the market data $\boldsymbol{X}_p^k$. To make the training procedure more efficient, we redesign much more meaningful representations of the market data by extracting return rate changes along the time line for each data product and return rate changes among different data products at the same time slot. The changes among the rate of returns can be quantified by a linear difference equation with given kernel vectors, $\boldsymbol{\kappa}_x=[\kappa_x^1,\ldots,\kappa_x^{N_1}]$ and $\boldsymbol{\kappa}_y=[\kappa_y^1,\ldots,\kappa_y^{N_2}]^T$, respectively, where N_1 and N_2 represent the size of the kernel vectors. We use $\boldsymbol{x}_i$ to denote the rate of return of data product i over l time slots, and $\boldsymbol{y}_j$ to the rate of return of n data products at the jth time slot in sample $\boldsymbol{X}_p$. The rate of return changes for data product i over l time slots is obtained by

$$\tilde{\boldsymbol{d}}_{\boldsymbol{x}}^{i}=\boldsymbol{x}_i\otimes\boldsymbol{\kappa}_x=\left\{\sum_{k=1}^{N_1}\kappa_x^k x_i^{j-k+1}\right\}_{j=1,\ldots,l} \tag{15}$$

where $\boldsymbol{x}_i$ is the extended row vector $\boldsymbol{X}_p^k(i,:)$, when the index is smaller than 1 or larger than l, the values are set to 0. The rate of return changes for n data products at the jth time slot

is obtained by

$$\tilde{\boldsymbol{d}}_{\boldsymbol{y}}^{\boldsymbol{j}} = \boldsymbol{y_j} \otimes \boldsymbol{\kappa_y} = \left\{ \sum_{k=1}^{N_2} h_y^k y_j^{i-k+1} \right\}_{i=1,\ldots,n}^{T} \tag{16}$$

where $\boldsymbol{y_j}$ could be regarded as the extended column vector of $\boldsymbol{X_p^k}(:, j)$, when the index is smaller than 1 or larger than n, the values are set to 0.

According to the formulation in (15) and (16), the computation results show that $\tilde{\boldsymbol{d}}_{\boldsymbol{x}}^{\boldsymbol{i}} \in R^{1\times(l+N_1-1)}$ and $\tilde{\boldsymbol{d}}_{\boldsymbol{y}}^{\boldsymbol{j}} \in R^{(n+N_2-1)\times 1}$. However, the rate of return changes among n different data products over l time slots for the given sample $\boldsymbol{X_p^k}$ is our target. Thus, we define the beginning index $s_x = \lfloor((N_1-1)/2)\rfloor$ for $\{\tilde{\boldsymbol{d}}_{\boldsymbol{x}}^{\boldsymbol{i}}\}_{i=1,\ldots,n}$ and $s_y = \lfloor((N_2-1)/2)\rfloor$ for $\{\tilde{\boldsymbol{d}}_{\boldsymbol{y}}^{\boldsymbol{j}}\}_{j=1,\ldots,l}$, respectively. Then, the oriented gradient with respect to the rate of returns in $\boldsymbol{X_p^k}$ is defined as

$$\left\{ \tilde{\boldsymbol{g}}_{i,j} = \begin{bmatrix} \tilde{\boldsymbol{d}}_{\boldsymbol{x}}^{\boldsymbol{i}}(s_x+i) \\ \tilde{\boldsymbol{d}}_{\boldsymbol{y}}^{\boldsymbol{j}}(s_y+j) \end{bmatrix} \right\}_{i=1,\ldots,l;\, j=1,\ldots,n.} \tag{17}$$

There are $n \times l$ oriented gradients recorded for sample $\boldsymbol{X_p^k}$. Because $\tilde{\boldsymbol{g}}_{i,j}$ consists of first-order changes of the rate of return along different data products at one time slot and one data product along different time slots, it captures the local information of a patch in terms of the intensity of the changes and changing directions. These properties can be found by computing the magnitude and angle of $\tilde{\boldsymbol{g}}_{i,j}$ denoted by $||\tilde{\boldsymbol{g}}_{i,j}||_2$ and $\rho_{i,j}$, for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, l$, respectively. Since we take the sign of local gradient $\tilde{\boldsymbol{g}}_{i,j}$ into account, the angle $\rho_{i,j}$ is in the range between $-\pi$ and π.

To summarize the intensive changes in the rate of returns of n data products over l time slots, we divide the $n \times l$ oriented gradients into a set of blocks. Each block represents the rate of return changing positions. We define the size of each block as $d_x \times d_y$ which means each block will focus on the rate of return changes within the local area covering d_y data products over d_x time slots. To divide $\boldsymbol{X_p^k}$ into multiple blocks, integer-valued parameters s_x and s_y known as strides are defined to indicate the number of dimensions moving along the time slots and different data products, respectively. When moving along the l time slots, the number of blocks generated can be denoted by $\lceil((l-d_x)/s_x)\rceil + 1$. Similarly, there are $\lceil((n-d_y)/s_y)\rceil + 1$ blocks when dividing $\boldsymbol{X_p^k}$ along the n data products dimension. Thus, there are $(\lceil((l-d_x)/s_x)\rceil + 1) \times (\lceil((n-d_y)/s_y)\rceil + 1)$ blocks in total. It is important to control the stride to ensure that adjacent blocks have appropriate overlaps. These overlaps can enable each HoG sample feature to contain the changing information among different data products over different time slots in multiple blocks which can make the HoG features much more robust to small local changes. Thus, when part of the block covers the area outside of $\boldsymbol{X_p^k}$, we simply use the nearby area to do the patch.

To generate HoG features in each block, we summarize the oriented gradients information located in the area covered by the block. We first evenly separate the oriented angle range from $-\pi$ to π into Ω bins and the value in each bin is initialized as 0. Then, we check the angle $\rho_{i,j}$ of the oriented gradients within the covering area of the block and add the magnitude $||\tilde{\boldsymbol{g}}_{i,j}||_2$ to the bin value where $\rho_{i,j}$ within its cover range. Thus, the HoG feature generated for each block can be represented by a Ω-dimensional feature vector whose ith component is the value contained in the ith bin.

TABLE III
SUMMARY OF NOTATIONS IN ALGORITHM DESIGN

Notation	Definition
$\boldsymbol{\kappa_x}$, $\boldsymbol{\kappa_y}$	The horizontal and vertical kernel vectors
$\tilde{\boldsymbol{g}}_{\boldsymbol{i,j}}$, $\rho_{i,j}$	The oriented gradient and its angle
$\hat{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}}$	HoG feature vector of $\boldsymbol{X_p^k}$
$\boldsymbol{g}_{\boldsymbol{i,p}}^{\boldsymbol{k}}$	The gradient of the i-th sub-model $\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}$ over sample $\hat{\boldsymbol{x}}_{\boldsymbol{p}}$
$\boldsymbol{\iota}$, $\boldsymbol{h}$	The coefficients vector of the low-pass and high-pass filters
$\boldsymbol{L}$, $\boldsymbol{H}$	Low-pass and high-pass filter operators
$\boldsymbol{M^J}$	Wavelet transformation matrix with J decomposition levels
$\boldsymbol{D_p^J}$	Wavelet coefficients of $\boldsymbol{X_p}$ with J decomposition levels
δ_p	The threshold for de-noising sample $\boldsymbol{X_p}$
σ^2	The estimate noise variance among features
$\tilde{\boldsymbol{X}}_{\boldsymbol{p}}$	The de-noised sample of $\boldsymbol{X_p}$
$\tilde{\boldsymbol{D}}_{\boldsymbol{p}}^{\boldsymbol{J}*}$	The wavelet coefficients of the de-noised sample $\tilde{\boldsymbol{X}}_{\boldsymbol{p}}$
$\tilde{\boldsymbol{T}}_{\boldsymbol{n}}^{\boldsymbol{J}}$, $\tilde{\boldsymbol{T}}_{\boldsymbol{l}}^{\boldsymbol{J}}$	Diagonal weight matrices re-scaling $\boldsymbol{M_n^J}$ and $\boldsymbol{M_l^J}$
$\tilde{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}}$	Wavelet-HoG feature vector of $\boldsymbol{X_p^k}$
$\hat{\boldsymbol{w}}_{\boldsymbol{k}}$	The local model of the k-th AEA
$\hat{\boldsymbol{w}}^{\boldsymbol{t}}$	The global model in round t
$\triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}}$	The k-th local model drift
$\hat{\boldsymbol{g}}_{\boldsymbol{k,p}}$	Gradient w.r.t. current global model
$\hat{\boldsymbol{g}}_{\boldsymbol{k,p}}^{\boldsymbol{t}}$	Gradient w.r.t. current local model
$\bar{\boldsymbol{g}}^{\boldsymbol{t}}$	The average global gradient at round t
$\hat{\mu}$	The quadratic perturbation control value
$\boldsymbol{u_i}$	The i-th frequency-related basis vectors
$\boldsymbol{z}^{\boldsymbol{k}}$	Frequency domain representation of the k-th local model
$\boldsymbol{z}_{\boldsymbol{\gamma}}^{\boldsymbol{k}}$	The compressed local model drift in frequency domain

Algorithm 1 HFSVRG

1: Initialize global model parameters $\hat{\boldsymbol{w}}^{\boldsymbol{0}}$
2: **for** $t \leftarrow 0$ to T **do**
3: Randomly sample a set of AEAs C_t
4: **for** $k \in C_t$ **in parallel do**
5: $\{\bar{\boldsymbol{g}}_{\boldsymbol{k}}, \hat{\boldsymbol{S}}_{\boldsymbol{k}}, \boldsymbol{G_k}\} \leftarrow$ **LGC-HFSVRG** (k, $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$)
6: **end for**
7: $\boldsymbol{|S| = \sum_{k\in C_t} |S_k|}$
8: $\bar{\boldsymbol{g}}^{\boldsymbol{t}} = \frac{1}{|S|}\sum_{k\in C_t} \bar{\boldsymbol{g}}_{\boldsymbol{k}}$
9: **for** $k \in C_t$ **in parallel do**
10: $\triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}} \leftarrow$ **LU-HFSVRG** ($\bar{\boldsymbol{g}}^{\boldsymbol{t}}$, $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$, $\hat{\boldsymbol{S}}_{\boldsymbol{k}}$, $\boldsymbol{G_k}$)
11: **end for**
12: $\hat{\boldsymbol{w}}^{\boldsymbol{t+1}} \leftarrow \hat{\boldsymbol{w}}^{\boldsymbol{t}} - \eta_{\text{global}} \sum_{k\in C_t} \triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}}$
13: **end for**

The FL procedure is shown in Algorithm 1 in detail. At the beginning of the tth training round, the global model parameter $\boldsymbol{w}^{\boldsymbol{t}}$ is distributed into the participated AEAs in set C_t. Then, in the parallel local training procedure, each AEA in C_t conducts Procedure 1 to collect the local gradient information. In Procedure 1, we randomly select a local sample set $\boldsymbol{S_k}$ and conduct HoG feature extraction over sample $\boldsymbol{X_p^k} \in \boldsymbol{S_k}$ in the above procedures and stack the Ω HoG features of each block from sample $\boldsymbol{X_p^k}$ into one vector $\hat{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}}$ as shown in Line 4 in Procedure 1.

The gradient of the ith submodel $\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}$ over $\hat{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}}$, that is, $\nabla_{\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}} f_k(\hat{\boldsymbol{w}}_{\boldsymbol{k}}, \hat{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}})$, is computed as

$$\nabla_{\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}} f_k(\hat{\boldsymbol{w}}_{\boldsymbol{k}}, \hat{\boldsymbol{x}}_{\boldsymbol{p}}^{\boldsymbol{k}}) = \nabla_{\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}} ||\tilde{\boldsymbol{\theta}}_{\boldsymbol{p}} - \boldsymbol{\theta}_{\boldsymbol{p}}||_2^2 = 2(\tilde{\boldsymbol{\theta}}_{\boldsymbol{p}} - \boldsymbol{\theta}_{\boldsymbol{p}}) \nabla_{\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}} \tilde{\boldsymbol{\theta}}_{\boldsymbol{p}}.$$

Procedure 1 Local-Gradient-Collection-HFSVRG (k, $\hat{\boldsymbol{w}}^t$)

1: Randomly select sample set $\boldsymbol{S_k}$
2: Initialize $\bar{\boldsymbol{g}}_k = 0$, $\hat{\boldsymbol{S}}_k \leftarrow \varnothing$ and $\boldsymbol{G_k} \leftarrow \varnothing$
3: **for** $\boldsymbol{X}_p^k \in \boldsymbol{S_k}$ **do**
4: $\quad \hat{\boldsymbol{x}}_p^k \leftarrow \text{HoG}(\boldsymbol{X}_p^k)$
5: $\quad$ Evaluate $\tilde{\boldsymbol{\theta}}_p$ by $\hat{\boldsymbol{w}}^t$ and $\hat{\boldsymbol{x}}_p^k$ in Eq. (7)
6: $\quad \boldsymbol{g}_{i,p}^k = 2(\tilde{\theta}_p^i - \theta_p^i)(\tilde{\theta}_p^i - (\tilde{\theta}_p^i)^2)\hat{\boldsymbol{x}}_p^k$ for $i = 1, \cdots, n$
7: $\quad \hat{\boldsymbol{g}}_{k,p} = [\boldsymbol{g}_{1,p}^{kT}, \cdots, \boldsymbol{g}_{n,p}^{kT}]^T$
8: $\quad \bar{\boldsymbol{g}}_k \leftarrow \bar{\boldsymbol{g}}_k + \hat{\boldsymbol{g}}_{k,p}$, $\hat{\boldsymbol{S}}_k \leftarrow \hat{\boldsymbol{S}}_k \cup \hat{\boldsymbol{x}}_p^k$, $\boldsymbol{G_k} \leftarrow \boldsymbol{G_k} \cup \hat{\boldsymbol{g}}_{k,p}$
9: **end for**
10: $\bar{\boldsymbol{g}}_k \leftarrow \frac{\bar{\boldsymbol{g}}_k}{|\boldsymbol{S_k}|}$
11: Return $\{\bar{\boldsymbol{g}}_k, \hat{\boldsymbol{S}}_k, \boldsymbol{G_k}\}$

From the formulation of the resource fetching decision $\tilde{\boldsymbol{\theta}}_p$ in (7), we can obtain that

$$\nabla_{\boldsymbol{w}_i^k} \tilde{\boldsymbol{\theta}}_p = \frac{\left(\sum_{j=1}^n e^{\boldsymbol{w}_j^{kT}\hat{\boldsymbol{x}}_p^k} - e^{\boldsymbol{w}_i^{kT}\hat{\boldsymbol{x}}_p^k}\right) e^{\boldsymbol{w}_i^{kT}\hat{\boldsymbol{x}}_p^k}}{\left(\sum_{j=1}^n e^{\boldsymbol{w}_j^{kT}\hat{\boldsymbol{x}}_p^k}\right)^2} \hat{\boldsymbol{x}}_p^k. \tag{18}$$

We define $\boldsymbol{g}_{i,p}^k = \nabla_{\boldsymbol{w}_i^k} f_k(\hat{\boldsymbol{w}}_k, \hat{\boldsymbol{x}}_p^k)$ for notation convenience as shown in Line 6 in Procedure 1. We use $\hat{\boldsymbol{g}}_{k,p}$ to represent the gradient of the local objective $f_k(\hat{\boldsymbol{w}}_k)$ over sample $\hat{\boldsymbol{x}}_p^k$ with the vectorized model parameter $\hat{\boldsymbol{w}}_k$, which is constructed by stacking all gradients with respect to submodels into one vector, that is, $\hat{\boldsymbol{g}}_{k,p} = [\boldsymbol{g}_{1,p}^{kT}, \ldots, \boldsymbol{g}_{n,p}^{kT}]^T$ as shown in Line 7 in Procedure 1.

To introduce the information from other AEAs into the local training of the kth AEA, we need to combine the local gradients together since they represent the local knowledge learned in the current round. In the local training procedure of the kth AEA, we randomly selected a subset $\boldsymbol{S_k}$ to evaluate the averaged gradient with respect to model parameter vector $\hat{\boldsymbol{w}}_k$ denoted by $\bar{\boldsymbol{g}}_k$ as shown in Line 10 in Procedure 1. The kth AEA, where $k \in C_t$, transmits $\bar{\boldsymbol{g}}_k, \hat{\boldsymbol{S}}_k, \boldsymbol{G_k}$ to the central server, where $\hat{\boldsymbol{S}}_k, \boldsymbol{G_k}$ collect the HoG features of the selected local samples and the corresponding gradients with respect to the $\boldsymbol{w}^t$, respectively. The central server can obtain the averaged global gradient at round t denoted by $\bar{\boldsymbol{g}}^t$.

Then, $\bar{\boldsymbol{g}}^t, \hat{\boldsymbol{w}}_t, \hat{\boldsymbol{S}}_k, \boldsymbol{G_k}$ are distributed to the kth AEAs, where $k \in C_t$, for the local updating in Procedure 2. In Procedure 2, the kth AEA performs $|\boldsymbol{S_k}|$ fast stochastic updates. In each stochastic update, we apply local step size η_{local} to update $\hat{\boldsymbol{w}}_k$. To ensure convergence and introduce the information from other AEAs to design the local update direction, we redesign the local objective function as

$$\tilde{f}_k(\hat{\boldsymbol{w}}_k) = f_k(\hat{\boldsymbol{w}}_k) + (\bar{\boldsymbol{g}}^t - \hat{\boldsymbol{g}}_k^t)^T \hat{\boldsymbol{w}}_k + \frac{\hat{\mu}}{2}||\hat{\boldsymbol{w}}_k - \hat{\boldsymbol{w}}^t||^2$$

where $\hat{\boldsymbol{g}}_k^t$ denotes the gradient with respect to the current $\hat{\boldsymbol{w}}^t$ pass the local samples in the kth AEA. In the local update in the kth AEA, for the selected sample $\hat{\boldsymbol{x}}_p^k$, guided by the new local objective, the local gradient is

$$\nabla \tilde{f}_k(\hat{\boldsymbol{w}}^*, \hat{\boldsymbol{x}}_p^k) = \hat{\boldsymbol{g}}_{k,p}^t - \hat{\boldsymbol{g}}_{k,p} + \bar{\boldsymbol{g}}^t + \hat{\mu}(\hat{\boldsymbol{w}}_k - \hat{\boldsymbol{w}}^t)$$

and the local model update is as shown in Line 6 in Procedure 2. In each updating, we only need to compute one gradient corresponding to the sample selected, that is, $\hat{\boldsymbol{g}}_{k,p}^t$, since $\hat{\boldsymbol{g}}_{k,p}^t$ and $\bar{\boldsymbol{g}}^t$ already exist. After the stochastic local updates in the participated AEAs, the difference vectors $\{\Delta\hat{\boldsymbol{w}}_k\}_{k \in \boldsymbol{C}_t}$ among the updated local model parameters and $\hat{\boldsymbol{w}}^t$ are transmitted to the central server to obtain the updated global model as shown in Line 12 in Algorithm 1.

Procedure 2 Local-Updating-HFSVRG ($\bar{\boldsymbol{g}}^t$, $\hat{\boldsymbol{w}}^t$, $\hat{\boldsymbol{S}}_k$, $\boldsymbol{G_k}$)

1: $\hat{\boldsymbol{w}}_k \leftarrow \hat{\boldsymbol{w}}^t$
2: **for** $\hat{\boldsymbol{x}}_p^k \in \hat{\boldsymbol{S}}_k$, $\hat{\boldsymbol{g}}_{k,p} \in \boldsymbol{G_k}$ **do**
3: $\quad$ Evaluate $\tilde{\boldsymbol{\theta}}_p$ by $\hat{\boldsymbol{w}}_k$ and $\hat{\boldsymbol{x}}_p^k$ in Eq. (7)
4: $\quad \boldsymbol{g}_{i,p}^k = 2(\tilde{\theta}_p^i - \theta_p^i)(\tilde{\theta}_p^i - (\tilde{\theta}_p^i)^2)\hat{\boldsymbol{x}}_p^k$ for $i = 1, \cdots, n$
5: $\quad \hat{\boldsymbol{g}}_{k,p}^t = [\boldsymbol{g}_{1,p}^{kT}, \cdots, \boldsymbol{g}_{n,p}^{kT}]^T$
6: $\quad \hat{\boldsymbol{w}}_k = \hat{\boldsymbol{w}}_k - \eta_{local}[\hat{\boldsymbol{g}}_{k,p}^t - \hat{\boldsymbol{g}}_{k,p} + \bar{\boldsymbol{g}}^t + \hat{\mu}(\hat{\boldsymbol{w}}_k - \hat{\boldsymbol{w}}^t)]$
7: **end for**
8: Return $\triangle\hat{\boldsymbol{w}}_k \leftarrow \hat{\boldsymbol{w}}^t - \hat{\boldsymbol{w}}_k$

B. Transform Domain and HoG-Wavelet-Enabled FSVRG

In the training procedure of the kth AEA, the rate of return data $\{\boldsymbol{X}_p^k\}_{p=1,\ldots,P_k}$ directly impact the gradient information. However, the rate of return data is highly dynamic with low SNR properties which leads to difficult training of the local models based on the gradient information. To tackle this challenge, we redesign the representation of each data sample by combining wavelet transform and HoG feature extraction as shown in Algorithm 2.

The wavelet transformation matrix is a combination of low-pass and high-pass filter matrices to achieve decomposition at different levels. The low-pass filter conducts averaging of local features while its mirror counterpart high-pass filter produces details. The coefficients of the low-pass filter are denoted by $\boldsymbol{\iota} = [\iota_1, \ldots, \iota_u]$ which is fully and uniquely specified by the choice of the wavelet basis. The coefficients of the quadrature mirror high-pass filter are denoted by $\boldsymbol{h} = [h_1, \ldots, h_u]$ where $h_i = (-1)^{\tau-i}\iota_{u-s-i}$ for arbitrary fixed integers τ and s.

We define $\boldsymbol{L}$ and $\boldsymbol{H}$ as filter operators that perform low-pass and high-pass filtering using quadrature mirror filters $\boldsymbol{\iota}$ and $\boldsymbol{h}$, respectively. This pair of filter operators is the basic block to construct the wavelet transformation matrix. For the wavelet transform with J decomposition levels, the transformation matrix $\boldsymbol{M^J}$ is generated by a set of filter operator pairs, that is, $\{(\boldsymbol{L}_j, \boldsymbol{H}_j)\}_{j=1,\ldots,J}$. The construction rule for the wavelet transformation matrix $\boldsymbol{M^J}$ with J decomposition levels is to cascade the inner product with the previous low-pass filter operator $\boldsymbol{L}_j$ by the current filter matrix $[\boldsymbol{L_{j+1}} \quad \boldsymbol{H_{j+1}}]$, that is, $[\boldsymbol{L_{j+1}} \quad \boldsymbol{H_{j+1}}]^T \boldsymbol{L}_j \in R^{2n\times n}$. To construct the submatrices $[\boldsymbol{L}_j, \boldsymbol{H}_j]^T$ for $j = 1, 2, \ldots, J$, we define the dilations of low-pass and high-pass filters by inserting $j-1$ zeros between every two neighboring filter coefficients and we denote the coefficients of the dilated low-pass and high-pass filters as $\boldsymbol{\iota}^{[j-1]}$ and $\boldsymbol{h}^{[j-1]}$, respectively. $\boldsymbol{L}_j$ and $\boldsymbol{H}_j$ are filter operators using filters $(\boldsymbol{\iota}^{[j-1]}/\sqrt{2})$ and $(\boldsymbol{h}^{[j-1]}/\sqrt{2})$, respectively.

Since sample $\boldsymbol{X}_p^k$ contains the rate of returns of n different data products over l time slots, the left and right wavelet transformation matrices should coincide with the number of

data products and the time slots, respectively. We define the size of the filter matrices $\boldsymbol{L}$ and $\boldsymbol{H}$ the same as $n \times n$ according to the number of data products to define the transformation matrix $\boldsymbol{M}_n^J \in R^{(J+1)n\times n}$ which contains $J+1$ stacked submatrices of size $n \times n$. Similarly, we define the size of the filter matrices $\boldsymbol{L}$ and $\boldsymbol{H}$ the same as $l \times l$ according to the number of time slots for each data product in $\boldsymbol{X}_p^k$ to define the transformation matrix $\boldsymbol{M}_l^J \in R^{(J+1)l\times l}$ which contains $J+1$ stacked submatrices of size $l \times l$. The wavelet transformation of $\boldsymbol{X}_p^k$ with J decomposition levels can be conducted as $\boldsymbol{D}_p^J = \boldsymbol{M}_n^J \boldsymbol{X}_p^k \boldsymbol{M}_l^{JT}$. Soft thresholding is applied to the components of the wavelet coefficients $\boldsymbol{D}_p^J$ which contain one approximation part and several detailed parts. It shrinks the wavelet coefficients according to a prescribed threshold that is determined by the noise variance of the features in sample $\boldsymbol{X}_p^k$. To choose an appropriate value for threshold δ_p for de-noising sample $\boldsymbol{X}_p^k$, we evaluate δ_p by $\delta_p = (2\log n \cdot l)^{1/2}\sigma$, where σ^2 is the estimate noise variance. We denote the denoised sample for $\boldsymbol{X}_p^k$ by $\tilde{\boldsymbol{X}}_p^k$, and the denoising problem with J wavelet decomposition levels can be formulated as the L_1–L_2 minimization problem

$$\min_{\tilde{\boldsymbol{D}}_p^J} \frac{1}{2}\left\|\tilde{\boldsymbol{D}}_p^J - \boldsymbol{D}_p^J\right\|_F^2 + \delta_p\left\|\tilde{\boldsymbol{D}}_p^J\right\|_1$$
$$\text{s.t. } \tilde{\boldsymbol{D}}_p^J = \boldsymbol{M}_n^J \tilde{\boldsymbol{X}}_p^k \boldsymbol{M}_l^{JT}. \tag{19}$$

The closed-form solution $\tilde{\boldsymbol{D}}_p^{J*}$ is the wavelet coefficients of the denoised sample $\tilde{\boldsymbol{X}}_p$ which is evaluated by

$$\tilde{\boldsymbol{D}}_p^{J*} = \text{sign}\left(\boldsymbol{D}_p^J\right) * \max\left\{\left\|\boldsymbol{D}_p^J\right\| - \delta_p, 0\right\}. \tag{20}$$

Then, we need to reconstruct the denoised sample $\tilde{\boldsymbol{X}}_p^k$ from its soft-shrinking wavelet coefficients $\tilde{\boldsymbol{D}}_p^{J*}$. To ensure the reconstruction can be done by wavelet transform matrices with J decomposition level, intermediate matrices need to be designed to be integrated with the wavelet transformation matrices $\boldsymbol{M}_n^J$ and $\boldsymbol{M}_l^J$, respectively. We design a diagonal weight matrix $\tilde{\boldsymbol{T}}_n^J \in R^{(J+1)n\times(J+1)n}$ that rescales the square submatrices comprising the wavelet transformation matrix $\boldsymbol{M}_n^J$. We separate the diagonal entries of $\tilde{\boldsymbol{T}}_n^J$ into J groups, the first group contains the first $2n$ diagonal entries, and each of them is defined as $1/2^J$. The rest groups contain n entries in each of them, and each entry in the jth group is defined as $1/2^{J-j+1}$. Similarly, for wavelet transformation matrix $\boldsymbol{M}_l^J$, we design a diagonal weight matrix $\tilde{\boldsymbol{T}}_l^J \in R^{(J+1)l\times(J+1)l}$ that rescales the square submatrices comprising the wavelet transformation matrix $\boldsymbol{M}_l^J$. The diagonal entries of $\tilde{\boldsymbol{T}}_l^J$ are separated into J groups, the first group contains the first $2l$ diagonal entries, and each of them is defined as $1/2^J$. The rest groups contain l entries in each of them, and each entry in the jth group is defined as $1/2^{J-j+1}$. The denoising sample features can be obtained by $\tilde{\boldsymbol{X}}_p^k = \boldsymbol{M}_n^{JT}\tilde{\boldsymbol{T}}_n^J\tilde{\boldsymbol{D}}_p^{J*}\tilde{\boldsymbol{T}}_l^J\boldsymbol{M}_l^J$. Then, we continue to extract HoG features from $\tilde{\boldsymbol{X}}_p^k$ to construct the new representations of the market data as $\tilde{\boldsymbol{x}}_p^k$.

The number of parameters in each local model $\{\hat{\boldsymbol{w}}_k\}_{k=1,\ldots,K}$ is denoted by N. We define a frequency-related coordinate system and transform the local model drifts from the current global model into a frequency space in each federated training round where the resolution of the spectrum is indicated by N.

Algorithm 2 TDHW-FSVRG

Initialize global model parameters $\hat{\boldsymbol{w}}^0$
for $t \leftarrow 0$ to T **do**
  Randomly sample a set of AEAs C_t
  for $k \in C_t$ **in parallel do**
    $\{\bar{\boldsymbol{g}}_k, \hat{\boldsymbol{S}}_k, \boldsymbol{G}_k\} \leftarrow$ **GC-TDHW-FSVRG** (k, $\hat{\boldsymbol{w}}^t$)
  end for
  $|\boldsymbol{S}| = \sum_{k\in C_t} |\boldsymbol{S}_k|$
  $\bar{\boldsymbol{g}}^t = \frac{1}{|S|}\sum_{k\in C_t} \bar{\boldsymbol{g}}_k$
  for $k \in C_t$ **in parallel do**
    $\tilde{\boldsymbol{z}}_k \leftarrow$ **LU-TDHW-FSVRG** ($\bar{\boldsymbol{g}}^t$, $\hat{\boldsymbol{w}}^t$, $\hat{\boldsymbol{S}}_k$, $\boldsymbol{G}_k$)
    for $i \leftarrow \lceil N*(1-\gamma)\rceil$ to N **do**
      $\tilde{\boldsymbol{z}}_k \leftarrow \tilde{\boldsymbol{z}}_k \cup 0$
    end for
  end for
  $\hat{\boldsymbol{w}}^{t+1} \leftarrow \hat{\boldsymbol{w}}^t - \eta_{\text{global}}\boldsymbol{U}(\sum_{k\in C_t}\tilde{\boldsymbol{z}}_k)$
end for

Procedure 3 Gradient-Collection-TDHW-FSVRG (k, $\hat{\boldsymbol{w}}^t$)

Randomly select sample set $\boldsymbol{S}_k$
Initialize $\bar{\boldsymbol{g}}_k = 0$, $\hat{\boldsymbol{S}}_k \leftarrow \varnothing$ and $\boldsymbol{G}_k \leftarrow \varnothing$
for $\boldsymbol{X}_p^k \in \boldsymbol{S}_k$ **do**
  $\boldsymbol{D}_p^J = \boldsymbol{M}_n^J \boldsymbol{X}_p^k \boldsymbol{M}_l^{JT}$
  $\tilde{\boldsymbol{D}}_p^{J*} = \text{sign}(\boldsymbol{D}_p^J).*\max\{||\boldsymbol{D}_p^J|| - \delta_p, 0\}$
  $\tilde{\boldsymbol{X}}_p^k = \boldsymbol{M}_n^{JT}\tilde{\boldsymbol{T}}_n^J\tilde{\boldsymbol{D}}_p^{J*}\tilde{\boldsymbol{T}}_l^J\boldsymbol{M}_l^J$
  $\tilde{\boldsymbol{x}}_p^k \leftarrow \text{HoG}(\tilde{\boldsymbol{X}}_p^k)$
  Evaluate $\tilde{\boldsymbol{\theta}}_p$ by $\hat{\boldsymbol{w}}^t$ and $\tilde{\boldsymbol{x}}_p^k$ in Eq. (7)
  $\boldsymbol{g}_{i,p}^k = 2(\tilde{\theta}_p^i - \theta_p^i)(\tilde{\theta}_p^i - (\tilde{\theta}_p^i)^2)\tilde{\boldsymbol{x}}_p^k$ for $i = 1, \cdots, n$
  $\hat{\boldsymbol{g}}_{k,p} = [\boldsymbol{g}_{1,p}^{kT}, \cdots, \boldsymbol{g}_{n,p}^{kT}]^T$
  $\bar{\boldsymbol{g}}_k \leftarrow \bar{\boldsymbol{g}}_k + \hat{\boldsymbol{g}}_{k,p}$, $\hat{\boldsymbol{S}}_k \leftarrow \hat{\boldsymbol{S}}_k \cup \tilde{\boldsymbol{x}}_p^k$, $\boldsymbol{G}_k \leftarrow \boldsymbol{G}_k \cup \hat{\boldsymbol{g}}_{k,p}$
end for
$\bar{\boldsymbol{g}}_k \leftarrow \frac{\bar{\boldsymbol{g}}_k}{|\boldsymbol{S}_k|}$
Return $\{\bar{\boldsymbol{g}}_k, \hat{\boldsymbol{S}}_k, \boldsymbol{G}_k\}$

Procedure 4 Local-Updating-TDHW-FSVRG ($\bar{\boldsymbol{g}}^t$, $\hat{\boldsymbol{w}}^t$, $\hat{\boldsymbol{S}}_k$, $\boldsymbol{G}_k$)

$\hat{\boldsymbol{w}}_k \leftarrow \hat{\boldsymbol{w}}^t$ and $z_\gamma^k \leftarrow \varnothing$
for $\tilde{\boldsymbol{x}}_p^k \in \hat{\boldsymbol{S}}_k$, $\hat{\boldsymbol{g}}_{k,p} \in \boldsymbol{G}_k$ **do**
  Evaluate $\tilde{\boldsymbol{\theta}}_p$ by $\hat{\boldsymbol{w}}_k$ and $\tilde{\boldsymbol{x}}_p^k$ in Eq. (7)
  $\boldsymbol{g}_{i,p}^k = 2(\tilde{\theta}_p^i - \theta_p^i)(\tilde{\theta}_p^i - (\tilde{\theta}_p^i)^2)\tilde{\boldsymbol{x}}_p^k$ for $i = 1, \cdots, n$
  $\hat{\boldsymbol{g}}_{k,p}^t = [\boldsymbol{g}_{1,p}^{kT}, \cdots, \boldsymbol{g}_{n,p}^{kT}]^T$
  $\hat{\boldsymbol{w}}_k = \hat{\boldsymbol{w}}_k - \eta_{local}[\hat{\boldsymbol{g}}_{k,p}^t - \hat{\boldsymbol{g}}_{k,p} + \bar{\boldsymbol{g}}^t + \hat{\mu}(\hat{\boldsymbol{w}}_k - \hat{\boldsymbol{w}}^t)]$
end for
$\triangle\hat{\boldsymbol{w}}_k \leftarrow \hat{\boldsymbol{w}}^t - \hat{\boldsymbol{w}}_k$
for $i \leftarrow 1$ to $\lceil N*(1-\gamma)\rceil$ **do**
  Evaluate z_i^k by Eq. (22)
  $z_\gamma^k \leftarrow z_\gamma^k \cup z_i^k$
end for
Return z_γ^k

To ensure that the frequency-related basis vectors are orthogonal, we set the number of the basis vectors as N. By designing the frequency-related basis vectors to be orthonormal vectors, the transformation of local model parameters from the time

domain into the frequency domain can be regarded as a projection of the model parameters on the frequency-related basis vectors. The frequency-related coordinate system is denoted by $\boldsymbol{U} = [\boldsymbol{u_1}, \ldots, \boldsymbol{u}_N] \in R^{N\times N}$, where

$$\boldsymbol{u_i} = \alpha_i \sqrt{\frac{2}{N}} \left[\cos\left(\frac{1 \cdot i\pi}{2N}\right) \quad \cdots \quad \cos\left(\frac{(2N-1)\cdot i\pi}{2N}\right) \right]^T \tag{21}$$

and

$$\alpha_i = \begin{cases} \dfrac{1}{\sqrt{2}}, & i = 0 \\ 1, & i = 1, 2, \ldots, N-1. \end{cases}$$

The orthonormality of frequency basis vectors is guaranteed. Basis vector $\boldsymbol{u_i}$ can combine the model parameter drifts $\triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}}$ and obtain the one corresponding spectral feature which can be regarded as the projection of $\triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}}$ on the frequency domain as shown in

$$z_i^k = \alpha_i \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} \triangle \hat{w}_k{}^n \cos^2\left(\frac{(2n+1)\cdot i\pi}{2N}\right) \tag{22}$$

and $\boldsymbol{z^k} = \{z_i^k\}_{i=0,\ldots,N-1}$ represents the projection of $\triangle\hat{\boldsymbol{w}}_{\boldsymbol{k}}$ in the frequency domain. Thanks to that after the transformation, most of the local model drift information compacts into a small portion of $\boldsymbol{z^k}$. We define reduction rate γ for federated training where the kth AEA only needs to deliver $\boldsymbol{z}_{\gamma}^{\boldsymbol{k}} = \{z_i^k\}_{i=0,\ldots,\lceil N*(1-\gamma)\rceil}$ which will greatly reduce the communication burden among AEAs. Then, the central server zero-padding the received compacted local drifts in the frequency domain denoted by $\{\tilde{\boldsymbol{z}}_{\boldsymbol{k}}\}_{k\in C_t}$. By transforming the aggregated frequency-domain drifts back into the time domain, the global model is updated as shown in Line 15 in Algorithm 2.

C. Algorithm Convergence Analysis

To the convergence of the algorithm, the stochastic selection of $\boldsymbol{S_k}$ introduces variance in the local update of the kth AEA. Instead of the traditional stochastic gradient update which tries to fast estimate the entire gradient of the local objective function $f_k(\hat{\boldsymbol{w}}_{\boldsymbol{k}})$ by randomly choosing $|\boldsymbol{S_k}|$ samples from the local dataset, we also measure the variance from the local updates. At the tth round, $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$ is the best estimation of the global model, and the local updates of the kth AEA make the local model drift away from $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$ which introduces the variance between the models $\hat{\boldsymbol{w}}_{\boldsymbol{k}} - \hat{\boldsymbol{w}}^{\boldsymbol{t}}$. When we apply the randomly selected sample to evaluate the local gradient with $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$, it introduces another variance from $\bar{\boldsymbol{g}}^{\boldsymbol{t}}$, that is, $\bar{\boldsymbol{g}}^{\boldsymbol{t}} - \hat{\boldsymbol{g}}_{\boldsymbol{k},\boldsymbol{p}}$. When combining these two variances with the local gradient $\hat{\boldsymbol{g}}_{\boldsymbol{k},\boldsymbol{p}}^{\boldsymbol{t}}$, it yields an unbiased estimate of the gradient of the global objective function. In the redesigned local objective $\tilde{f}_k(\hat{\boldsymbol{w}}_{\boldsymbol{k}})$, the quadratic term has very nice properties that the curvature information in $f_k(\hat{\boldsymbol{w}}_{\boldsymbol{k}})$ is nearly intact since the added quadratic term only makes perturbation on the diagonal of the original Hessian with a constant value defined by $\hat{\mu}$. Furthermore, it is also guaranteed that after sufficient rounds, the minimum of each local objective with this perturbation will converge to the optimal solution $\hat{\boldsymbol{w}}^*$ for the federated optimization. When the global model $\hat{\boldsymbol{w}}^{\boldsymbol{t}}$ approaches the optimal design $\hat{\boldsymbol{w}}^*$, the variance of both $\bar{\boldsymbol{g}}^{\boldsymbol{t}} - \hat{\boldsymbol{g}}_{\boldsymbol{k},\boldsymbol{p}}$ and $\hat{\boldsymbol{w}}_{\boldsymbol{k}} - \hat{\boldsymbol{w}}^{\boldsymbol{t}}$ will approach $\boldsymbol{0}$. Then, there will be no change in the local updating step, which means the local training procedure converges to the global optimal design $\hat{\boldsymbol{w}}^*$.

The careful data representation design in HFSVRG and TDHW-FSVRG plays a crucial role in ensuring data confidentiality by minimizing the risk of reverse engineering. HFSVRG specifically focuses on capturing the relative changes in the rate of return within local data blocks. By encoding the local data into statistical features before the training process, the reconstruction attack becomes infeasible since the operation is non-invertible.

In addition, TDHW-FSVRG leverages transform-domain model aggregation, offering benefits beyond improved communication efficiency, including increased confidentiality. Unlike the original local model aggregation, TDHW-FSVRG reduces the dimensionality of local model parameters by projecting the original model into the frequency domain and eliminating high-frequency features with small magnitudes. This approach not only improves efficiency, but also defends against local model reconstruction attacks, as the model aggregation now occurs in the transform-domain model parameters.

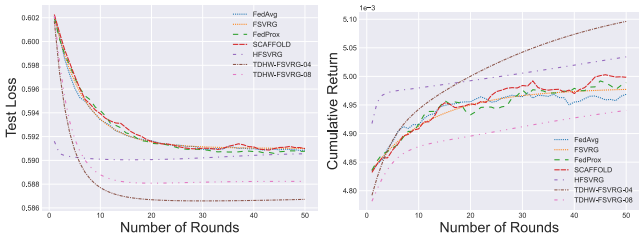
VI. Simulations

A. Dataset Design

As we aim to demonstrate the benefits of the data trading market for both resource-demanding companies and individual data owners, we do not have specific real datasets for this problem. However, we can leverage real stock pricing data as a suitable substitute since it shares similarities with the revenue return concept in our work, displaying similar fluctuations and randomness. While the data trading market differs from the traditional stock market, we utilize historic price data from stocks listed in the S&P 500 index to create multiple local datasets that reflect market information for data products. Specifically, we select a subset of stocks that have been part of the index from January 4, 2007, until June 25, 2021. The adjusted close prices of these stocks emulate the prices of various data products. Next, we divide the available data into two parts: one for constructing local training datasets and the other for the test dataset. The division follows a chronological order, ensuring the sequence of rate of return data. Around 80% of the data is allocated to creating 20 local training datasets, while the remaining 20% is reserved for the test dataset.

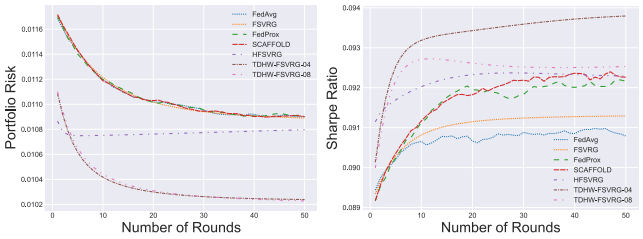
B. Parameter Settings and Performance Metrics

To generate $\{\boldsymbol{X}_{\boldsymbol{p}}^{\boldsymbol{k}}\}_{p=1,\ldots,P_k}$ and $\{\boldsymbol{B}_{\boldsymbol{p}}^{\boldsymbol{k}}\}_{p=1,\ldots,P_k}$, we set the window sizes as $l = 10$ and $m = 10$. Each AEA covers five different types of data products ($n = 5$) in the data trading market. The learning rate in the local training procedure is set to 0.1 by default. When designing the labels, we set the tradeoff between risk and revenue returns as $\lambda = 20$ [34]. For wavelet denoising, we estimate the noise standard deviation as $\sigma = 0.01$ to compute the threshold δ. In HoG feature extraction, the block size is set as 3×3, and the number of bins to record information in each block is $\Omega = 5$. The stride



(a) (b)

Fig. 2. Convergence of different federated schemes with (a) test loss and (b) cumulative return.



(a) (b)

Fig. 3. Convergence of different federated schemes with (a) portfolio risk and (b) Sharpe ratio.

size for moving the local block is set to half of the block size, which is $\lceil 3/2 \rceil = 2$.

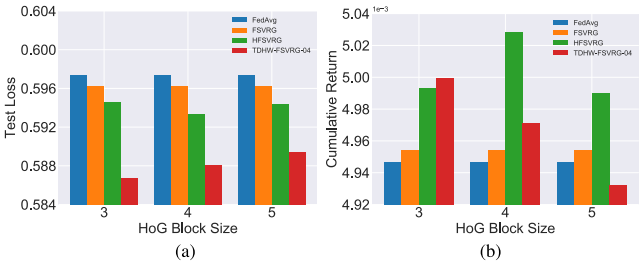
We utilize root mean square error (RMSE), cumulative return, portfolio risk, and Sharpe ratio as performance metrics for evaluating the AEAs' portfolio allocation decisions. The revenue period is defined by the validity period of the portfolio allocation decision, and the expected revenue and volatility are measured accordingly. To compare the performance, we evaluate our proposed HFSVRG and TDHW-FSVRG algorithms against existing FL algorithms, including FedAvg [25], FSVRG [30], FedProx [27], and SCAFFOLD [29]. In the evaluation of TDHW-FSVRG, we consider different settings with $\gamma = 0.4$ and $\gamma = 0.8$ for local model drift compression, denoted as TDHW-FSVRG-04 and TDHW-FSVRG-08, respectively. These variations achieve a reduction of transmitted data by 40% and 80% among the AEAs and the central server.

C. Simulation Results and Analysis

We begin our analysis by examining the convergence behavior of our proposed methods in comparison to various baseline approaches. As illustrated in Fig. 2(a), the convergence of HFSVRG, TDHW-FSVRG-04, and TDHW-FSVRG-08 outperforms that of the existing methods, including FedAvg, FSVRG, FedProx, and SCAFFOLD. This result clearly demonstrates the superior convergence speed achieved by our proposed approaches. In addition, it is worth highlighting that TDHW-FSVRG exhibits remarkable performance, converging to significantly lower test loss values compared to the other schemes, even when employing an 80% reduction in transmitted data. This remarkable outcome is attributed to the denoising capabilities of TDHW-FSVRG, particularly when dealing with nonstationary features.

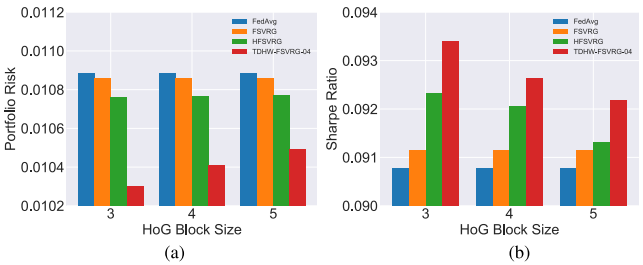
The significantly faster convergence rates observed in HFSVRG, TDHW-FSVRG-04, and TDHW-FSVRG-08, coupled with TDHW-FSVRG's ability to achieve lower test loss through efficient denoising, unequivocally establish the effectiveness and superiority of our approach in addressing the challenges of significant delays and a substantial communication burden on the network infrastructure. This is particularly important when it is difficult to ensure AEA participation during overextended periods.

Moving on to the convergence performance, as demonstrated by the averaged cumulative return in Fig. 2(b), HFSVRG achieves rapid convergence to a relatively high cumulative return compared to the benchmark methods. TDHW-FSVRG outperforms HFSVRG, converging to an even better cumulative return, even with a conservative reduction



(a) (b)

Fig. 4. (a) Test loss and (b) cumulative return for different HoG block sizes.



(a) (b)

Fig. 5. (a) Portfolio risk and (b) Sharpe ratio for different HoG block sizes.

of no more than 40% in transmitted data. This superior convergence performance demonstrates the significant benefits of TDHW-FSVRG in achieving favorable cumulative returns. The portfolio risk analysis, as depicted in Fig. 3(a), exhibits a similar trend, wherein the predicted portfolio allocations from our proposed schemes yield less risky investment strategies, requiring significantly fewer federated rounds. This outcome highlights the effectiveness of our approaches in generating portfolio allocations with reduced risk. Furthermore, the evaluation of the Sharpe ratio, as depicted in Fig. 3(b), highlights the advantages of employing feature extraction-enabled FL for generating promising portfolio allocations that consider both investment risk and returns. Both HFSVRG and TDHW-FSVRG achieve substantially higher Sharpe ratios compared to the benchmark methods. Notably, TDHW-FSVRG surpasses HFSVRG in both convergence rate and converged performance on the Sharpe ratio, even with an 80% reduction in transmitted data. TDHW-FSVRG, particularly in the context of investment risk and return, demonstrates superior performance in generating robust and promising portfolio allocations.

We conducted an investigation into the impact of different settings for the local block size when constructing HoG features while maintaining a transmission data reduction rate of 40% for TDHW-FSVRG. The results, presented in Figs. 4 and 5, demonstrate that both HFSVRG and TDHW-FSVRG-04 consistently outperform the benchmark methods

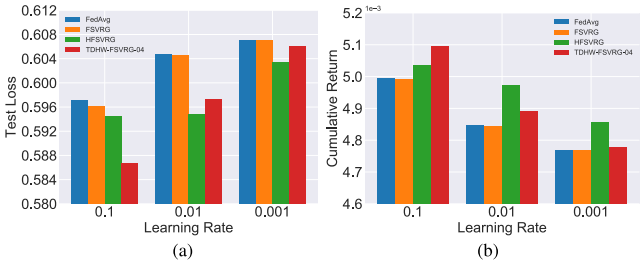


Fig. 6. (a) Test loss and (b) cumulative return for different learning rates.

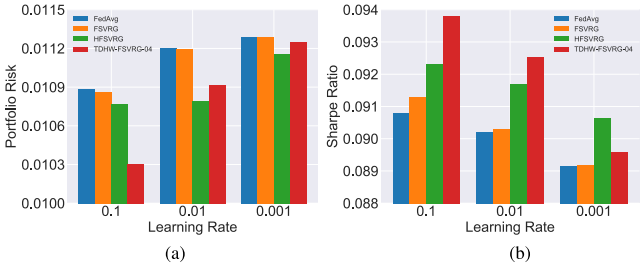


Fig. 7. (a) Portfolio risk and (b) Sharpe ratio for different learning rates.

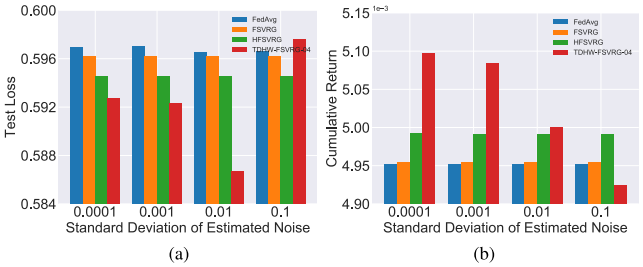


Fig. 8. (a) Test loss and (b) cumulative return for different standard deviations of estimated noise.

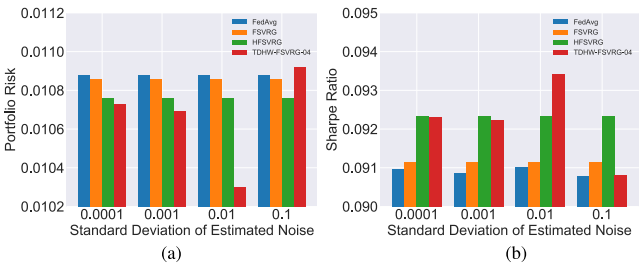


Fig. 9. (a) Portfolio risk and (b) Sharpe ratio for different standard deviation of estimated noise.

across all four evaluation metrics. TDHW-FSVRG-04 exhibits substantial improvements over the benchmarks in terms of test loss and investment metrics. Among the different block sizes tested, TDHW-FSVRG-04 achieved the best performance with the smallest block size of 3 in all four metrics. A smaller block size allows for a more focused analysis of detailed information contained in the HoG features. In addition, a smaller block size results in increased overlaps between adjacent blocks, as we set the stride size to move the block by half of the block size. This overlap enhances the availability of local gradient information. The results show that the performance of TDHW-FSVRG-04 deteriorates with larger block sizes.

On the other hand, the situation is different for HFSVRG, as it does not apply denoising to the nonstationary raw features. In Fig. 4, it is evident that the performance of HFSVRG in both test loss and cumulative return is worse when using a block size of 3 compared to a block size of 4, due to the presence of noisy local detailed features. However, this does not imply that enhancing the local feature information is inapplicable to noisy features. The result also reveals that the performance of HFSVRG deteriorates when increasing the block size from 4 to 5, indicating the importance of a suitable enhancement for local feature information even in the presence of noise. Overall, our analysis highlights the significance of selecting an appropriate local block size when constructing HoG features. While TDHW-FSVRG-04 benefits from a smaller block size, which focuses on detailed information, HFSVRG may face challenges when handling noisy features. These findings emphasize the importance of carefully considering the characteristics of the features and the denoising capabilities of the algorithms to achieve promising performance in AEA trading scenarios.

We conduct an investigation in different learning rate settings while maintaining a transmission data reduction rate of 40% for TDHW-FSVRG. As depicted in Figs. 6 and 7, decreasing the learning rate leads to a decline in performance for all FL schemes. This can be attributed to the fact that a smaller learning rate requires more federated training rounds. However, it is important to note that the proposed HFSVRG and TDHW-FSVRG-04 consistently outperform the benchmark methods across all learning rate settings. In practice, it is often difficult to guarantee a sufficient number of training rounds due to the dynamic nature of the environment. In Figs. 6 and 7, it is evident that when we set the learning rate to 0.1 with 50 federated training rounds, TDHW-FSVRG-04 achieves significantly better performance compared to the other methods. However, TDHW-FSVRG-04 exhibits higher sensitivity to changes in the learning rate compared to HFSVRG. Further analysis in Fig. 6(b) reveals that HFSVRG achieves higher cumulative returns compared to the other methods when the learning rate is no larger than 0.01. However, it is worth noting that the Sharpe ratio of HFSVRG does not surpass that of TDHW-FSVRG-04 until the learning rate is reduced to 0.001, as shown in Fig. 7(b). This suggests that the portfolio predictions generated by TDHW-FSVRG-04 exhibit greater robustness against investment risk, even in the face of significant changes in the learning rate.

We conducted a detailed investigation into different settings of soft thresholding while maintaining a transmission data reduction rate of 40% for TDHW-FSVRG. The threshold in the feature denoising procedure is directly influenced by the standard deviation of the estimated noise. The results in Fig. 8(b) highlight that the cumulative return performance of TDHW-FSVRG-04 progressively deteriorates with increasing standard deviation of the estimated noise, ranging from 0.0001 to 0.1. Notably, when the standard deviation of the estimated noise does not exceed 0.01, TDHW-FSVRG-04 achieves the highest cumulative return among all methods. Additionally, the risk associated with TDHW-FSVRG-04 is also the lowest, as demonstrated in Fig. 9(a).

However, when the standard deviation of the estimated noise reaches 0.1, the performance of TDHW-FSVRG-04 experiences a significant deterioration in terms of both cumulative return and risk. This is because larger standard deviations

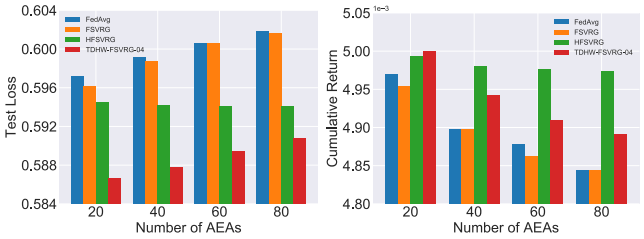


Fig. 10. (a) Test loss and (b) cumulative return for different scales of the number of AEAs.

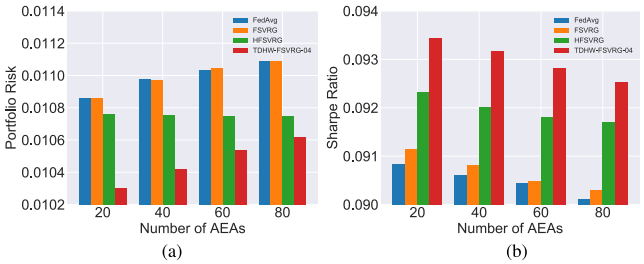


Fig. 11. (a) Portfolio risk and (b) Sharpe ratio for different scales of the number of AEAs.

result in the elimination of more information in the features during the thresholding process. When features fall below the threshold, they are shrunk to zero, leading to a reduction in cumulative return as shown in Fig. 8(b). Moreover, the thresholding operation smooths all feature values that exceed the threshold, and when the estimated noise closely approximates the true noise, better performance is achieved. This explains why TDHW-FSVRG-04 attains the smallest risk when the standard deviation of the estimated noise is set to 0.01 as shown in Fig. 9(a).

The results demonstrate that the standard deviation of the estimated noise plays a crucial role in the performance of TDHW-FSVRG-04. Setting a lower standard deviation allows for better preservation of relevant information, resulting in higher cumulative return and lower risk. However, excessively high standard deviations lead to information loss and degradation in performance. Careful selection of the thresholding parameters is vital to strike a balance between noise reduction and feature preservation, ultimately enhancing the performance of TDHW-FSVRG-04 in portfolio allocation.

We conducted a scalability analysis by investigating different scales of participated AEAs while maintaining a 40% transmission data reduction rate for TDHW-FSVRG. Figs. 10 and 11 present the results, showing the performance across various metrics. Both HFSVRG and TDHW-FSVRG-04 consistently outperformed the benchmarks, demonstrating their scalability in handling larger-scale AEAs. HFSVRG showcased its advantage of fast convergence as the number of participated AEAs increased. The test loss decreased with more AEAs, indicating improved accuracy in portfolio allocation as shown in Fig. 10(a). Conversely, the other schemes experienced an increase in test loss, as they required more federated rounds to converge.

However, as the number of participated AEAs grew, the performance of TDHW-FSVRG-04 on cumulative return and risk deteriorated as shown in Figs. 10(b) and 11(a). This

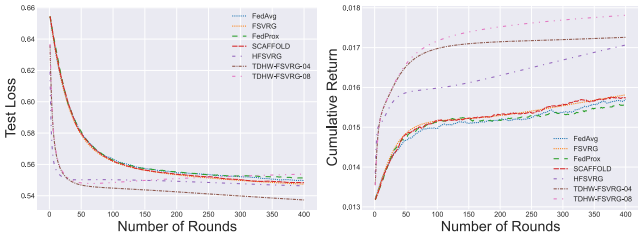


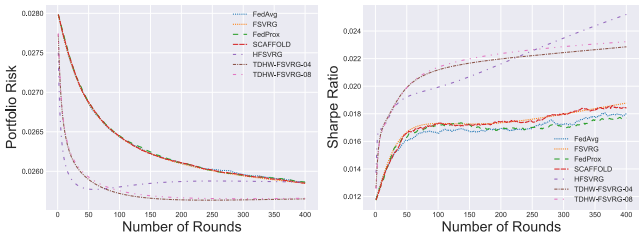
Fig. 12. Convergence of different federated schemes with (a) test loss and (b) cumulative return.

can be attributed to the increased demand for federated training rounds, which posed challenges for maintaining performance due to the dynamic environments. On the other hand, HFSVRG exhibited fast convergence and maintained favorable cumulative returns. Despite HFSVRG's superior cumulative return performance for higher numbers of AEAs, TDHW-FSVRG-04 consistently achieved higher Sharpe ratios as shown in Fig. 11(b). This indicates its robust risk control capabilities even in the face of scalability challenges. The scalability analysis demonstrates that HFSVRG excels in fast convergence and improved accuracy with an increasing number of participated AEAs, while TDHW-FSVRG-04 showcases robust risk control. These findings provide valuable insights into the tradeoffs between convergence speed, cumulative return, and risk management when scaling the participated AEAs.

To evaluate the effectiveness and generalizability of our proposed approaches, we conducted a comparison using a new dataset consisting of five popular cryptocurrencies: Bitcoin, Ethereum, Dogecoin, Cardano, and Ripple. The results, as depicted in Figs. 12 and 13, demonstrate that our proposed methods consistently outperform the baselines across multiple metrics. Both HFSVRG and TDHW-FSVRG exhibit faster convergence compared to the other baselines, yielding superior results. While HFSVRG excels in convergence speed, it falls short of TDHW-FSVRG in terms of the final performance metrics, including test loss, cumulative return, and risk. Additionally, TDHW-FSVRG demonstrates greater stability in its performance compared to HFSVRG. It is worth noting that the advantage of HFSVRG's fast convergence diminishes when compared to TDHW-FSVRG, particularly in terms of cumulative returns. Overall, our proposed methods surpass the baselines in both convergence speed and final performance using this new dataset. These findings validate the effectiveness and generalizability of our proposed approaches, highlighting their ability to outperform the baselines across multiple metrics when applied to a new dataset of popular cryptocurrencies. The results obtained from the new dataset reinforce the superiority of our methods in terms of convergence speed and converged performance.

D. Recommendations for System Settings

Based on our analysis, we have the following recommendations for optimizing system settings in FL for portfolio allocations. When dealing with highly dynamic local data, it is advisable to use larger block sizes for HFSVRG when



(a) (b)

Fig. 13. Convergence of different federated schemes with (a) portfolio risk and (b) Sharpe ratio.

generating HoG features. This prevents performance degradation caused by excessive focus on details. On the other hand, TDHW-FSVRG benefits from smaller block sizes due to its denoising capability, which enhances local gradient information. The choice of block size should be based on the specific characteristics of the data and noise levels. The learning rate plays a critical role in achieving fast convergence and maintaining promising performance. While larger learning rates can expedite convergence, they can also introduce oscillations near the solution, leading to poorer performance, especially in the dynamic data trading market. Finding the right balance between convergence speed and stability is crucial. Careful tuning of the learning rate based on convergence and accuracy requirements is recommended. To optimize the performance of TDHW-FSVRG, it is important to accurately estimate the standard deviation of the noise among the local return data. TDHW-FSVRG exhibits superior performance when the estimated noise closely matches the true noise since the precise estimation of the noise level enables effective denoising and preservation of valuable information in the features. HFSVRG demonstrates its strength when operating with large-scale participated AEAs, showcasing the robust performance of the cumulative return with increasing participated AEAs. On the other hand, TDHW-FSVRG excels in providing more promising automotive portfolio allocations, particularly with relatively smaller-scale participated AEAs. By following these recommendations and appropriately adjusting the system settings, practitioners can optimize the performance of the proposed methods according to their current data training scenario, which will result in improved convergence, accuracy, and scalability of the FL process.

E. Wilcoxon Test

We conducted the Wilcoxon signed-rank test [35] to analyze the experimental results of the test loss, cumulative return, portfolio risk, and Sharpe ratio between our proposed methods and the baselines. The purpose of the Wilcoxon test is to determine whether the results obtained from our proposed algorithms are significantly better than those of the baselines. To perform this analysis, we paired the results from our proposed algorithms with those from the baselines. This paired sample test was then transformed into a one-sample test by replacing each pair with their difference. The differences between the pairs were ranked accordingly.

The p-value serves as a measure of the smallest level of significance at which a significant difference can be observed between the performance of our proposed methods and the baselines. We obtained p-values for the test loss and cumulative return metrics, which are presented in Table IV. Similarly, for the portfolio risk and Sharpe ratio metrics, we obtained p-values, which are shown in Table V. A smaller p-value indicates stronger evidence in favor of our proposed methods outperforming the baselines. The Wilcoxon signed-rank test provides statistical evidence to support the superiority of our proposed methods over the baselines in terms of test loss, cumulative return, portfolio risk, and Sharpe ratio metrics.

TABLE IV

p-VALUES FOR TEST LOSS AND CUMULATIVE RETURN

Wilcoxon Test Pairs	Test Loss	Cumulative Return
TDHW-FSVRG v.s. FedAvg	3.553×10^{-15}	1.019×10^{-11}
TDHW-FSVRG v.s. FSVRG	4.547×10^{-13}	2.569×10^{-11}
HW-FSVRG v.s. FedAvg	1.421×10^{-14}	9.313×10^{-10}
HW-FSVRG v.s. FSVRG	1.819×10^{-12}	4.657×10^{-10}

TABLE V

p-VALUES FOR PORTFOLIO RISK AND SHARPE RATIO

Wilcoxon Test Pairs	Portfolio Risk	Sharpe Ratio
TDHW-FSVRG v.s. FedAvg	3.553×10^{-15}	1.776×10^{-15}
TDHW-FSVRG v.s. FSVRG	9.095×10^{-13}	4.547×10^{-13}
HW-FSVRG v.s. FedAvg	1.137×10^{-13}	2.842×10^{-14}
HW-FSVRG v.s. FSVRG	1.819×10^{-12}	1.819×10^{-12}

VII. CONCLUSION

Our work has effectively addressed the obstacles associated with smooth data access by introducing a data trading market with AEAs. Through the integration of federated training, feature extraction, and transform-domain techniques, we have provided a robust and efficient solution for autonomous trading among AEAs. The demonstrated improvements in terms of convergence speed, cumulative return, and risk reduction validate the effectiveness of our approach. This research opens up new opportunities for innovation and progress in various industries that heavily rely on data-driven intelligence and sets the stage for further developments in this exciting field. In future research, we will concentrate on designing more versatile FL algorithms to cater to the diverse requirements of individual AEAs. Additionally, we aim to address the challenge of accelerating the FL training procedure, considering the highly dynamic participation environment of AEAs. These areas of focus will contribute to further advancements in data trading and enable more efficient and effective collaboration among AEAs.

REFERENCES

[1] W. Wang, Y. Zhang, J. Gu, and J. Wang, "A proactive manufacturing resources assignment method based on production performance prediction for the smart factory," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 46–55, Jan. 2022.

[2] A. Belhadi, Y. Djenouri, G. Srivastava, and J. C.-W. Lin, "Reinforcement learning multi-agent system for faults diagnosis of mircoservices in industrial settings," *Comput. Commun.*, vol. 177, pp. 213–219, Sep. 2021.

[3] S. Lee and D.-H. Choi, "Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 488–497, Jan. 2022.

[4] N. Rieke et al., "The future of digital health with federated learning," *npj Digit. Med.*, vol. 3, no. 1, p. 119, Sep. 2020.

[5] T. AlSkaif, J. L. Crespo-Vazquez, M. Sekuloski, G. van Leeuwen, and J. P. S. Catalão, "Blockchain-based fully peer-to-peer energy trading strategies for residential energy systems," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 231–241, Jan. 2022.

[6] D. Minarsch, S. A. Hosseini, M. Favorito, and J. Ward, "Autonomous economic agents as a second layer technology for blockchains: Framework introduction and use-case demonstration," in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2020, pp. 27–35.

[7] W. Y. B. Lim et al., "When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 457–466, Jan. 2022.

[8] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 923–937, Jul. 2021.

[9] K. Atkinson and T. Bench-Capon, "States, goals and values: Revisiting practical reasoning," *Argument Comput.*, vol. 7, nos. 2–3, pp. 135–154, Nov. 2016.

[10] D. Minarsch, M. Favorito, S. A. Hosseini, Y. Turchenkov, and J. Ward, "Autonomous economic agent framework," in *Proc. Int. Workshop Eng. Multi-Agent Syst.* Springer, 2021, pp. 237–253.

[11] M. P. Clements, P. H. Franses, and N. R. Swanson, "Forecasting economic and financial time-series with non-linear models," *Int. J. Forecasting*, vol. 20, no. 2, pp. 169–183, Apr. 2004.

[12] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 28, 2022, doi: 10.1109/TNNLS.2022.3160699.

[13] X. Lin, J. Wu, A. K. Bashir, J. Li, W. Yang, and M. J. Piran, "Blockchain-based incentive energy-knowledge trading in IoT: Joint power transfer and AI design," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14685–14698, Aug. 2022.

[14] G. Gao, Y. Wen, and D. Tao, "Distributed energy trading and scheduling among microgrids via multiagent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 12, 2022, doi: 10.1109/TNNLS.2022.3170070.

[15] I. Lederer, R. Mayer, and A. Rauber, "Identifying appropriate intellectual property protection mechanisms for machine learning models: A systematization of watermarking, fingerprinting, model access, and attacks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 2, 2023, doi: 10.1109/TNNLS.2023.3270135.

[16] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[17] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1407–1418, May 2019.

[18] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017.

[19] X.-Y. Liu et al., "FinRL-meta: Market environments and benchmarks for data-driven financial reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 1835–1849.

[20] K. Park, H.-G. Jung, T.-S. Eom, and S.-W. Lee, "Uncertainty-aware portfolio management with risk-sensitive multiagent network," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 23, 2022, doi: 10.1109/TNNLS.2022.3174642.

[21] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Deep adaptive input normalization for time series forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3760–3765, Sep. 2020.

[22] A. Tsantekidis, N. Passalis, A.-S. Toufa, K. Saitas-Zarkias, S. Chairistanidis, and A. Tefas, "Price trailing for financial trading using deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 2837–2846, Jul. 2021.

[23] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[24] P. Tiwari, A. Lakhan, R. H. Jhaveri, and T.-M. Gronli, "Consumer-centric Internet of Medical Things for cyborg applications based on federated reinforcement learning," *IEEE Trans. Consum. Electron.*, early access, Feb. 7, 2023, doi: 10.1109/TCE.2023.3242375.

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[26] T. V. Nguyen et al., "A novel decentralized federated learning approach to train on globally distributed, poor quality, and protected private medical data," *Sci. Rep.*, vol. 12, no. 1, p. 8888, 2022.

[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[28] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local SGD with lower communication complexity," 2019, *arXiv:1912.12844*.

[29] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[30] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[31] J. Konečný and P. Richtárik, "Semi-stochastic gradient descent methods," 2013, *arXiv:1312.1666*.

[32] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Adv. neural Inf. Process. Syst.*, vol. 26, 2013.

[33] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate Newton-type method," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.

[34] H. M. Markowitz and G. P. Todd, *Mean-Variance Analysis in Portfolio Choice and Capital Markets*, vol. 66. Hoboken, NJ, USA: Wiley, 2000.

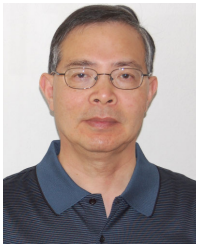
[35] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics: Methodology and Distribution*. Springer, 1992, pp. 196–202.

Lei Zhao (Member, IEEE) received the B.S. and M.A.Sc. degrees in computer science and technology from Xidian University, Xi'an, China, in 2015 and 2018, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 2023.

He is currently an Instructor with the Department of Electrical and Computer Engineering, University of Victoria.

Lin Cai (Fellow, IEEE) has been with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, since 2005, where she is currently a Professor. Her research interests include communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things (IoT).

Ms. Cai is an NSERC E.W.R. Steacie Memorial Fellow, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, and a member of the Royal Society of Canada's College of New Scholars, Artists, and Scientists.



Wu-Sheng Lu (Life Fellow, IEEE) received the B.Sc. degree in mathematics from Fudan University, Shanghai, China, in 1964, and the M.S. degree in electrical engineering and the Ph.D. degree in control science from the University of Minnesota, Minneapolis, MN, USA, in 1983 and 1984, respectively.

Since 1987, he has been with the University of Victoria, Victoria, BC, Canada, where he is currently a Professor Emeritus. He is the coauthor with A. Antoniou of *Two-Dimensional Digital Filters* (Marcel Dekker, 1992) and *Practical Optimization: Algorithms and Engineering Applications* (Second Edition, Springer, 2021), and with E. K. P. Chong and S. H. Zak of *An Introduction to Optimization* (Fifth Edition, Wiley, 2023).