Denoising and Adaptive Online Vertical Federated Learning for Sequential Multi-Sensor Data in IIoT

Heqiang Wang, Xiaoxiong Zhong, Kang Liu, Fangming Liu, Weizhe Zhang

***Abstract*—With the advancement of computational capabilities in edge devices such as intelligent sensors in the Industrial Internet of Things (IIoT), these sensors evolving beyond simple data collection to support complex computational tasks. This advancement provides new opportunities for adopting distributed learning approaches in IIoT. In this study, we focus on enhancing learning performance in an industrial assembly line scenario where multiple distributed sensors sequentially collect real-time data with distinct feature spaces. However, existing research lacks an online distributed learning framework tailored for such IIoT settings. To address this gap, we propose the Denoising and Adaptive Online Vertical Federated Learning (DAO-VFL) algorithm, a novel algorithm that leverages the computing potential of edge sensors while addressing key challenges such as communication overhead and data privacy. DAO-VFL effectively manages continuous data streams and adapts to shifting learning objectives. Furthermore, it can address critical challenges prevalent in industrial environment, such as communication noise and heterogeneity of sensor capabilities. To support the proposed algorithm, we provide a comprehensive theoretical analysis, highlighting the effects of noise reduction and adaptive local iteration decisions on the regret bound. Experimental results on two real-world datasets further demonstrate the superior performance of DAO-VFL compared to benchmarks.**

***Index Terms*—Industrial Internet of Things, Vertical Federated Learning, Online Learning, Deep Reinforcement Learning, Communication Noise Reduction, Federated Learning for Edge AI**

I. Introduction

Recent advancements in communication technologies and the proliferation of intelligent edge devices, coupled with the rapid pace of industrial informatization, have driven the widespread adoption of the Industrial Internet of Things (IIoT) [1]. This growth is largely attributed to IIoT's potential to significantly enhance productivity and efficiency across various industries. As we move forward into future industrial revolutions, particularly Industry 4.0 [2], the IIoT is expected to play a pivotal role in the development of new applications such as smart manufacturing, smart factories, smart transportation, and smart healthcare. To enable intelligent services and applications within the IIoT ecosystem, artificial intelligence (AI) techniques, particularly machine learning (ML) and deep learning (DL), are widely used to train models on industrial data. Traditionally, this training has been conducted in centralized cloud environments or data centers. However, this approach faces significant challenges as IIoT data volumes continue to grow. The need to transfer large volumes of IIoT data to centralized servers for model training demands substantial network bandwidth and introduces considerable communication overhead, which is impractical for time-sensitive IIoT applications like autonomous driving [3] and real-time healthcare [4]. Moreover, uploading sensitive data to the cloud increases the risk of privacy breaches. In response to these challenges, distributed learning based on edge devices, particularly federated learning (FL) [5], has gained attention as a promising solution. FL offers a more cost-effective and privacy-preserving alternative for deploying intelligent IIoT applications in a distributed manner, minimizing the need for data transfer while ensuring that sensitive information remains processed locally.

H. Wang, X. Zhong, K. Liu, F. Liu and W. Zhang are with Peng Cheng Laboratory, Shenzhen, 518066, China.

Corresponding Authors: Xiaoxiong Zhong, Kang Liu.

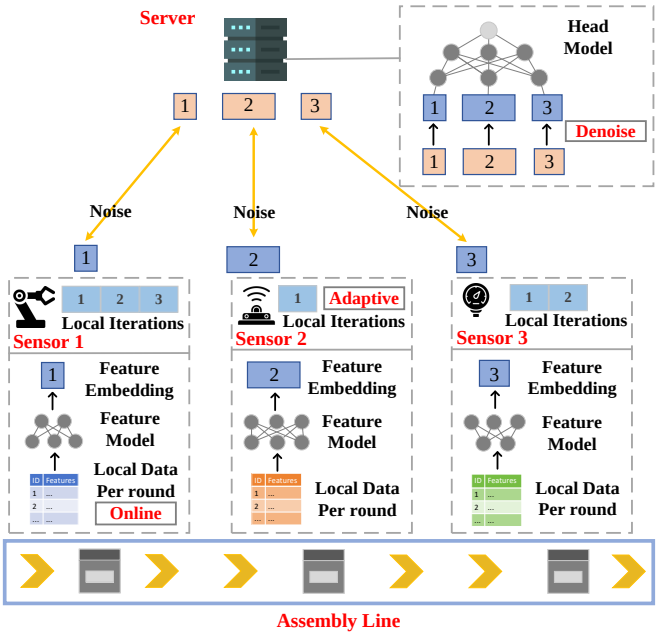


Fig. 1: DAO-VFL for IIoT-Based Assembly Line

The IIoT encompasses a broad array of application scenarios, each presenting its own set of problems, challenges, and potential solutions. In this work, we focus on an industrial assembly line scenario, as illustrated in Fig. 1, where multiple sensors collaborate for tasks like quality control and fault detection. As sensors become increasingly advanced, their roles have expanded beyond simple data collection and transmission to encompass more complex tasks such as model training and inference. In industrial assembly lines, numerous sensors are distributed along the line. Due to variations in

the types and locations of these sensors, the data they collect for a product moving through the assembly line can be seen as different features of the same dataset. This setup aligns perfectly with the concept of vertical federated learning (VFL) [6], [7], where participants have distinct feature spaces but share the same set of data samples. The core idea behind VFL is to partition the DNN into distinct segments, each trained separately by individual sensors and the server, ensuring that raw data remains at the sensors. The DNN is divided into a head classifier and multiple vertically separable feature extractors, one for each sensor. Instead of transmitting raw data to the server, each sensor processes its local data through its own feature extractor, generating a feature embedding, which is then sent to the server. The server's head classifier then processes the feature embeddings from all the sensors to derive the final results.

To date, the deployment of VFL frameworks in IIoT-based assembly line scenarios has not been thoroughly explored in existing research. In addition to addressing this gap, we also consider three key challenges that are commonly encountered in such industrial environments. First, since sensor data in industrial assembly line is collected in real-time and may not be fully available before training starts, the algorithm must be designed using an online learning approach, unlike traditional offline learning methods that rely on a static datasets. Second, communication between sensors over wireless networks in industrial environments often experiences interference and is subject to various levels of noise [8]. Therefore, it is critical to develop effective noise reduction techniques to mitigate these effects, which would enhance the overall learning performance. Third, considering the varying computational capabilities and network conditions of sensors in industrial assembly lines [9], and in order to achieve synchronized update, it is crucial to design algorithms that determine the appropriate number of local iterations for each sensor. This approach helps reduce overall latency while enhancing learning performance.

Building on the gaps and challenges discussed above, this work introduces the concept of Denoising and Adaptive Online Vertical Federated Learning (DAO-VFL) within an IIoT based assembly line scenario. Unlike previous VFL studies, our approach directly addresses practical issues encountered in IIoT environments. It emphasizes the integration of Denoising Adaptive and Online mechanisms within the VFL framework, which have not been systematically explored in prior VFL research. Our primary contributions can be summarized as follows:

1) We formulate the multi-sensor distributed learning problem in IIoT-based assembly lines as an online VFL problem. Building on this problem, and addressing challenges such as communication noise and sensor heterogeneity in industrial environments, we propose the DAO-VFL algorithm, which can facilitate practical online training by incorporating noise reduction techniques and adaptive local iteration decisions.

2) We conduct a comprehensive theoretical analysis of the DAO-VFL algorithm, accounting for the impacts of communication noise and adaptive local iteration decisions. The derived regret bound highlights these impacts and provides insights into selecting appropriate local iteration decisions for each sensor.

3) To achieve noise reduction, we incorporated a denoising autoencoder seamlessly into the training process. To determine the local iteration decisions for each sensor, we formulated an optimization problem that accounts for learning performance, overall latency, and local iteration disparities. This problem is solved using a deep reinforcement learning approach, to derive the local iteration decisions for each sensor at every global round.

4) In the experimental section, we evaluated the DAO-VFL algorithm using two datasets: the widely used CIFAR-10 dataset and the real-world IIoT-based C-MAPSS dataset for residual life prediction. Our results not only assess the overall performance of the DAO-VFL algorithm but also provide a detailed analysis of its noise reduction capabilities and adaptive local iteration decision-making mechanisms.

The rest of this paper is organized as follows. Section II reviews related works on FL for IIoT, VFL, and online FL. Section III introduces the system model and problem formulation. In Section IV, we introduce the specific process of DAO-VFL algorithm. Section V presents the regret analysis of DAO-VFL. Section VI formulates and addresses the optimization problem for determining the local iterations decision for each sensor. The experimental results of DAO-VFL are presented in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

A. FL for Industrial Internet of Things

As the IIoT continues to evolve, the demand for increased intelligence within IIoT is becoming increasingly urgent. AI techniques, particularly DL, play a crucial role in enabling this intelligence and are widely utilized in IIoT environments. However, the inherent characteristics of IIoT, such as the large number of distributed computing nodes and complex network architectures [10], pose significant challenges to traditional centralized learning approaches, making them inadequate for these scenarios. To address these challenges, FL has emerged as a promising alternative, gaining significant attention for its ability to better align with the unique characteristics of IIoT systems. The works of [11], [12] provide comprehensive overviews of FL for IIoT. Unlike conventional FL, IIoT-based FL is typically tailored to specific application domains, such as smart manufacturing [13], smart transportation [14] and smart grid [15]. Additionally, IIoT-based FL often addresses critical and urgent industrial issues, including anomaly detection [16], network prediction [17] and intrusion detection [18], [19]. However, research in IIoT-based FL remains in its early stages, with most current studies focusing on ideal conditions and overlooking critical challenges specific to IIoT environments. These challenges include noise in the industrial environment [20], limited data storage capacity of edge devices [21], and the heterogeneity of IIoT edge devices [9]. In this work, we

systematically investigate FL deployment within IIoT-based assembly line scenarios. We aim to address and provide solutions to some of the critical challenges posed by real industrial environments, thereby advancing the practical applicability of FL in IIoT environment.

B. Vertical Federated Learning

In recent years, VFL has garnered significant attention. Originally introduced by [22], VFL operates on vertically partitioned data, setting it apart from the concept of horizontal FL (HFL) [23]. Comprehensive surveys such as [6], [7] have further expanded on the scope of VFL. Unlike horizontal FL, VFL faces its own set of unique challenges. Some studies [24]–[26] have focused on optimizing data utilization to enhance the effectiveness of the joint model in VFL. Other research efforts [27], [28] have concentrated on creating privacy-preserving protocols to mitigate the risks of data leakage. Additionally, efforts have been made to reduce communication overhead, either by incorporating multiple local updates per iteration [29], [30] or by employing data compression techniques [31], [32]. VFL's practical benefits, particularly in enabling data collaboration among diverse institutions across various industries, have heightened interest from both academic and industrial communities. VFL has found applications in a wide range of fields, including recommendation systems [33], finance [34], and healthcare [35]. As VFL is applied in increasingly diverse scenarios, scalability has become a key research concern. Work [36] improves scalability by using a mutual information-based party-level evaluation to select informative participants. Work [37] proposes a practical private sample alignment protocol, which supports multi-client scenarios through a lightweight delegated private set intersection scheme and a threshold-based mechanism, enabling reliable sample alignment at scale without compromising privacy. However, the VFL approaches outlined above exhibit two significant limitations. First, they primarily focus on offline learning with static datasets, overlooking the dynamic nature of stream data. Second, they do not adequately address the unique challenges inherent to real IIoT scenarios. These limitations form the core issues that this work seeks to resolve.

C. Online Federated Learning

Online learning is tailored to process data sequentially and update models incrementally, making it particularly effective for applications where data continuously arrives and models need to adapt to new patterns in real-time [38]. These methods are computationally efficient and have the distinct advantage of not needing the entire dataset to be available at the outset, making them ideal for scenarios with limited memory resources. Consider the FL scenario, online federated learning (OFL) [39] has emerged as an innovative paradigm that extends the principles of online learning to a network of multiple learners or agents. The key difference between OFL algorithms and traditional FL algorithms lies in the goal of local updates. While traditional FL algorithms aim to find a single global model that minimizes a global loss function, OFL algorithms focus on identifying a sequence of global models that minimize cumulative regret. Recent years have seen limited research on OFL. For instance, the authors of [40] propose a communication-efficient OFL method that balances reduced communication overhead with strong performance. Similarly, [41] introduces FedOMD, an OFL algorithm tailored for uncertain environments, capable of processing streaming data from devices without relying on statistical assumptions about loss functions. While the aforementioned studies primarily focus on the HFL scenario, work [42] addresses the VFL context by proposing an online VFL architecture tailored to cooperative spectrum sensing problems, achieving a sublinear regret rate $\mathcal{O}(1/\sqrt{T})$. However, these studies are based on idealized scenarios and do not address the challenges of applying online VFL in real industrial scenarios, such as noise interference and device heterogeneity. Non-idealized scenarios can also impact the algorithm's performance, potentially preventing it from achieving a sublinear regret rate.

D. Noise Mitigation Strategies

FL in IIoT-based scenarios is inherently vulnerable to various types of noise, which can be broadly categorized into three types. First, raw data noise may occur during sensor data collection due to environmental interference or device limitations [20], and can be mitigated using signal processing techniques such as the Wiener filter [43]. Second, even when raw data is reliable, label noise may arise due to inaccurate or corrupted annotations [44]. This can be addressed using noise adaptation layers [45] or loss correction methods [46]. Third, communication noise can occur during data transmission over unstable IIoT networks, which may degrade the learning performance [47]. To address communication noise, various methods have been proposed in the context of HFL. For instance, [48] introduces a robust FL design using regularization and successive convex approximation to mitigate communication noise. The FedNMUT algorithm [49] incorporates gradient tracking in decentralized FL with graph-based communication, improving robustness against noisy channels. In [50], a median-based gradient clipping technique is proposed to suppress outlier noise while preserving essential gradient information. However, these methods are designed for HFL settings, where communication typically involves model parameters or gradients. In contrast, VFL requires the exchange of intermediate feature representations, making direct adaptation of HFL-based noise mitigation strategies ineffective. Consequently, the core objective in VFL under communication noise is to reconstruct clean feature representations. This can be achieved using techniques such as Denoising Autoencoder (DAE) [51] or Transformer-based denoising method [52]. Considering the trade-off between performance and deployment efficiency, our work adopts a DAE-based approach as an solution.

III. System Model

Before presenting the details of the system model, we first summarize the essential notations used in DAO-VFL in Table I. Consider an IIoT assembly line system consisting of

a single server and K smart sensors. Each sensor collects distinct system parameters along the assembly line, such as temperature, pressure, humidity, as illustrated in Fig. 1. These parameters are treated as independent features within a single data sample, reflecting the unique identities and locations of each sensor. As a result, the distributed training process involving these sensors falls under the category of a VFL problem, since each sensor operates within a distinct feature space. During the operation of the assembly line, the sensors continuously gather new data over time, with the timeline divided into discrete periods represented as $t = 1, 2, ..., T$.

TABLE I: Key Notations of DAO-VFL

Symbol	Semantics
K	The number of sensors
T	The number of global rounds
N	The number of data samples collected by sensors
P	The dimension of data samples
θ_k	The feature model
θ_0	The head model
Θ	The overall model
$E_{t,k}$	The adaptive local iterations
$h_k(\cdot)$	The original feature embedding
$\tilde{h}_k(\cdot)$	The noise feature embedding
$\hat{h}_k(\cdot)$	The denoise feature embedding
Φ	The model representation
$\mathbf{G}^t$	The stacked partial derivatives
η	The learning rate

In each time period, each sensor $k \in \mathcal{K}$ collects a local training dataset consisting of N data samples, represented as $\mathbf{x}_k^t \in \mathbb{R}^{N \times P_k}$, where P_k is the dimension of the raw data collected by sensor k. The individual data samples, denoted as $x_k^{t,n}$ for all sensors, are gathered simultaneously and linked to a common label, $y^{t,n}$, which indicates, for instance, the product's conformity. The collective training dataset at period t is denoted as $\mathbf{x}^t \in \mathbb{R}^{N \times P}$, where $P = \sum_{k=1}^{K} P_k$. It should be emphasized that although $\mathbf{x}^t$ represents the entirety of data collected at a given period, these data are gathered and utilized locally by each sensor individually and are not uploaded to the server during the training process.

In the VFL framework, each sensor trains a distinct local feature model parameterized by θ_k to process its collected raw data. Concurrently, the server trains a server head model represented by the parameter θ_0. The combined parameters of the entire model are denoted as $\Theta = [\theta_0^\top, \theta_1^\top, ..., \theta_K^\top]^\top$. The raw data $x_k^{t,n}$ collected by each sensor is processed through its local feature model, a process referred to as feature extraction, and represented as $h_k(\theta_k; x_k^{t,n})$. This operation transforms high-dimensional raw data into low-dimensional feature representations, which facilitate the learning process of the server head model. Based on these definitions, the loss function for the collective training dataset at period t is expressed as follows:

$$F_t(\Theta; \mathbf{x}^t, \mathbf{y}^t) = \frac{1}{N} \sum_{n=1}^{N} l_t(\theta_0, \{h_k(\theta_k; x_k^{t,n})\}_{k=1}^{K}; y^{t,n}) \quad (1)$$

where $l_t(\cdot)$ represents the loss function for the single data sample. For simplicity, the feature embedding of the dataset is denoted as $\mathbf{x}_k^t$ as $h_k(\theta_k; \mathbf{x}_k^t)$, which is often abbreviated as $h_k(\theta_k; \mathbf{x}_k^t) = h_k^t(\theta_k)$. Additionally, we assign $k = 0$ to the server, defining $h_0(\theta_0) = \theta_0$. In this context, $h_0(\theta_0)$ refers to the head model rather than the feature embedding. Furthermore, the overall loss function is expressed as $F_t(\Theta) = F_t(h_0(\theta_0), h_1(\theta_1), ..., h_K(\theta_K))$.

Since the training process relies on a dynamically collected, real-time data instead of a static dataset, an online learning approach becomes essential. Let the overall model at each period be represented as $\Theta^1, \ldots, \Theta^T$. The learning regret, Reg_T is defined to quantify the gap between the cumulative loss incurred by the learner and the cumulative loss of an optimal fixed model in hindsight. Specifically:

$$\text{Reg}_T = \sum_{t=1}^{T} F_t(\Theta^t; \mathbf{x}^t, \mathbf{y}^t) - \sum_{t=1}^{T} F_t(\Theta^*; \mathbf{x}^t, \mathbf{y}^t) \quad (2)$$

Here, $\Theta^* = \arg\min_\Theta \sum_{t=1}^{T} F_t(\Theta; \mathbf{x}^t, \mathbf{y}^t)$ represents the optimal fixed model selected in hindsight. Our objective is to minimize the learning regret, which equates to minimizing the cumulative loss. Importantly, if the learning regret grows sublinearly with respect to T, it indicates that the online learning algorithm can progressively reduce the training loss asymptotically.

The fixed optimal strategy in hindsight refers to a strategy determined by a centralized entity with full prior knowledge of all per-round loss functions. In our problem, achieving such an optimal strategy would require access to future information, including upcoming data collection for all rounds. However, this information is inherently unpredictable due to its randomness. As a result, the complete loss functions are also unknown at the outset and evolve dynamically over time. Therefore, regret just serves as a metric to quantify the performance gap between the proposed algorithm and the theoretical optimal strategy in our theoretical analysis. For experimental validation, we evaluate the learning performance of the proposed algorithm using metrics such as test loss and test accuracy.

IV. Denoising and Adaptive Online Vertical Federated Learning

In this section, we introduce the details of the DAO-VFL algorithm. To provide an overview, we summarize the challenges faced by DAO-VFL and the corresponding solutions in Fig.2. Unlike traditional VFL approaches, the inherently complex environments of IIoT scenarios necessitate addressing three additional challenges. First, sensors in industrial wireless network environments are subject to noise interference [53], which can negatively impact training performance. Therefore, it is crucial to design effective and easily deployable noise reduction methods to mitigate noise effects. Second, the varying positions of sensors along the assembly line, coupled with differences in their computational capacities and channel conditions, add further complexity. Achieving synchronized updates within the learning system requires different sensors to perform varying numbers of local iterations. Developing an adaptive local iteration strategy that enhances overall training efficiency and performance is also a crucial aspect in this

work, as it has not been adequately explored before. Third, sensor data in industrial assembly lines is collected in real-time. As a result, the algorithm must align with an online learning approach to effectively handle the dynamic dataset.

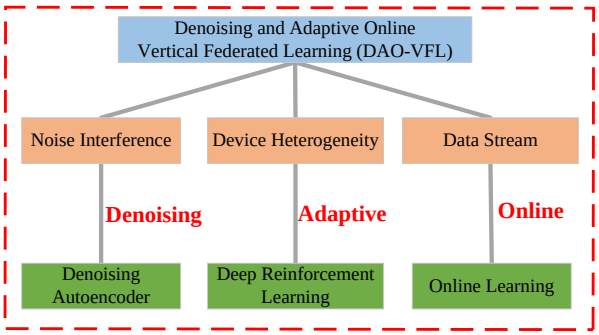


Fig. 2: Challenges and Solutions in DAO-VFL

Next, we will outline the detailed process of the DAO-VFL algorithm, beginning with the introduction of some additional concepts. For convenience, a single time period in our framework is also defined as one global training round. During each global round $t \in \mathcal{T}$, where $\mathcal{T} = \{0, 1, 2, \ldots, T-1\}$, both the sensors and the server execute a specified number of local training iterations, represented by the parameter $E_{t,k}$. This parameter is dynamically determined based on the real-time status of the sensors in the current global round. The approach for deciding the number of local iterations for each sensor will be discussed in the later section. Each local training iteration is indexed as $\tau = 0, 1, 2, ..., E_{t,k}$. Notably, the DAO-VFL algorithm is designed for an online synchronized scenario, even though sensors perform varying numbers of local training iterations. The detailed flow of the DAO-VFL algorithm is outlined in Algorithm 1. Subsequently, we provide an in-depth explanation of the key steps.

1) Feature Embedding Extracting: At the start of each global round t, each sensor k in the assembly line incrementally collects the new training dataset, $\mathbf{x}_k^t$ and $\mathbf{y}^t$. It is important to note that the data samples $\mathbf{x}_k^t$ are collected independently by each sensor in chunks, with each sensor capturing only one or several features. These data samples are then processed by each sensor's local feature model, $\theta_k^{t,0}$ to generate feature embeddings $h_k^t(\theta_k^{t,0})$. The initial feature model $\theta_k^{t,0}$ at current global round is inherited from the previous global round. Following this, each sensor uploads its feature embeddings to the server to further obtain the model representation.

2) Feature Embedding Denoising: Given the interference present in complex wireless network environments within the IIoT based assembly line scenario, feature embeddings transmitted through the wireless network are inevitably affected by noise. Suppose we define $\tilde{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$ as the noisy feature embedding. To address this, it is essential to develop effective noise reduction techniques to obtain the denoised feature embedding, denoted as $\hat{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$. To achieve this, we employ Denoising Autoencoders (DAE) for noise reduction on the server side. Given the server's robust computational capabilities, this process does not impose a significant computational burden. DAE is a specialized type of neural network designed to learn robust feature representations by reconstructing input data from a corrupted version. Unlike traditional autoencoder, which focus on compressing and then reconstructing the original input, DAE is explicitly trained to remove noise from the input, thereby revealing the underlying structure of the data.

Algorithm 1 DAO-VFL

Input: Local datasets $\{\mathbf{x}_k^t\}_{k\in\mathcal{K}}^{t\in\mathcal{T}}$, Label sets $\{\mathbf{y}^t\}^{t\in\mathcal{T}}$, The number of client K, The number of global rounds T.
Output:The overall model $\{\Theta^t\}^{t\in\mathcal{T}}$ each global round.
Initialize: The initial feature model $\theta_k^{t=0,\tau=0}$ for all sensors k and the initial head model $\theta_0^{t=0,\tau=0}$ for server.
for $t = 0, 1, 2, ..., T-1$ **do**
  for Sensor $k = 1, 2, ..., K$ in parallel **do**
    if $\tau = 0$ **then**
      Collects new data samples $\mathbf{x}_k^t$.
      Gets the feature embedding $h_k(\theta_k^{t,0}; \mathbf{x}_k^t)$.
      Sends $h_k(\theta_k^{t,0}; \mathbf{x}_k^t)$ to server.
    end if
  end for
  Server collects noise feature embedding $\tilde{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$ from sensors.
  Server applies the denoising autoencoders and get the denoised embedding $\hat{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$.
  Server collects model representation $\hat{\Phi}^{t,0}$.
  Server sends $\hat{\Phi}^{t,0}$ to all sensors.
  for $k = 0, 1, 2, ..., K$ in parallel **do**
    Each sensor k decide the $E_{t,k}$ via Algorithm 2.
    for $\tau = 1, 2, ..., E_{t,k} - 1$ **do**
      Gets $\hat{\Phi}_k^{t,\tau} \leftarrow \left\{\hat{\Phi}_{-k}^{t,0}; h_k^t(\theta_k^{t,\tau})\right\}$.
      Updates feature or head model $\theta_k^{t,\tau+1}$.
    end for
    Inherit the model $\theta_k^{t+1,0} \leftarrow \theta_k^{t,E_{t,k}}$ for next round.
  end for
end for

The noise reduction process operates as follows: once the server receives the noisy feature embedding, it first uses an encoder to map the embedding from a high-dimensional space to a lower-dimensional latent space. The decoder then reconstructs the feature embedding, mapping it back to the original space. During the initial T_{dl} global rounds, also referred to as the denoising learning period, it is assumed that the original feature embeddings from the sensors are available to the server. Achieving optimal DAE performance requires close cooperation between the sensors and the server and is strongly influenced by the T_{dl}. Theoretically, a larger T_{dl} allows for more effective noise reduction. Based on the network conditions, the server can employ either a single DAE or multiple DAEs to denoise the noisy feature embeddings received from the sensors. Then for the paired feature embedding $\left\{\tilde{h}_k^t(\theta_k^{t,0}), h_k^t(\theta_k^{t,0})\right\}$, the training target of DAE is:

$$\arg\min_{\theta_d} \mathbb{E}_{\left\{\tilde{h}_k^t(\theta_k^{t,0}), h_k^t(\theta_k^{t,0})\right\}} \left\{Ls\left(\Lambda_{\theta_d}(\tilde{h}_k^t(\theta_k^{t,0})), h_k^t(\theta_k^{t,0})\right)\right\} \tag{3}$$

where $\mathbb{E}$ denotes the expectation operator, $\Lambda_{\theta_d}(\cdot)$ represents

the trainable DAE, θ_d refers to the weights of the DAE, and $Ls(\cdot)$ is the loss function used to evaluate the DAE's learning performance. Once processed through the trained DAE, the noisy feature embedding $\tilde{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$ is transformed into the denoised feature embedding $\hat{h}_k(\theta_k^{t,0}; \mathbf{x}_k^t)$. In later sections, we will validate the effectiveness of noise reduction through both theoretical analysis and experimental results.

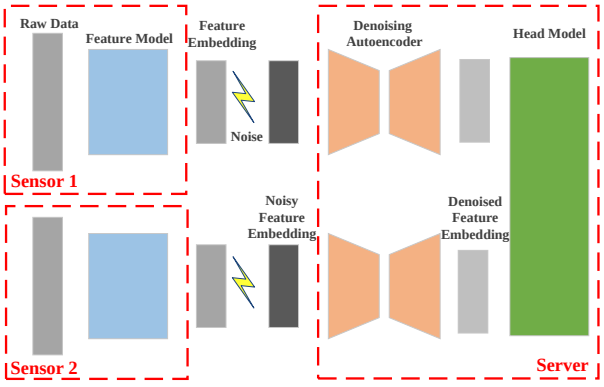


Fig. 3: The Architecture of DAO-VFL Noise Reduction

3) Model Representation Distributing: After collecting all the denoised feature embedding, the server compiles the model representation $\hat{\Phi}^{t,0}$, which includes the server head model and all denoised feature embeddings. The model representation is formally defined as follows:

$$\hat{\Phi}^{t,0} \leftarrow \left\{\theta_0^{t,0}, \hat{h}_1^t(\theta_1^{t,0}), \cdots, \hat{h}_k^t(\theta_k^{t,0}), \cdots, \hat{h}_K^t(\theta_K^{t,0})\right\} \quad (4)$$

The server then distributes the model representation $\hat{\Phi}^{t,0}$ to all sensors. In this distribution process, the impact of noise is not considered. This is because servers can utilize directional antennas to focus transmission beams towards specific sensors, ensuring reliable communication.

4) Feature and Head Model Updating: Each sensor k and the server employ the received model representation $\hat{\Phi}^{t,0}$ to update their respective feature or head models over for $E_{t,k}$ iterations. Here, $E_{t,k}$ representing the adaptive local iteration decisions, varies across sensors. The updates follow this formula for all $\tau = 0, ..., E_{t,k} - 1$:

$$\theta_k^{t,\tau+1} = \theta_k^{t,\tau} - \eta \nabla_k F_t\left(\hat{\Phi}_{-k}^{t,0}, h_k^t(\theta_k^{t,\tau})\right) \quad (5)$$

Here, $\hat{\Phi}_{-k}^{t,0}$ is the collection of feature embeddings from all sensors and the head model from server, excluding sensor k. Notably, although the server and sensors may have different numbers of local iterations $E_{t,k}$, they operate within the same wall clock time, ensuring that the training process remains synchronized. Furthermore, since the training is based on a dynamic dataset rather than a subset of a static dataset, both the feature and head models are updated using online gradient descent (OGD) [54] instead of traditional offline gradient descent methods. To enhance the understanding of the DAO-VFL process described earlier, we also present the DAO-VFL noise reduction model architecture in Fig. 3.

During inference, the process mirrors that of training: data collected by each sensor is first processed through its local feature model, and the resulting feature embeddings are uploaded to the server. The server then completes the inference using the head model. Notably, unlike training, there is no need to redistribute the feature embeddings back to the sensors during inference.

Complexity Analysis: The computational complexity of the DAO-VFL algorithm is $\mathcal{O}(T \times K \times E_{\max} \times N \times V)$, where V denotes the per-sample cost of a forward-backward pass. The algorithm supports high parallelism across clients, and its total cost scales linearly with the number of global rounds (T), clients (K), and the size of local dataset (N), making it computationally efficient and scalable for large-scale VFL deployments.

Remark 1 (Privacy Concern): Consistent with the scenarios outlined in previous VFL studies [31], [42], we assume that all sensors and the server have access to label information. Furthermore, the IIoT based assembly line scenario is considered to operate in a low-risk environment, where label sharing between sensors and the server does not pose significant privacy concerns.

Remark 2 (Noise Concern): To train the DAE, clean data are typically required. While this could theoretically be achieved by deploying additional costly redundant sensors or using dedicated communication lines [55], however, such approaches are not feasible for widespread, long-term deployment. An alternative method is to rely solely on noisy data for training. Notably, certain DAE-based noise reduction techniques, such as Noise2Noise [56], Noise2void [57], are specifically designed to function in scenarios where clean data is unavailable. These approaches can also be integrated into our algorithm. However, due to the complexity of the current algorithm, we do not explore these additional extensions in this work.

Remark 3 (Scalability Concern): DAO-VFL demonstrates strong scalability, having been enhanced from the foundation of online vertical federated learning to address challenges such as communication noise and sensor heterogeneity commonly encountered in industrial scenarios. Building upon this foundation, DAO-VFL can further improve system performance by incorporating targeted modules. For example, integrating a client selection mechanism enables the identification of optimal feature contributors in each training round, thereby enhancing learning effectiveness while reducing communication overhead. Additionally, the inclusion of an asynchronous update module allows clients to perform updates independently, improving training efficiency by maximizing time utilization across heterogeneous clients.

V. Regret Analysis

In this section, we provide a comprehensive regret analysis of the proposed DAO-VFL algorithm. To support our analysis, we begin by introducing several additional definitions. Specifically, we define $\hat{\mathbf{G}}^t$ as the stacked partial derivatives that incorporate the effects of denoising at global round t:

$$\hat{\mathbf{G}}^t := \left[\sum_{\tau=0}^{E_{t,0}-1} \nabla_0 F_t(\hat{\Phi}_0^{t,\tau}), \ldots, \sum_{\tau=0}^{E_{t,K}-1} \nabla_K F_t(\hat{\Phi}_K^{t,\tau})\right] \quad (6)$$

where $\hat{\Phi}_k^{t,\tau} = (\hat{\Phi}_{-k}^{t,0}, h_k^t(\theta_k^{t,\tau}))$. Using $\hat{\mathbf{G}}^t$, we can define the updates to the global model during a global round t with the following equation:

$$\Theta^{t+1,0} = \Theta^{t,0} - \eta \hat{\mathbf{G}}^t \tag{7}$$

For comparison, we also define $\mathbf{G}^t$ as the stacked partial derivatives at global round t without accounting for the impact of noise:

$$\mathbf{G}^t := \left[\sum_{\tau=0}^{E_{t,0}-1} \nabla_0 F_t(\Phi_0^{t,\tau}), \ldots, \sum_{\tau=0}^{E_{t,K}-1} \nabla_K F_t(\Phi_K^{t,\tau}) \right] \tag{8}$$

For the subsequent theoretical analysis, we consider the overall gradient and model as D-dimensional vector. We denote an arbitrary vector element $d \in [1, D]$ of the overall gradient as $\mathbf{G}_{k,d}$, and the arbitrary vector element $d \in [1, D]$ of the overall model as $\Theta_{k,d}$.

Next, we will present the assumptions typically employed in the analysis of online convex optimization, as referenced in [58]. Some of these assumptions are defined at the vector element level and are specifically tailored to align with the requirements of our proof within the context of online learning in VFL scenarios [42].

Assumption 1. *For any $(\boldsymbol{x}^t; \boldsymbol{y}^t)$, the loss function $F_t(\Theta; \boldsymbol{x}^t; \boldsymbol{y}^t)$ is convex with respect to Θ and differentiable.*

Assumption 2. *The loss function is L-Lipschitz continuous, the partial derivatives satisfies: $\|\nabla_k F_t(\Theta)\|^2 \leq L^2$.*

Assumption 3. *The partial derivatives corresponding to a consistent loss function satisfy the following condition:*

$$\left\| \boldsymbol{G}_k^{t,\tau'} - \boldsymbol{G}_k^{t,\tau} \right\| \leq \lambda \left\| \theta_k^{t,\tau'} - \theta_k^{t,\tau} \right\|$$

In the context of the online learning scenario, where the loss function changes over time, we use t to indicate that gradients and models correspond to the same loss function. To differentiate their origins from various local iterations, we use τ' and τ, respectively.

Assumption 4. *The arbitrary vector element d in the overall model $\Theta_{k,d}$ is bounded as follows: $|\Theta_{k,d}| \leq \rho$.*

Next, we will provide the assumption for the overall gradient after noise reduction, comparing it to the original overall gradient.

Assumption 5. *The arbitrary vector element d in the overall gradient, adjusted through a denoising method, has a bounded range range of variation as: $\left| \hat{\boldsymbol{G}}_{k,d}^{t,\tau} - \boldsymbol{G}_{k,d}^{t,\tau} \right| \leq \beta_d$.*

Assumption 1 ensures the convexity of the function, allowing us to leverage the associated properties of convexity. Assumption 2 constrains the magnitude of the loss function's partial derivatives. Assumption 3 guarantees that the variation in partial derivatives remains within a specific range, consistent with the model's variation across different local iterations under a consistent loss function. Assumption 4 defines the allowable range for any vector element within the overall model. Finally, Assumption 5 bounds the impact of the application of noise reduction methods on the overall model relative to the original model. Based on the above assumptions, we can derive the main result, presented as Theorem 1, as follows:

Theorem 1. *Under Assumption 1-5, DAO-VFL with adaptive local iterations decisions $E_{t,k} \geq 1$, while considering the impact of denoising, achieves the following regret bound:*

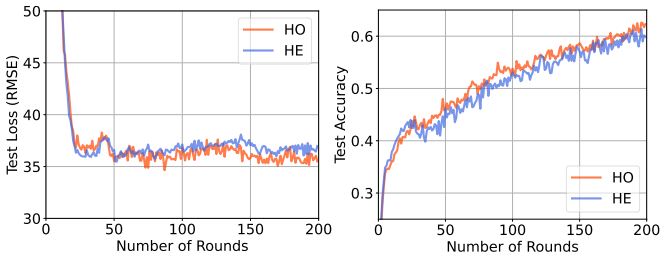
$$\begin{aligned} Reg_T &= \sum_{t=1}^{T} \mathbb{E}_t \left[F_t(\Theta^{t,0}; \boldsymbol{x}^t; \boldsymbol{y}^t) \right] - \sum_{t=1}^{T} F_t(\Theta^*; \boldsymbol{x}^t; \boldsymbol{y}^t) \\ &\leq \frac{\left\| \Theta^{1,0} - \Theta^* \right\|^2}{2\eta E_{min}} + \frac{\eta T D \beta_d^2}{E_{min}} + \frac{\eta T E_{max} L^2 K}{E_{min}} \\ &+ 2DT\rho(\eta\lambda E_{max} L + \beta_d) \\ E_{max} &= \max_{t \in \mathcal{T}, k \in \mathcal{K}} E_{t,k}, \quad E_{min} = \min_{t \in \mathcal{T}, k \in \mathcal{K}} E_{t,k} \end{aligned} \tag{9}$$

Proof. The proof can be found in Appendix B. □

Based on Theorem 1, we derive the following findings: First by setting $\eta = \mathcal{O}(1/\sqrt{T})$, the DAO-VFL can achieve the regret bound of $\mathcal{O}(\sqrt{T} + T\beta_d)$ over T time rounds. The regret bound is affected by the term $\mathcal{O}(T\beta_d)$, which reflects the effectiveness of the noise reduction method. Second, by incorporating Assumption 5 and defining β_n as the bounded range of variation between the noisy gradient and the original gradient, we can substitute β_d with β_n in the regret bound from Theorem 1 to derive the regret bound in the presence of noise. It becomes clear that the primary factors influencing the regret bound are the magnitudes of β_n and β_d. If the noise reduction method is effective, it consistently leads to a tighter regret bound. This comparison will be demonstrated through experimental results in the later sections. Third, the tightness of the regret bound is also influenced by E_{max} and E_{min}. A smaller E_{max} and a larger E_{min} lead to a tighter bound. This highlights the importance of ensuring that all sensors are as similar as possible in terms of $E_{t,k}$ during each global round, aligning with the principle of fairness in the number of local iterations across all sensors. In the following, we will use simulations to further validate this finding.

Validation Study: To validate the impact of fairness in the number of local iterations $E_{t,k}$ across all sensors, we conducted simulations using the C-MAPSS and CIFAR-10 datasets under different $E_{t,k}$ patterns. Further experimental details are provided in the experimental section. We tested two patterns: **Homogeneity** (HO), where all sensors had the same $E_{t,k}$ value in each global round, and **Heterogeneity** (HE), where sensors had significantly different $E_{t,k}$ values in each global round. To ensure a controlled comparison, the total local iterations, $\sum_{k \in \mathcal{K}} E_{t,k}$ were kept constant across all global rounds for both patterns. The simulation results over 10 rounds are presented in Fig. 4(a) for C-MAPSS dataset and Fig. 4(b) for the CIFAR-10 dataset.

The experimental results reveal that maintaining similar $E_{t,k}$ values across sensors enhances learning performance, as reflected by higher test accuracy and lower test loss. This improvement is observed across both datasets. However, in real-world industrial scenarios, achieving identical $E_{t,k}$ values



(a) Test Loss with C-MAPSS (b) Test Accuracy with CIFAR-10

Fig. 4: Learning Performance of DAO-VFL with Different Local Iteration Decisions Patterns $(E_{t,k})$.

for all sensors at each global round can be challenging due to variations in sensors' computational and communication capabilities, particularly in synchronized training scenarios. In the following section, we will define the optimization problem specific to the IIoT-based assembly line scenario and utilize deep reinforcement learning to determine adaptive local iteration decisions for all sensors in each global round, taking sensor heterogeneity into account.

VI. Adaptive Local Iteration Decisions

Before detailing the adaptive local iteration decisions, we first summarize the key notations used to simulate the workflow of sensors in the IIoT-based assembly line scenario in Table II.

TABLE II: Key Notations in Adaptive Iteration Decisions

Symbol	Semantics
$\Upsilon^{co}_{t,k}$	The collection latency
$\Upsilon^{cm}_{t,k}$	The communication latency
$\Upsilon^{cp}_{t,k}$	The computation latency
W_{fe}	The number of weights in the feature embedding
$r_{t,k}$	The transmission rate
B	The total bandwidth
$g_{t,k}$	The channel gain
p_k	The transmission power
σ^2	The noise power
C	The number of CPU cycles per weight
W_{lc}	The number of weights in the local model
$f_{t,k}$	The CPU frequency
Υ_t	The total latency
H_t	The local iteration disparity

In the IIoT-based assembly line scenario, sensors are positioned at various locations along the assembly line. Each sensor undergoes three primary phases within a single global training round: local data collection, feature embedding upload, and local feature model update. The detailed timeline of these phases is illustrated in Fig. 5. In this problem, we disregard the time needed for the server to broadcast the model representation and omit the time spent on server-side noise reduction, as these factors are not directly relevant to the optimization problem and is not influenced by the sensors' capacities. Next, we will provide definitions for each of the three distinct phases in detail and present a comprehensive formulation of the optimization problem.

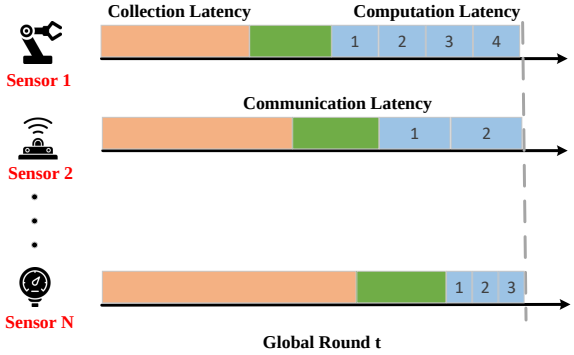


Fig. 5: Timeline of Sensors in One Global Round

1) Collection Latency: Given that our scenario is based on online learning, where training is conducted using real-time collected data rather than pre-selected batches from a static dataset, data collection latency becomes a crucial factor. Additionally, in the context of an assembly line, we consider the intervals between the sequential data collection times of each sensor. To simulate this sequential process characteristic of an assembly line, we define the **collection latency** as follows:

$$\Upsilon^{co}_{t,k} = \mu k + \mu_0 \tag{10}$$

For simplicity in the analysis, we assume that sensors collect data sequentially based on their index, such that sensors with smaller indices collect data earlier.

2) Communication Latency: In this scenario, sensors share bandwidth during the upload process, as they operate within a wireless network and transmit their feature embeddings to the server via this wireless network. The **communication latency** is equal to:

$$\Upsilon^{cm}_{t,k} = \frac{W_{fe}}{r_{t,k}} \tag{11}$$

where W_{fe} represents the number of weights in the feature embedding. We assume that the network bandwidth is equally distributed among all sensors. The transmission rate between sensor k and the server at round t, denoted by $r_{t,k}$, as:

$$r_{t,k} = \frac{B}{K} \log_2 \left(1 + \frac{g_{t,k} p_k}{\sigma^2}\right) \tag{12}$$

where B is the total bandwidth, $g_{t,k}$ represents the channel gain between sensor k and the server at global round t, p_k denotes the transmission power of sensor k, and σ^2 is the noise power.

3) Computation Latency: In each global round, the sensor performs at least one local update, with the number of updates depending on its specific conditions. The **computation latency** can be computed as follows:

$$\Upsilon^{cp}_{t,k} = \frac{E_{t,k} C W_{lc}}{f_{t,k}} \tag{13}$$

Here, C represents the number of CPU cycles required for sensor to update a single model weight, and W_{lc} denotes the number of weights in the local model. The CPU frequency

of sensor k during computation at round t equals $f_{t,k}$, which varies across sensors and ranges from $f_k^{\min}$ to $f_k^{\max}$.

Based on the definitions of latency provided above, the total latency for each global round t can be expressed as:

$$\Upsilon_t = \max_{1\leq k\leq K} \left(\Upsilon_{t,k}^{co} + \Upsilon_{t,k}^{cm} + \Upsilon_{t,k}^{cp}\right) \tag{14}$$

To ensure fairness and reduce disparities in local iteration decisions among sensors during each global round, as indicated by the theoretical analysis and validation study from previous section, we define the local iteration disparity for global round t as:

$$H_t = \sum_{k=1}^{K} \left|E_{t,k} - \bar{E}_t\right| = \frac{1}{K}\sum_{k=1}^{K} |KE_{t,k} - \sum_{k=1}^{K} E_{t,k}| \tag{15}$$

Defining H_t in this manner minimizes the disparity in local iterations among sensors.

Taking into account the definitions of total latency and the insights on disparity, it is clear that the optimization problem must strike a balance between test accuracy and these two additional components. Based on this, we define the optimization problem **P1** as follows:

$$\mathbf{P1}: \max_{\mathcal{E}} \mathbb{E}\left[\alpha_1 \sum_{t\in T} Acc(t) - \alpha_2 \sum_{t\in T} \Upsilon_t - \alpha_3 \sum_{t\in T} H_t\right] \tag{16}$$

where $\mathcal{E} = [E_{t,k}]$ represents a $T \times K$ matrix for each sensor's local iterations decision with $E_{t,k} \in [1, E_{max}]$, here E_{max} denotes the upper limit of local iterations. The parameters α_1, α_2, and α_3 control the relative importance of each objective, as determined by the designers. Our objective is to solve the optimization problem **P1**, Given the large number of parameters and their complex interdependencies, which make direct solutions challenging. We utilize Deep Reinforcement Learning (DRL) techniques [59], including the actor-critic method [60] and the Proximal Policy Optimization (PPO) algorithm [61], to solve the optimization problem **P1** and derive the adaptive local iteration decisions for each sensor.

The objective of the DRL agent is to find the best policy mapping a state to an action that maximizes the expected reward. In the following, we will provide a detailed explanation of the state space, action space, reward and training methodology relevant to the proposed DRL problem.

State: The state consists of the information uploaded by each sensor prior to the local feature model update phase of each global round. The state in the DRL framework is represented as vectors: the data collection latency vector for round t, $\hat{\Upsilon}_t^{co} = (\Upsilon_{t,1}^{co}, \dots \Upsilon_{t,K}^{co})$, the communication latency vector for round t, $\hat{\Upsilon}_t^{cm} = (\Upsilon_{t,1}^{cm}, \Upsilon_{t,K}^{co})$, and CPU frequency vector of sensors for round t, $\hat{f}_t = (f_{t,1}, \dots, f_{t,K})$, all uploaded by the sensors. The state vector $\mathbf{S}_t$ for the DRL framework at round t is defined as a vector with the following four components:

$$\mathbf{S}_t = [\hat{\Upsilon}_t^{co}, \hat{\Upsilon}_t^{cm}, \hat{f}_t, t] \tag{17}$$

To expedite the training of DRL, we normalize each element in the state vector to ensure they are on the same scale.

Action: At global round t, the DRL agent generates a local iteration decision for all sensors as the action based on the state collected and uploaded to the server, This action is defined as:

$$\mathbf{A}_t = [E_{t,1}, \cdots, E_{t,k}, \cdots, E_{t,K}], \quad E_{t,k} \in [1, E_{max}] \tag{18}$$

where the action space is discrete, and E_{max} denotes the empirically predefined global upper limit for local iterations of all sensors.

Reward: To optimize the FL performance outlined in **P1**, the reward function should capture the changes in learning performance, total latency, and local iteration disparity. Learning performance is measured after sensors execute the specified actions to update the model and subsequently evaluate it on the test datasets. Total latency and local iteration disparity are computed directly by the server based on the selected actions. Then the reward $\mathbf{R}_t$ at global training round t is defined as:

$$\mathbf{R}_t = \alpha_1 Acc(t) - \alpha_2 \Upsilon_t - \alpha_3 H_t \tag{19}$$

The reward can be derived from the **P1** problem by decomposing it into subproblems for each global round, allowing the reward for each specific time slot to be computed directly.

Algorithm 2 The DRL Agent Training Process

Input: The number of client K, The number of agent training rounds T_{ag}, The number of updates M.
Output:The adaptive local iteration decisions $\{E_{t,k}\}_{k\in\mathcal{K}}^{t\in\mathcal{T}}$.
Randomly initialize actor network $\pi(\cdot)$ and critic network $V(\cdot)$ with weight θ_a and θ_v.
Initialize experience replay buffer $\mathcal{D}$.
for $t = 0, 1, 2, ..., T_{ag} - 1$ **do**
    Each sensor records the information $(\Upsilon_{t,k}^{co}, \Upsilon_{t,k}^{co}, f_{t,k})$.
    Each sensor uploads its information to server.
    The server integrate the state $\mathbf{S}_t$.
    Get action $\mathbf{A}_t$ by feeding $\mathbf{S}_t$ into the actor network.
    Each sensor performs a number of local updates based on the action $\mathbf{A}_t$.
    Infer on test dataset to obtain test accuracy.
    The server calculate the reward $\mathbf{R}_t$.
    Update the state of FL from $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$.
    Store transition sample $(\mathbf{S}_t, \mathbf{A}_t, \mathbf{R}_t, \mathbf{S}_{t+1})$ into $\mathcal{D}$.
    for $m = 0, 1, 2, ..., M$ **do**
        The server update the actor network θ_a using PPO.
        The server update the critic network θ_v by maximizing the reward via Eq.20.
    end for
end for

Training Methodology: PPO provides a balanced approach by combining ease of implementation, sample efficiency, and straightforward tuning. It is designed to compute updates that minimize the objective function while constraining deviations from the previous policy. Additionally, PPO is well-suited for scenarios involving discrete action spaces. Consequently, in this work, the DRL agent's actor network update process employs the PPO algorithm.

The detailed training process of the DRL agent is outlined in Algorithm 2. At the start of the DRL agent's training

process, the parameters of both the actor and critic networks are randomly initialized, and the experience replay buffer is set up. During each agent training round $t = 0, 1, 2, ..., T_{ag} - 1$, sensor k uploads the observation to the server, including the data collection latency, the communication latency and CPU frequency. The server consolidates these observations into a unified state and determines the local iteration decisions as action for all sensors by inputting the state into the actor network $\pi(\cdot)$. Each sensor subsequently executes the specified number of local iterations based on the action it receives. Once the local updates are completed, the server evaluates the test accuracy and calculates the reward, factoring in test accuracy, overall latency, and local iterations disparity. The training then transitions to the next state, $\mathbf{S}_{t+1}$, while the experience from round t is stored in the experience replay buffer. Then server subsequently updates the DRL agent using the experiences stored in the replay buffer, performing M times. During this process, the actor network $\pi(\cdot)$ is updated by the PPO algorithm and the critic network $V(\cdot)$ is updated with the following gain function:

$$\max_{\theta_v} \frac{1}{|\mathcal{D}|} \sum_{t=1}^{|\mathcal{D}|} \left[\mathbf{R}_t + \gamma V(\mathbf{S}_{t+1}; \theta_v) - V(\mathbf{S}_t; \theta_v)\right]^2 \tag{20}$$

After the DRL agent has been trained for T_{ag} rounds, the server retains the actor network. During the subsequent DAO-VFL learning process, as outlined in Algorithm 1, the state is input into the actor network to generate the output action. This action is then used to guide the local iteration decisions of the sensors in each global round, enabling adaptive local iteration in DAO-VFL.

Computational Cost: The computational cost of Algorithm 2 consists of three main phases: the Sampling Phase ($\mathcal{O}_{sp}$), the Function Optimization Phase ($\mathcal{O}_{op}$), and the Advantage Estimation Phase ($\mathcal{O}_{ae}$). These correspond to the three primary sub-modules updated in the PPO algorithm. Specifically, the sampling phase has complexity $\mathcal{O}_{sp} = T_{ag} \cdot d_{ac} \cdot d_{ob}$, where T_{ag} is the number of agent training rounds, d_{ac} is the action space dimension, and d_{ob} is the observation space dimension. The function optimization phase has complexity $\mathcal{O}_{op} = T_{ag} \cdot \mathcal{D}$, where $\mathcal{D}$ denotes the size of the replay buffer. The advantage estimation phase has complexity $\mathcal{O}_{ae} = T_{ag}$. In large-scale IIoT deployments, the term $d_{ac} \cdot d_{ob}$ becomes relatively negligible. Therefore, the overall computational cost of Algorithm 2 can be approximated as $\mathcal{O}(T_{ag} \cdot \mathcal{D})$.

VII. Experiments

In this section, we present the experimental evaluation of the DAO-VFL algorithm. The experiments were conducted on an Ubuntu 18.04 machine equipped with an Intel Core i7-10700KF 3.8GHz CPU and a GeForce RTX 3070 GPU. The model training module was built upon PyTorch. The detailed experimental settings are outlined below.

A. Datasets

To simulate DAO-VFL in IIoT assembly line scenarios, we utilize the CIFAR-10 dataset, a widely used benchmark, alongside the real-world IIoT sensor-based dataset, C-MAPSS. Detailed descriptions of both datasets are provided below.

CIFAR-10: The CIFAR-10 dataset is widely used for image classification tasks, containing a total of 60,000 32x32 color images in 10 distinct classes. In the training setup, 4 sensors are involved, with each sensor handling a specific quadrant of every image. This setup resembles a multi-camera system that collectively acquires full visual information. At each global round, the trained models are evaluated on the test dataset to measure the current test accuracy.

C-MAPSS: The C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset [62], created by NASA, is extensively utilized in research on Remaining Useful Life (RUL) prediction, particularly in the field of aerospace engineering for prognostics. This dataset models the degradation processes in aircraft turbofan engines under a range of operational and fault conditions. It contains four subsets, each varying in the number of operating and fault conditions, with each subset divided into training and test sets. For our experiments, we use the FD002 subset, consisting of 50,119 data samples for training and 30,365 data samples for testing. Each row in the dataset provides a snapshot from a single operating cycle and contains 27 columns: the first column indicates engine ID, the second the current operational cycle number, columns 3-5 represent three operational settings influencing engine performance, columns 6-26 capture readings from 21 sensors, and the 27th column shows the actual RUL. In this setup, we assume the data is collected among 2 sensors. After removing irrelevant features, the first sensor is assigned the first 10 columns as feature, while the second sensor receives the remaining 10. This setup reflects a realistic industrial scenario where different sensors collect distinct aspects of the same product's condition.

Each time series begins with the engine operating under normal conditions, with a fault developing at an unknown point in time. In the training set, this fault progresses in severity until it results in system failure. In the test set, data is available up to a point shortly before failure. The objective is to predict the number of operational cycles remaining before failure in the test data. The performance of the RUL estimation model is evaluated using Root Mean Square Error (RMSE), a widely used metric for assessing RUL estimation.

B. Online Data Generation

In the experiment, operating within an online learning scenario requires the training dataset to be dynamic, with data collected at the start of each global round. To ensure sufficient data samples for good training performance, we collect the initial dataset at the beginning of the training process. Considering the differences in dataset types and sizes, distinct online data generation methods are employed for the CIFAR-10 and C-MAPSS datasets. Details for each case are provided below.

CIFAR-10: For the CIFAR-10 dataset, as the data samples are independent and lack time-series correlation, we adopt a fixed-size training dataset approach for each global round. In this approach, for every new data sample added, an equal

number of older data samples are removed. The initial training dataset contains 5,000 samples. Starting from the first global round, each sensor collects 200 new data samples corresponding to its partial features and removes 200 old data samples in each global round. This ensures that training dataset is consistently updated every global round with the same amount of data samples. Similarly, the test dataset is updated in the same manner every global round.

C-MAPSS: For the C-MAPSS dataset, where data samples are interconnected due to their representation of the RUL of a aircraft turbofan engines over time, we adopt an incremental training dataset approach. In this case, the training dataset gradually accumulates new data samples until it encompasses the entire original training dataset. The initial training dataset contains 1,000 samples. After the first global round, each sensor collects 100 new data samples for the corresponding partial features in each subsequent global round. For the test dataset, the full set of samples from the original test dataset is used in this case.

C. Model Details

Here, we will individually present all the models used in our experiments, including the feature model, head model, denoising autoencoder model, and the Actor and Critic network models for DRL, separately for the two datasets.

CIFAR-10: The feature model consists of 13 convolutional layers followed by pooling layers. It concludes with fully connected layers, and outputs a feature embedding of size 4096. The head model is a single fully connected layer that takes the concatenated feature embeddings from all sensors (4096×4) and maps them to the final 10 classes. The feature and head model employs the OGD optimizer with a learning rate of 0.01. The denoising autoencoder model includes an encoder with three fully connected layers, taking an input of size 4096 and producing an output of size 512. It also features a decoder with three fully connected layers, taking an input of size 512 and generating an output of size 4096. The DAE model employs the Adam optimizer with a learning rate of 0.01. The Actor network consists of three fully connected layers, each with 64 neurons. The input size of the Actor network corresponds to the state size, while the output size matches the action size. Similarly, the Critic network comprises three fully connected layers with 64 neurons each. It takes the state as input and outputs a single state-value, estimating the expected return from a given state under the current policy. The Actor model uses the Adam optimizer with a learning rate of 0.0001, while the Critic model uses the Adam optimizer with a learning rate of 0.001.

C-MAPSS: The feature model consists of two convolutional layers with 8 and 14 channels, respectively, followed by a flattening layer that outputs a feature embedding of size 28 for each sensor. The head model is a single fully connected layer that processes the concatenated feature embeddings from all sensors (28×2) and maps them to the final 131 classes for the classification task. The feature and head model employs the OGD optimizer with a learning rate of 0.01 The denoising autoencoder model includes an encoder with three fully connected layers, taking an input of size 28 and producing an output of size 3. It also features a decoder with three fully connected layers, which takes an input of size 3 and reconstructs an output of size 28. The DAE model employs the Adam optimizer with a learning rate of 0.01. The Actor and Critic networks share a similar structure to those described earlier, with the only difference being the size of the action space, which varies depending on the number of sensors involved. The Actor model uses the Adam optimizer with a learning rate of 0.0001, while the Critic model uses the Adam optimizer with a learning rate of 0.001.

For the reader's convenience, we summarize several key experimental parameters in the following Table III, based on the discussion above.

TABLE III: Key Parameters in Experiments

Parameter	CIFAR-10	C-MAPSS
Size of initial dataset	5000	1000
Size of new arrival data	200	100
Size of feature embedding	4096	28
Classification class	10	131
Optimizer (DAO)	OGD (0.01)	OGD (0.01)
Optimizer (DAE)	Adam (0.01)	Adam (0.01)
Optimizer (Actor)	Adam (0.0001)	Adam (0.0001)
Optimizer (Critic)	Adam (0.001)	Adam (0.001)

D. Environment Details

Here, we address the environment setup considerations for the adaptive local iteration decisions part. The setup focuses on designing collection latency, communication latency, and computation latency. To ensure a balanced contribution from each component and prevent any single factor from dominating due to significantly higher values, we adjust their values to be of the same order of magnitude. For collection latency, we set $\mu_0 = 2$, and μ is assigned a random value between $[2, 4]$. For communication latency, we set the number of weights $W_{fe} = 1 \times 10^5$, total bandwidth $B = 1 \times 10^7$HZ, transmission power $p_k = 1$W, noise power $\sigma^2 = 5 \times 10^{-2}$W, and the channel gain is a random value between $[1 \times 10^{-4}, 1 \times 10^{-5}]$, varying across sensors and rounds. For computation latency, we set the number of CPU cycles for a single model weight $C = 1000$ and the number of weights in the local model to $W_{lc} = 5 \times 10^5$, the CPU frequency of sensor is selected from two categories: high performance sensors are chosen randomly from $[2\times10^7, 4\times10^7]$, and ow-performance sensors are chosen from $[1 \times 10^7, 3 \times 10^7]$ in each global round.

To summarize, the setup reflects the sensor heterogeneity and communication environment variations, specifically in terms of channel gain and CPU frequency, which align with the challenges present in IIoT-based assembly lines.

E. Benchmarks

In the experiment, the following benchmarks are employed for performance comparison. Since the two key features of the DAO-VFL algorithm: noise reduction and adaptive local iteration decisions are essentially independent and do not

influence each other, a control variable approach is employed to design independent benchmarks for comparison. We begin by presenting the benchmarks related to noise reduction part.

Noise Excluded (NE). In this approach, the communication process assumes no noise impact, enabling the server to receive noise-free feature embeddings from the sensors.

Noise Included (NI). In this approach, communication noise is not addressed, resulting in the server receiving noisy feature embeddings from the sensors, which are then used in subsequent processes. Here, we consider using a uniform scalar quantizer to model the noise added to the feature embeddings, where the noise level is influenced by the quantization level. A smaller quantization level corresponds to higher noise.

Next, we will present the benchmarks related to adaptive local iteration decisions part.

Homogeneity (HO). In this approach, all sensors are trained using the same maximum number of fixed local iterations $E_{t,k} = E_{max}$. This scenario completely disregards the impact of total latency.

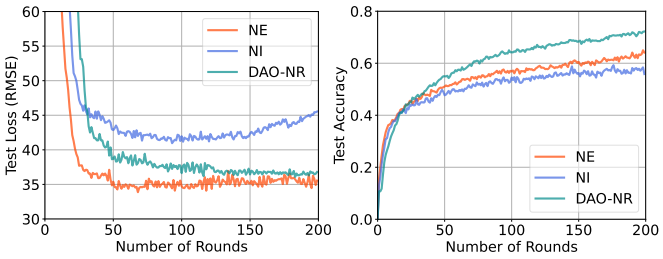
Heterogeneity (HE). In this approach, sensors are trained with significantly different predetermined numbers of fixed local iterations. Specifically, one sensor is trained with E_{max}, while all other sensors perform the minimum local iterations, set as $E_{t,k} = 1$.

F. Simulation Results

Next, we present the experimental results of DAO-VFL, starting with an analysis of the impact of noise reduction on learning performance. To ensure a controlled comparison, we initially disregard the effect of local iteration decisions and assume that both our proposed method and the benchmarks follow identical local iteration decisions. All simulation results are averaged over 10 random runs.

Performance Comparison (Noise Reduction). We first evaluate the learning performance between proposed algorithm and benchmarks with $E_{t,k} = 2$ and $T_{dl} = 40$. Given the aforementioned simulation setup, the performance comparison between DAO-VFL and benchmarks is shown in Fig. 6(a) and Fig. 6(b) based on C-MAPSS dataset and the CIFAR-10 dataset, respectively. From both figures, several key observations can be drawn. First, we observe that noise significantly impacts learning performance across all datasets. This effect is particularly pronounced in the case of the C-MAPSS dataset, where failing to address noise can even hinder convergence. Second, we observe that the learning performance of the DAO-NR approach varies across different datasets. For the C-MAPSS dataset, the DAO-NR approach effectively mitigates the impact of noise, achieving learning performance comparable to that of the NE. For the CIFAR-10 dataset, the DAO-NR approach not only eliminates the effect of noise but also further enhances learning performance. This improvement may be attributed to the regularization effect of the DAE and its robust feature learning capabilities. In summary, the DAO-NR approach for noise reduction is both effective and essential for mitigating noise during online VFL training process.

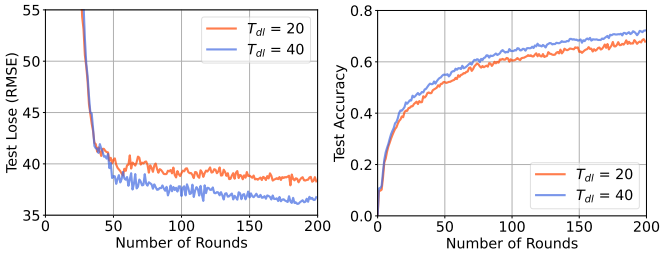
Impact of Denoising Learning Period T_{dl}. In this section, we evaluate the impact of denoising learning period



(a) Test Loss with C-MAPSS (b) Test Accuracy with CIFAR-10

Fig. 6: Performance Comparison of DAO-VFL and Benchmarks Considering Noise Reduction Effects.

$T_{dl} \in [20, 40]$ on the learning performance. Fig. 7(a) and Fig. 7(b) depict the relationship between test loss/accuracy and the denoising learning period T_{dl} for the C-MAPSS and CIFAR-10 datasets, respectively. The results indicate that extending the denoising learning period T_{dl} improves learning performance in both cases. Notably, the improvement is more pronounced for the C-MAPSS dataset than for the CIFAR-10 dataset. This difference may be attributed to the incremental dataset approach used for the C-MAPSS dataset, where the increasing data volume significantly enhances the training of the DAE. In contrast, the CIFAR-10 dataset employs a fixed-size dataset approach, which limits the amount of data available for training the DAE.



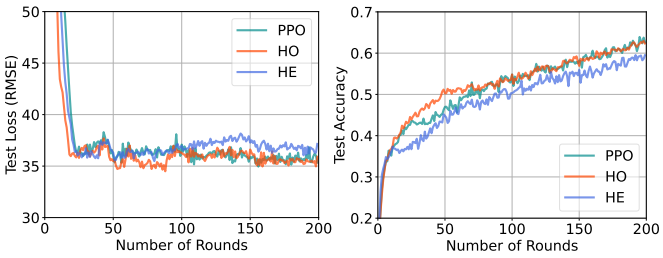
(a) Test Loss with C-MAPSS (b) Test Accuracy with CIFAR-10

Fig. 7: Performance Comparison of DAO-VFL and Benchmarks with Different Denoising Learning Period T_{dl}.

Next, we will examine the impact of local iteration decisions on learning performance. To ensure consistency, we control variables so that the effect of noise reduction remains the same for both DAO-VFL and the benchmarks. We begin by comparing the learning performance between the adaptive local iteration decisions, derived using PPO, with the previously introduced benchmarks.

Performance Comparison (Adaptive Iterations). In this section, we evaluate the learning performance of DAO-PPO compared with the benchmarks HO and HE. Using the simulation setup described earlier, we present the performance comparison between DAO-PPO and the benchmarks, considering the local iteration decisions, is illustrated in Fig. 8(a) and Fig. 8(b) based on C-MAPSS dataset and the CIFAR-10 dataset, respectively. We observe that the learning performance of DAO-PPO is better than that of HE throughout the learning process but not as good as HO. This is because DAO-PPO does not completely eliminate local iteration disparity as HO does, leading to slightly inferior performance compared to HO. In the following experimental results on total latency and reward,

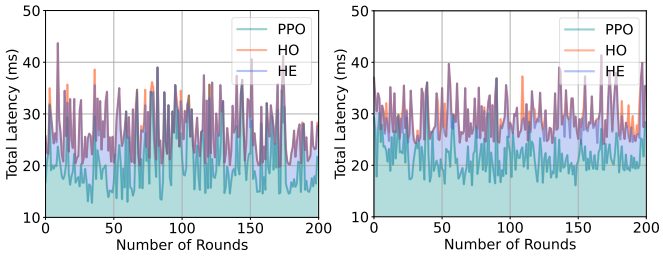
we will demonstrate the advantages of using DAO-PPO.



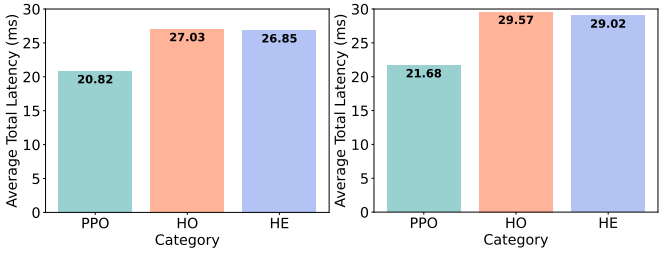
(a) Test Loss with C-MAPSS (b) Test Accuracy with CIFAR-10

Fig. 8: Performance Comparison of DAO-VFL and Benchmarks Considering Adaptive Local Iteration Decision.

Comparison of Total Latency. In this section, we compare the total latency of DAO-PPO with benchmarks within a single run, focusing on two key metrics: per-round total latency and average total latency. Fig.9(a) and Fig.9(b) illustrate the per-round total latency for the C-MAPSS and CIFAR-10 datasets, respectively, while Fig.9(c) and Fig.9(d) present the average total latency for the C-MAPSS and CIFAR-10 dataset. The experimental results reveal two findings: first, DAO-PPO outperforms both HO and HE in terms of per-round and average total latency, demonstrating its effectiveness in reducing overall latency; second, HE slightly surpasses HO in per-round and average total latency, which can be attributed to occasional rounds where the sensor with the minimum local iteration decision avoids becoming a bottleneck, thereby reducing the overall latency.



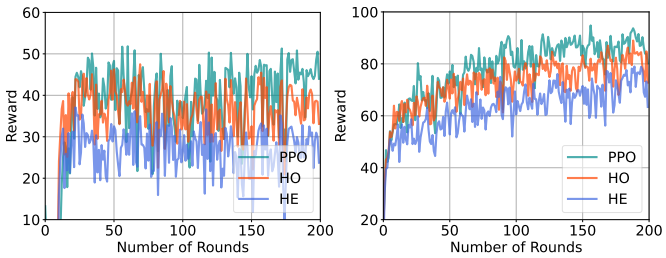
(a) Per-Round Total Latency with C-MAPSS (b) Per-Round Total Latency with CIFAR-10



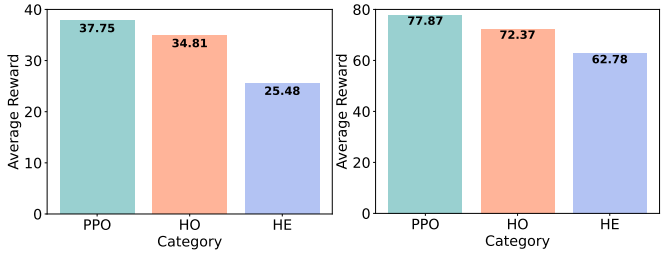
(c) Average Total Latency with C-MAPSS (d) Average Total Latency with CIFAR-10

Fig. 9: Comparison of Total Latency.

Comparison of Reward. In this section, we compare the reward of DAO-PPO with benchmarks within a single run. The evaluation focuses on two key metrics: per-round reward and average reward. Fig. 10(a) and Fig. 10(b) present the per-round reward for the C-MAPSS and CIFAR-10 dataset, while Fig. 10(c) and Fig. 10(d) show the average reward for the C-MAPSS and CIFAR-10 dataset. The results reveal two key findings: first, DAO-PPO achieves higher per-round and average rewards compared to HO and HE, as its learning objective is to maximize overall reward by effectively balancing learning performance, total latency, and local iteration disparity. Second, unlike the total latency results where HE and HO are relatively close, HO significantly outperforms HE in terms of reward. This is because HO is unaffected by local iteration disparity, whereas HE is impacted by it.



(a) Per-Round Reward with C-MAPSS (b) Per-Round Reward with CIFAR-10



(c) Average Reward with C-MAPSS(d) Average Reward with CIFAR-10

Fig. 10: Comparison of Reward.

The experimental results above demonstrate the superiority of the proposed DAO-VFL algorithm over its corresponding benchmarks in both noise reduction and adaptive local iteration decision-making.

Discussion on Sensor or Network Failure: In some cases, sensors may experience failures during training, and network connectivity issues can also arise. To mitigate the impact of sensor failures, deploying redundant sensors or sensor clusters can enhance system robustness and maintain training continuity. For network-related failures, implementing more reliable wired connections in key point can enhance communication stability and reduce the risk of data transmission disruptions.

VIII. Conclusion

In this work, we proposed the DAO-VFL algorithm to address key challenges in real-world IIoT scenarios, specifically focusing on multi-sensor co-training framework for industrial assembly lines. In addition to the inherent challenge of integrating online learning with VFL, we tackled critical issues prevalent in industrial environments, such as communication noise and sensor heterogeneity. To mitigate the impact of communication noise, we employed a denoising autoencoder to reduce noise in feature embeddings during the training process. To address sensor heterogeneity, we leveraged deep reinforcement learning to determine local iteration decisions for each sensor as actions. This approach reduces total latency and local iteration disparity while maintaining robust learning

performance. Our findings are supported by detailed theoretical analysis and extensive experimental validation. Future research will focus on the development of a testbed and optimizing performance in real-world industrial assembly line environments using actual industrial data, aiming to advance intelligent solutions for IIoT applications.

REFERENCES

[1] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[2] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[3] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.

[4] F. Al-Turjman and S. Alturjman, "Context-sensitive access in industrial internet of things (iiot) healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2736–2744, 2018.

[5] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[6] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[7] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, and T. Ranbaduge, "Vertical federated learning: challenges, methodologies and experiments," *arXiv preprint arXiv:2202.04309*, 2022.

[8] Y. Li, S. Wang, C.-Y. Chi, and T. Q. Quek, "Differentially private federated learning in edge networks: The perspective of noise reduction," *IEEE Network*, vol. 36, no. 5, pp. 167–172, 2022.

[9] H. Wu, X. Lyu, and H. Tian, "Online optimization of wireless powered mobile-edge computing for heterogeneous industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9880–9892, 2019.

[10] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[11] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning for industrial internet of things in future industries," *IEEE Wireless Communications*, vol. 28, no. 6, pp. 192–199, 2021.

[12] P. Boobalan, S. P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P. K. R. Maddikunta, T. R. Gadekallu, and T. Huynh-The, "Fusion of federated learning and industrial internet of things: A survey," *Computer Networks*, vol. 212, p. 109048, 2022.

[13] I. Kevin, K. Wang, X. Zhou, W. Liang, Z. Yan, and J. She, "Federated transfer learning based cross-domain prediction for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4088–4096, 2021.

[14] C. Xu, Y. Qu, T. H. Luan, P. W. Eklund, Y. Xiang, and L. Gao, "An efficient and reliable asynchronous federated learning scheme for smart public transportation," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 5, pp. 6584–6598, 2022.

[15] Z. Su, Y. Wang, T. H. Luan, N. Zhang, F. Li, T. Chen, and H. Cao, "Secure and efficient federated learning for smart grid with edge-cloud collaboration," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1333–1344, 2021.

[16] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. J. Piran, and M. S. Hossain, "Toward accurate anomaly detection in industrial internet of things using hierarchical federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7110–7119, 2021.

[17] S. U. Khan, C. E. Garca, T. Hwang, and I. Koo, "Radio environment map construction based on privacy-centric federated learning," *IEEE Access*, vol. 12, pp. 28 109–28 121, 2024.

[18] J. Mao, Z. Wei, B. Li, R. Zhang, and L. Song, "Towards ever-evolution network threats: a hierarchical federated class-incremental learning approach for network intrusion detection in iiot," *IEEE Internet of Things Journal*, 2024.

[19] J. Zhang, C. Luo, M. Carpenter, and G. Min, "Federated learning for distributed iiot intrusion detection using transfer approaches," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 7, pp. 8159–8169, 2022.

[20] Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa, "Noise removal in the presence of significant anomalies for industrial iot sensor data in manufacturing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7084–7096, 2020.

[21] H. Wang, J. Bian, and J. Xu, "On the local cache update rules in streaming federated learning," *IEEE Internet of Things Journal*, 2023.

[22] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017.

[23] H. Zhu, H. Zhang, and Y. Jin, "From federated learning to federated neural architecture search: a survey," *Complex & Intelligent Systems*, vol. 7, no. 2, pp. 639–657, 2021.

[24] S. Feng, "Vertical federated learning-based feature selection with non-overlapping sample utilization," *Expert Systems with Applications*, vol. 208, p. 118097, 2022.

[25] Y. Kang, Y. Liu, and X. Liang, "Fedcvt: semi-supervised vertical federated learning with cross-view training," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–16, 2022.

[26] J. Li, R. Deng, T. Zang, M. Kong, and K. Zhu, "Efficient and secure contribution estimation in vertical federated learning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 1205–1214.

[27] C. Fu, X. Zhang, S. Ji, J. Chen, J. Wu, S. Guo, J. Zhou, A. X. Liu, and T. Wang, "Label inference attacks against vertical federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1397–1414.

[28] P. Ye, Z. Jiang, W. Wang, B. Li, and B. Li, "Feature reconstruction attacks and countermeasures of dnn training in vertical federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[29] T. Castiglia, S. Wang, and S. Patterson, "Flexible vertical federated learning with heterogeneous parties," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[30] J. Zhang, S. Guo, Z. Qu, D. Zeng, H. Wang, Q. Liu, and A. Y. Zomaya, "Adaptive vertical federated learning on unbalanced features," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4006–4018, 2022.

[31] T. J. Castiglia, A. Das, S. Wang, and S. Patterson, "Compressed-vfl: Communication-efficient learning with vertically partitioned data," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2738–2766.

[32] H. Wang, J. Bian, and L. Wang, "Computation and communication efficient lightweighting vertical federated learning," *arXiv preprint arXiv:2404.00466*, 2024.

[33] J. Cui, C. Chen, L. Lyu, C. Yang, and W. Li, "Exploiting data sparsity in secure cross-platform social recommendation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 524–10 534, 2021.

[34] W. Ou, J. Zeng, Z. Guo, W. Yan, D. Liu, and S. Fuentes, "A homomorphic-encryption-based vertical federated learning scheme for rick management," *Computer Science and Information Systems*, vol. 17, no. 3, pp. 819–834, 2020.

[35] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[36] J. Jiang, L. Burkhalter, F. Fu, B. Ding, B. Du, A. Hithnawi, B. Li, and C. Zhang, "Vf-ps: How to select important participants in vertical federated learning, efficiently and securely?" *Advances in Neural Information Processing Systems*, vol. 35, pp. 2088–2101, 2022.

[37] Y. Xi, Y. Guo, S. Xu, C. Cai, and X. Jia, "Private sample alignment for vertical federated learning: An efficient and reliable realization," *IEEE Transactions on Information Forensics and Security*, 2025.

[38] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: learning deep neural networks on the fly," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.

[39] S. Hong and J. Chae, "Communication-efficient randomized algorithm for multi-kernel online federated learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9872–9886, 2021.

[40] D. Kwon, J. Park, and S. Hong, "Tighter regret analysis and optimization of online federated learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[41] A. Mitra, H. Hassani, and G. J. Pappas, "Online federated learning," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 4083–4090.

[42] H. Wang and J. Xu, "Online vertical federated learning for cooperative spectrum sensing," *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[43] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction wiener filter," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1218–1234, 2006.

[44] X. Ji, Z. Zhu, W. Xi, O. Gadyatskaya, Z. Song, Y. Cai, and Y. Liu, "Fedfixer: mitigating heterogeneous label noise in federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 830–12 838.

[45] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *International conference on learning representations*, 2017.

[46] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1944–1952.

[47] X. Wei and C. Shen, "Federated learning over noisy channels: Convergence analysis and design examples," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 1253–1268, 2022.

[48] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3452–3464, 2020.

[49] V. P. Chellapandi, A. Upadhyay, A. Hashemi, and S. H. Żak, "Fednmut–federated noisy model update tracking convergence analysis," *arXiv preprint arXiv:2403.13247*, 2024.

[50] J. Li, Z. Chen, K. F. E. Chong, B. Das, T. Q. Quek, and H. H. Yang, "Robust federated learning over the air: Combating heavy-tailed noise with median anchored clipping," *arXiv preprint arXiv:2409.15100*, 2024.

[51] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[52] M. Zhao, G. Cao, X. Huang, and L. Yang, "Hybrid transformer-cnn for real image denoising," *IEEE Signal Processing Letters*, vol. 29, pp. 1252–1256, 2022.

[53] M. Cheffena, "Industrial wireless communications over the millimeter wave spectrum: opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 66–72, 2016.

[54] Y. Ying and M. Pontil, "Online gradient descent learning algorithms," *Foundations of Computational Mathematics*, vol. 8, pp. 561–596, 2008.

[55] R. Zurawski, *Industrial communication technology handbook*. CRC Press, 2014.

[56] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: learning image restoration without clean data," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2965–2974, 2018.

[57] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

[58] J. Park, D. Kwon, and S. Hong, "Fedqogd: Federated quantized online gradient descent with distributed time-series data," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 536–541.

[59] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[60] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[61] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[62] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 International Conference on Prognostics and Health Management*. IEEE, 2008, pp. 1–9.