This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

BR-MTFL: A Novel Byzantine Resilience Enhanced Multi-Task Federated Learning Framework for High-Speed Train Fault Diagnosis

Junxian You, Rui Yang, *Senior Member, IEEE*, Yifan Zhan,
Baoye Song, Yong Zhang, Zidong Wang, *Fellow, IEEE*

***Abstract*—In high-speed train systems deployed across diverse geographical regions, robust fault diagnosis techniques are essential for ensuring operational safety. This paper proposes the Byzantine resilience enhanced multi-task federated learning framework (BR-MTFL), a novel framework tailored for the complexities of fault diagnosis in traction asynchronous motors under varying operational conditions. This framework innovatively introduces multi-task federated learning to accommodate regional law restrictions and varying fault diagnosis requirements. In addition, the Byzantine resilience of our proposed framework is specially enhanced to address the challenges posed by inconsistent and potentially misleading feature distributions across different train networks. BR-MTFL is practically validated through experiments conducted across 9 clients, each representing a distinct set of fault types and operational conditions typical of high-speed trains. The experiments demonstrate the ability of BR-MTFL to outperform conventional federated learning frameworks in terms of accuracy and resilience to Byzantine threats. BR-MTFL establishes a new standard for federated learning applications in high-speed train fault diagnosis, particularly where data diversity and privacy dominate.**

***Index Terms*—High-speed train, fault diagnosis, federated learning, multi-task learning, Byzantine resilience.**

I. Introduction

HIGH-SPEED train systems are vital to modern transportation, providing fast and reliable service across diverse geographical regions [1] [2]. The traction asynchronous motor, a key element of the drive system, is especially critical to the performance of high-speed train systems. Faults such as broken rotor bars, stator inter-turn short-circuits, and air-gap eccentricity can severely impact the operation of motors [3]. Consequently, developing robust fault diagnostic techniques is essential for ensuring the safe operation of high-speed trains across diverse regions [4]. Deep learning-based methods have revolutionized the field of fault diagnosis [5], by providing powerful tools to analyze complex data streams [6]. However, the implementation of these methods faces challenges in data sharing and data heterogeneity among clients.

On one hand, legal and regulatory constraints, such as the General Data Protection Regulation [7], [8], limit data sharing among clients across regions. To cope with this challenge, federated learning has been introduced into fault diagnosis [9]. This decentralized approach enables multiple clients to collaboratively train a shared global model without transferring raw data to a central server, ensuring data privacy and compliance with legal regulations [10]–[12]. However, the strategy of creating a single global model for all clients is not suitable for high-speed train systems. Therefore, customized diagnostic strategies need to be designed for clients in different regions.

On the other hand, models trained on datasets from one region often fail to perform well in other regions, as high-speed train systems face varying faults in different regions [13], [14]. The fault type obtained by clients varies significantly due to regional differences, resulting in data heterogeneity and the need for client-specific diagnosis tasks. For example, in mountainous regions, the frequent need for trains to ascend and descend slopes imposes mechanical stress on traction asynchronous motors, causing broken rotor bars [15], reducing motor efficiency and increasing vibrations. In areas with high humidity, moisture can degrade the insulation of motor stators, leading to inter-turn short-circuits, causing overheating and potentially burning out the motor [7]. In the plains, where trains operate mainly at high speeds, continuous operation can accelerate mechanical wear, leading to air-gap eccentricity [15]. The specific challenges faced by each client in the corresponding region underscore the need for customized diagnostic strategies to effectively maintain the stability of high-speed train systems [16].

Fortunately, multi-task learning is well-suited for designing customized diagnostic strategies for high-speed train systems, as it allows clients to collaboratively learn region-specific models without sharing raw data, addressing both data heterogeneity and privacy concerns. It leverages shared feature representations across tasks, enabling the model to learn generalizable knowledge from different tasks [17]. Integrating multi-task learning into federated learning seems to be a promising

This work is partially supported by National Natural Science Foundation of China (62233012), Jiangsu Provincial Qinglan Project (2021), and Suzhou Science and Technology Programme (SYG202106).

J. You and Y. Zhan are with School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China, and also with School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Liverpool, L69 3BX, United Kingdom (e-mail: Junxian.You23@student.xjtlu.edu.cn, Yifan.Zhan22@student.xjtlu.edu.cn);

R. Yang is with School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China (e-mail: R.Yang@xjtlu.edu.cn);

B. Song is with the College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao 266590, China (email: Songbaoye@sdust.edu.cn);

Y. Zhang is with the Department of Information Science and Engineering, Wuhan University of Science and Technology, Wuhan 430081, China (email: zhangyong77@wust.edu.cn).

Z. Wang is with the Department of Computer Science, Brunel University of London, Uxbridge, Middlesex UB8 3PH, United Kingdom (email: Zidong.Wang@brunel.ac.uk).

Corresponding author: R. Yang.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

solution to fault diagnosis complicated by regional variations.

Recent studies on multi-task federated learning (MTFL) have demonstrated wide applications in various fields, such as Internet of Vehicles [18], [19] and smart healthcare [20]–[22]. However, MTFL faces the challenge of Byzantine threats [23]–[25]. Byzantine threats in federated learning refer to the introduction of faulty or misleading updates by certain clients during the model aggregation process, which can significantly disrupt the learning process [26]. These threats arise from inconsistencies across clients, where some updates deviate from the correct direction of model aggregation due to local data heterogeneity, software malfunctions, or even intentional malicious behavior. Such discrepancies can cause imbalanced contributions from clients, leading to a global model that is skewed towards irrelevant or non-generalizable features, ultimately degrading its performance. The potential harms of Byzantine threats are considerable, including the degradation of model accuracy, reduced generalization ability, and increased vulnerability to adversarial attacks. These issues can destabilize the learning process, causing the model to converge to incorrect solutions or become ineffective in real-world applications [27].

For further illustration of Byzantine threats in a high-speed train fault diagnosis, suppose the diagnostic data of most participating clients in the aggregation process mainly involves faults of broken rotor bars. In such cases, their current signals typically exhibit frequency anomalies, causing the models to focus on frequency features. Conversely, a few participating clients may concentrate on stator inter-turn short-circuits, characterized by current spike variations, leading their models to emphasize spike features. During aggregation, the models that focus on spike features might interfere with the aggregation of models emphasizing frequency features, ultimately affecting the model's ability to capture critical yet subtle frequency features accurately [7]. Although these clients are not malicious, the feature learning bias towards their local data characteristics due to data heterogeneity can lead to a decline in aggregated model performance [28], triggering potential Byzantine threats.

To handle the challenge of Byzantine threats, significant advancements have been made in enhancing Byzantine resilience across various fields, such as blockchain [29] and cloud computing [30]. Integrating Byzantine resilience enhancement into MTFL allows the aggregated model to remain reliable against internal inconsistencies among client tasks. Current strategies for boosting Byzantine resilience often rely on filtering client updates, typically by removing outliers [31] or selecting median values [32] to achieve a balanced parameterization of the model. While these methods effectively mitigate the impact of Byzantine threats, they inevitably lead to information loss. Specifically, even if all updates in a given iteration are valid and free from Byzantine threats, conventional methods would still discard certain updates due to their strict filtering criteria. This is particularly problematic in the context of high-speed train fault diagnosis, where the datasets are both scarce and valuable. Simply removing outliers may fail to distinguish useful data from noise and result in the loss of key fault patterns, affecting the model's reliability and diagnostic performance. Therefore, it is necessary to develop a filtering mechanism for high-speed train systems that maximizes the retention of valuable data while effectively mitigating the impact of unreliable updates.

To overcome the challenges posed by data sharing and data heterogeneity, and to enhance the system's Byzantine resilience, this paper proposes BR-MTFL, a novel Byzantine resilience enhanced MTFL framework applied in high-speed train fault diagnosis. Specifically, this paper proposes the formation of spontaneous clusters among clients as a method to address Byzantine threats. The clustering facilitates the gathering of updates from a reliable group of clients, focusing on preserving the most universally applicable and widely accepted model updates. Incorporation of all client contributions should be allowed when no Byzantine threats are present in the aggregation iteration. This allowance distinguishes our method from conventional approaches by balancing robustness against Byzantine threats and the preservation of contributions from clients with critical data.

The main contributions of this paper are summarized as follows:

1) By introducing Byzantine resilience into MTFL, this study reduces the influence of misleading local client updates and enhances reliability in high-speed trains;
2) This research explores the integration of MTFL in high-speed trains, aiming to overcome the challenges posed by data sharing and data heterogeneity across geographically distributed train systems, in order to meet the requirements for lawful and reliable fault diagnosis;
3) The client selection and aggregation process of federated learning is innovatively optimized, allowing substantial advances in the Byzantine resilience of the framework.

The rest of this paper is organized as follows: Section II provides a detailed architecture of the BR-MTFL framework; Section III presents a comprehensive analysis of experimental results, discussing the implications for the deployment of high-speed train fault diagnosis; Section IV concludes the paper with a discussion of the broader impact of this paper, highlighting the potential future research directions.

II. Proposed Byzantine Resilience Enhanced Multi-Task Federated Learning Framework

This section details the methodology adopted in the BR-MTFL framework, with related symbols presented in Table I. BR-MTFL is structured into three pivotal components, including adapted multi-task learning, Byzantine resilience enhancement, and weighted aggregation optimization.

A. Adapted Multi-Task Learning

Multi-task learning is adapted to optimize shared federated learning across multiple fault diagnosis tasks. Each client maintains a specific classifier while sharing a common feature extractor across the network. The integration of multi-task learning within federated learning is strategically designed to boost tailored solutions to diverse diagnosis tasks and accommodate the variability of operational conditions across different clients. This section outlines the architecture and

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

TABLE I
DEFINITIONS OF SYMBOLS.

Symbol	Description
$\theta_i'^{(t)}$	Locally trained model for client i at iteration t
$\theta_i^{(t)}$	Updated model for client i at iteration t
$\phi_i^{(t)}$	Feature extractor for client i at iteration t
$\phi^{(t)}$	Aggregated feature extractor at iteration t
$l_i^{(t)}$	Classifier for client i at iteration t
$\Phi^{(t)}$	Feature extractor vector for client i at iteration t
$\phi_{il}^{(t)}$	The l-th element of feature extractor vector
∇L_i	Gradient of the loss function for client i
$D^{(t)}$	Euclidean distance matrix between clients
$D_{i,j}^{(t)}$	Distance between client i and j feature extractors
$C_k^{(t)}$	Set of feature extractors in cluster k at iteration t
$S_k^{(t)}$	Number of clients in cluster k at iteration t
$B^{(t)}$	Set of clusters of Byzantine clients at iteration t
$R^{(t)}$	Set of non-Byzantine clients at iteration t
$\mu_1, \mu_2, \ldots, \mu_m$	Initial cluster centers
$\tilde{\mu}_k$	Updated cluster center for cluster k
$w_i^{(t)}$	Adjusted weight for client i at iteration $t+1$
η	Learning rate of local training
m, τ, κ, α	Hyperparameters
$\mathrm{Acc}_{\theta_i'^{(t)}}$	Local accuracy tested for client i at iteration t
$\mathrm{Acc}_{\theta_i^{(t)}}$	Updated accuracy tested for client i at iteration t

operational mechanics of the MTFL adopted in our framework, as illustrated in Fig. 1.

1) Architecture Description: The multi-task learning approach in our framework is based on the premise that, while common fault features may exist across clients, each client also faces unique diagnostic challenges stemming from differences in deployed regions, such as mountainous areas, high-humidity regions, and plains. To address this, our model architecture includes:

- **Shared Feature Extractor:** The shared feature extractor is composed of several convolutional layers, each coupled with batch normalization and activation functions. It is designed to capture universal features presented by various clients and ensure the generalization of feature representations.
- **Client-specific Classifiers:** Each client employs a classifier specifically configured to address its unique diagnostic needs. The configuration of classifiers, including layer numbers and activation functions, is optimized based on the specific type and number of faults each client regularly encounters.

2) Operational Mechanics: Each client updates its model parameters by training on local datasets, focusing on unique tasks. The diversity in fault types and numbers differentiates the updated model parameters, especially in feature extractors dealing with rare fault identification. The update process involves gradient descent optimization, performed iteratively as follows:

$$\theta_i'^{(t)} = \theta_i^{(t-1)} - \eta \nabla L_i(\theta_i^{(t-1)}) \tag{1}$$

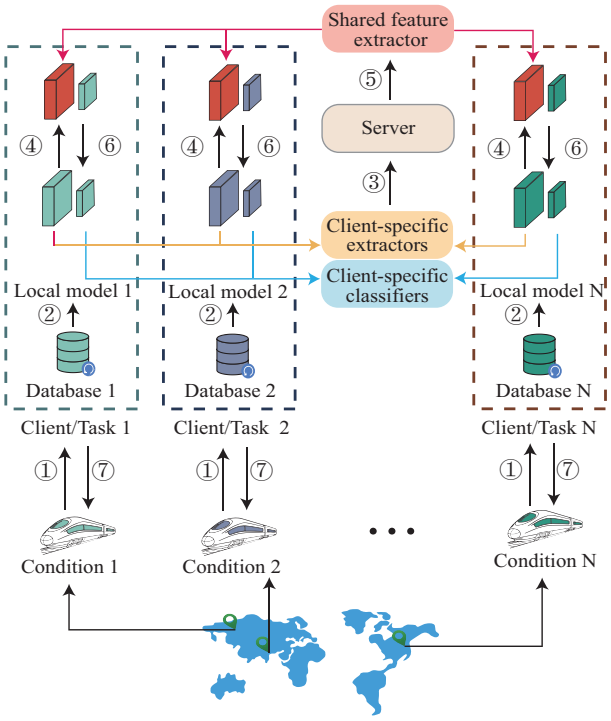

Fig. 1. Basic process of proposed MTFL framework applied in practical high-speed train fault diagnosis. In the setting of N clients, different working conditions influence data collection in ①. Each client receives its database and trains a local model consisting of the client-specific extractor and classifier in ②. In ③, client-specific extractors are uploaded to the cloud server and aggregated into a shared feature exactor. In ④, client-specific classifiers are reserved in each client for further process. In ⑤, the shared feature extractor is delivered to local clients and combined with corresponding client-specific classifiers as updated local models which are downloaded back to each client in ⑥. Finally, the client applies the client-specific local model to each working condition in ⑦.

where $\theta_i'^{(t)}$ represents the parameters of the locally trained model for client i at iteration t, η denotes the learning rate, and ∇L_i represents the gradient of the loss function with respect to the parameters of previous iterations $\theta_i^{(t-1)}$.

In the subsequent subsections, the feature extractors and the classifiers of the locally trained models will be discussed separately, as illustrated below:

$$\theta_i'^{(t)} = \left(\phi_i^{(t)}, l_i^{(t)}\right) \tag{2}$$

where $\theta_i'^{(t)}$ represents the sequential combination of both the feature extractor $\phi_i^{(t)}$ and the classifier $l_i^{(t)}$ for client i at iteration t.

Remark 1. *Both the feature extractor and the classifier are jointly trained locally using the above process. However, during the following Byzantine resilience enhancement and weighted aggregation optimization phases, the parameters of the classifier will remain fixed, and only the parameters of the feature extractor will be transferred to the server side for updating. The classifier avoids undergoing aggregations unsuitable for client-specific diagnosis tasks and maintains a stable classification performance while the feature extractor's generalization is enhanced.*

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

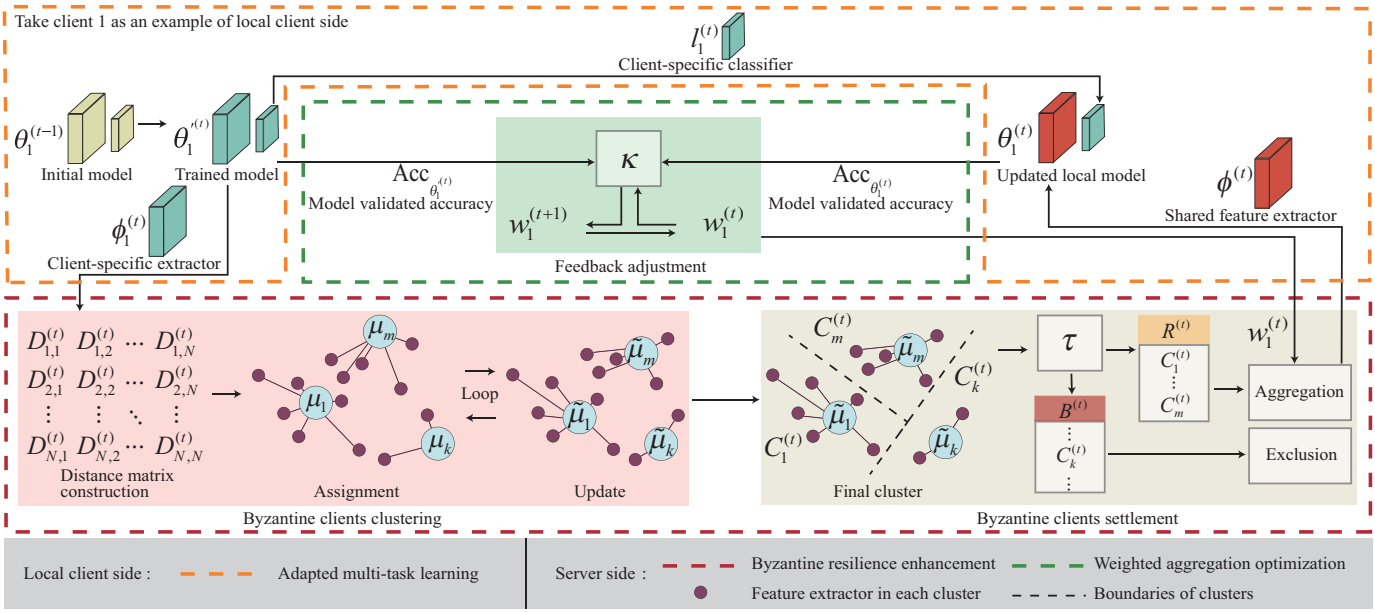

Fig. 2. Detailed process of proposed BR-MTFL framework.

B. Byzantine Resilience Enhancement

The detailed BR-MTFL framework, illustrated in Fig. 2, demonstrates an adaptive and secure approach to federated learning, capable of handling complex feature spaces and ensuring high levels of model reliability and accuracy. The challenge of enhancing Byzantine resilience in federated learning involves identifying conflicting or misleading clients that could potentially degrade the performance of the aggregated model. This framework adopts a reliable approach to address this challenge during the aggregation phase of the model. Specifically, clustering techniques based on feature extraction outputs are adopted to isolate outliers, ensuring that only reliable data can influence the model aggregation process. Upon receiving the feature extractors from the clients, the central server constructs a distance matrix and applies the K-means clustering algorithm to perform Byzantine client clustering and settlement.

1) Byzantine Client Clustering: The detection of potential Byzantine clients is achieved by analyzing the feature extraction layers submitted to the central server. The K-means clustering algorithm is employed for the distance matrix constructed from these feature extractor layers to further detect potential Byzantine clients. The algorithm helps in grouping clients into clusters based on the similarity of their updated layers. By inspecting these clusters, groups with significantly few clients can be identified as potential Byzantine clients, based on the assumption that their feature distributions or model updates are unrepresentative among the aggregation or diverge from the majority.

In the federated learning setting with N clients, the model parameters of feature extractor $\phi_i^{(t)}$ for each client i at iteration t is represented as a vector $\Phi_i^{(t)}$:

$$\Phi_i^{(t)} = [\phi_{i1}^{(t)}, \phi_{i2}^{(t)}, \ldots, \phi_{iL}^{(t)}], \tag{3}$$

where L is the length of the feature extractor vector, corresponding to the dimensionality of the feature extractor. $\phi_{il}^{(t)}$ is the l_{th} element of the feature extractor vector, representing the i_{th} client's feature value at the t_{th} iteration.

The distance matrix $D^{(t)}$ is constructed using the Euclidean distance between the vectors of the feature extractors from different clients, with matrix elements $D_{i,j}^{(t)}$ computed as:

$$D_{i,j}^{(t)} = \sqrt{\sum_{l=1}^{L} (\phi_{il}^{(t)} - \phi_{jl}^{(t)})^2} \tag{4}$$

where $\phi_{il}^{(t)}$ and $\phi_{jl}^{(t)}$ are the l_{th} elements of the feature extractor vectors of length L from clients i and j respectively. The whole distance matrix is defined as:

$$D^{(t)} = \begin{bmatrix} D_{1,1}^{(t)} & D_{1,2}^{(t)} & \cdots & D_{1,N}^{(t)} \\ D_{2,1}^{(t)} & D_{2,2}^{(t)} & \cdots & D_{2,N}^{(t)} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1}^{(t)} & D_{N,2}^{(t)} & \cdots & D_{N,N}^{(t)} \end{bmatrix} \tag{5}$$

The matrix $D^{(t)}$ is an $N \times N$ symmetric matrix with zeros along its diagonal, representing the distance between the feature extractors of each client and every other feature extractor. $D^{(t)}$ is used to select m initial cluster centers that are maximally distant from each other, denoted as $\mu_1, \mu_2, \ldots, \mu_m$. The selection of hyperparameter m can be guided by estimating the proportion of Byzantine clients in the network and evaluating clustering performance metrics across different values. The constructed matrix improves the convergence of the subsequent clustering process, which is iterated until the cluster centers do not change:

- **Assignment:** Each feature extractor vector $\phi_i^{(t)}$, $i \in \{1, 2, \ldots, N\}$, is assigned to the nearest cluster center

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

based on the Euclidean distance:

$$C_k^{(t)} = \{\phi_i^{(t)} \mid k = \arg\min_{j\in\{1,2,\ldots,m\}} \left\|\phi_i^{(t)} - \mu_j\right\|\} \quad (6)$$

where $C_k^{(t)}$ represents the set of feature extractor vectors assigned to cluster k from N clients.

- **Update:** Recompute each cluster center, denoted as $\tilde{\mu}_k$ for the updated cluster center for k_{th} cluster, with the mean of the feature extractor vectors assigned to it:

$$\tilde{\mu}_k = \frac{1}{S_k^{(t)}} \sum_{\phi_i^{(t)} \in C_k^{(t)}} \phi_i^{(t)} \quad (7)$$

where $S_k^{(t)}$ is the number size of clients in cluster k.

2) Byzantine Client Settlement: The potential Byzantine clients are identified and excluded in this phase. Clusters of unusually small sizes, suspected as the cluster of Byzantine clients, are included in $B^{(t)}$, while $R^{(t)}$ denotes the set of remaining non-Byzantine clients after excluding any identified as Byzantine clients or outliers. The process is formulated as:

$$B^{(t)} = \left\{C_k^{(t)} \mid k \in \{1, 2, \ldots, m\}, S_k^{(t)} < \tau\right\} \quad (8)$$

$$R^{(t)} = \bigcup_{\substack{k=1 \\ C_k^{(t)} \notin B^{(t)}}}^{m} C_k^{(t)} \quad (9)$$

where hyperparameter τ plays a critical role in balancing sensitivity and accuracy in identifying Byzantine clients.

The value of τ is chosen based on a combination of initial experimental analysis under a baseline scenario without Byzantine clients and insights from related literature [33], serving as a threshold below which the cluster size is considered insufficient to represent a group of common clients. Specifically, a preliminary experimental analysis of cluster sizes in a controlled baseline federated learning setting can be employed to determine an effective value for τ. By analyzing the distribution of cluster sizes in a baseline federated learning scenario without Byzantine clients, the minimum typical cluster size among common clients can be identified and adjusted. This empirical reference point is then slightly adjusted to set the threshold τ. An appropriately determined τ ensures that misleading clients are effectively identified and excluded without adversely affecting the contributions of other clients. Once τ is fixed, it directly influences the sets $B^{(t)}$ and $R^{(t)}$, thereby affecting the subsequent aggregation steps and the overall performance.

The clusters assigned to $B^{(t)}$ are excluded from the final model aggregation to prevent potential misleading updates from suboptimizing the model generalization. The aggregation of remaining non-Byzantine clusters of $R^{(t)}$ ensures that the federated learning process continues without discrete updates. The final aggregation process on the server side can be formulated as:

$$\phi^{(t)} = \frac{\sum_{\phi_i^{(t)} \in R^{(t)}} w_i^{(t)} \phi_i^{(t)}}{\sum_{\phi_i^{(t)} \in R^{(t)}} w_i^{(t)}} \quad (10)$$

where $w_i^{(t)}$ represents the adjusted weight of the client obtained in the $(t-1)_{th}$ iteration, and $\phi^{(t)}$ is the aggregated feature extractor. This procedure is vital for gradually enhancing collective knowledge over successive learning iterations.

C. Weighted Aggregation Optimization

The optimization serves as the final stage of the framework, including model validation and feedback adjustment of the federated learning process.

1) Model Validation: Upon receiving the aggregated feature extractor layer from the server, each client integrates this layer with their local classifier to form the updated model:

$$\theta_i^{(t)} = \left(\phi^{(t)},\ l_i^{(t)}\right) \quad (11)$$

where $\theta_i^{(t)}$ represents the updated model parameters of client i after feature extractor aggregation.

Clients then perform validation on their local datasets to assess the performance of the updated model. The accuracies of models with locally trained and shared feature extractor are represented by $\text{Acc}_{\theta_i'^{(t)}}$ and $\text{Acc}_{\theta_i^{(t)}}$ respectively. More precisely, $\text{Acc}_{\theta_i'^{(t)}}$ represents the accuracy of the model combining the local client-specific feature extractor $\phi_i^{(t)}$ with the client-specific classifier $l_i^{(t)}$. On the other hand, $\text{Acc}_{\theta_i^{(t)}}$ refers to the accuracy of the model combining the shared feature extractor $\phi^{(t)}$ with the local client-specific classifier $l_i^{(t)}$.

2) Feedback Adjustment: A weight adjustment mechanism is introduced to account for the unique contributions of individual clients. Specifically, if the locally trained model shows superiority in accuracy over the model sharing feature extractor, it indicates that the client's local data contains unique information not fully captured by the aggregated model. The client's weight is adjusted to emphasize and integrate these unique contributions. The client weight $w_i^{(t+1)}$ adjusted for the next iteration can be represented as:

$$w_i^{(t+1)} = w_i^{(t)} + \kappa(\text{Acc}_{\theta_i'^{(t)}} - \text{Acc}_{\theta_i^{(t)}}) \quad (12)$$

where the hyperparameter κ controls the rate of weight adjustment. The value of κ is set after analyzing the expected range of accuracy differences, ensuring weight adjustments within reasonable bounds. Proper choice of κ effectively balances the integration of unique client contributions without causing excessive weight fluctuations.

The initial aggregation weight for each client i in the first iteration is defined as $w_i^{(1)} = \frac{1}{N}$, where N is the total number of participating clients. Weights are normalized to maintain a balanced contribution throughout the network. This mechanism ensures that models with a rather unique feature extraction perspective have a greater influence and contribution when selected into aggregation in the following iteration. The pseudocode of the algorithm for BR-MTFL is described in Algorithm 1.

III. Experimental Results and Analysis of BR-MTFL

This paper illustrates the dataset from the HIL-HST simulation platform [34], [35]. The following subsections present

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

Algorithm 1: BR-MTFL Framework

Input: Set of clients N, cluster number m, accuracy threshold α
Output: Client-specific model θ_i

repeat
  for *each client* i **do**
    Train local model into $\theta_i'^{(t)}$ in (1)
    Process trained model as feature extractor $\phi_i^{(t)}$ and classifier $l_i^{(t)}$ in (2)
  end
  Process feature extractors into vectors $\Phi_i^{(t)}$ in (3)
  Compute distances between client feature extractor vectors $D_{i,j}^{(t)}$ in (4)
  Initial cluster centers with generated distance matrix $D^{(t)}$ in (5)
  repeat
    for *each cluster* k **do**
      Assign feature extractor vectors to nearest cluster $C_k^{(t)}$ in (6)
      Update new cluster centers $\tilde{\mu}_k$ in (7)
    end
  until *cluster centers do not change*;
  Detect potential Byzantine clients $B^{(t)}$ in (8)
  Gather remaining normal clients $R^{(t)}$ in (9)
  Aggregate into shared extractor $\phi^{(t)}$ in (10)
  for *each client* i **do**
    Combine classifier with shared feature extractor to form local model $\theta_i^{(t)}$ in (11)
    Test local and shared model accuracy
    Adjust client weight $w_i^{(t+1)}$ in (12)
  end
until *all clients achieve accuracy* $\geq \alpha$;
return Specific model for each client i with shared feature extractor θ_i

the dataset allocation and discuss the advantages of the proposed BR-MTFL framework through comparison and ablation studies.

A. Experimental Configuration

The HIL-HST simulation platform, illustrated in Fig. 3, was jointly established by Central South University and Zhuzhou Electric Locomotive Research Institute. The platform mainly consists of a digital real-time simulator (DRTS), a physical traction control unit (TCU), and a signal conditioner. The DRTS is responsible for the real-time simulation of the motor, power electronics system, and fault injection signals. The physical TCU generates PWM signals essential for controlling the motor and simulating realistic operating conditions. Meanwhile, the signal conditioner facilitates signal conversion, ensuring interaction between components. The platform investigates three primary types of faults: broken rotor bars, stator inter-turn short-circuits, and air gap eccentricity. These faults are generated through simulation on the DRTS and

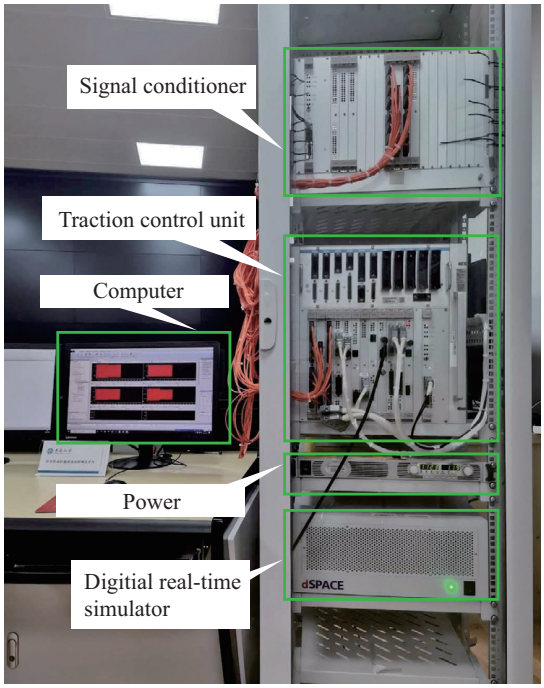

Fig. 3. HIL-HST traction control system simulation platform.

introduced into the motor model. The platform emulates how faults interact with other system components, replicating the electromagnetic and mechanical responses expected in real-world scenarios. The platform provides high-precision fault data under various fault severities and operating speeds. This ensures the ability to simulate diverse fault scenarios efficiently and safely, without relying on physical components.

The fault severity is represented on a scale from 0 (indicating no failure) to 1 (indicating complete failure). In this experiment, each fault can be categorized into three different levels: minor (0.005), medium (0.08), and serious (1.00). The medium level is set at 0.08 to reflect a fault intensity that, while significant, does not cause immediate system instability, aligning with real-world conditions where moderate faults may persist undetected over time. Fault data are collected at three speeds: 20, 169, and 280 km/h. The dataset includes sensor channels, with a sampling interval of $400\mu s$, as follows: 1) Stator three-phase current measurement data; 2) Rectifier AC side current measurement data; 3) DC voltage measurement data; 4) Rotation speed measurement data. Under normal operating conditions, the data collection period is approximately 40 seconds, starting after the system transitions to the corresponding operating state in the semi-physical simulation. This period captures data points reflecting steady-state performance. For faulty operating conditions, the data collection period includes 30 seconds before fault injection and 30 seconds after fault injection. These data points represent both the pre-fault and post-fault behaviors of the system.

Table II illustrates the type and severity of faults involved in this experiment, along with their specific labels, covering a total of 10 scenarios. The data used in the experiment are gathered from the A-phase current of high-speed trains operating at 280 km/h. Table III lays out the assignment of the

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

TABLE II
FAULT LABEL DESCRIPTION.

Fault label	Fault location	Fault level	Fault type	Working condition	Sensor data
N	Motor	Normal	Normal	280km/h	A-phase current
B1	Motor	Minor	Broken rotor bar	280km/h	A-phase current
B2	Motor	Medium	Broken rotor bar	280km/h	A-phase current
B3	Motor	Serious	Broken rotor bar	280km/h	A-phase current
S1	Motor	Minor	Stator inter-turn short-circuit	280km/h	A-phase current
S2	Motor	Medium	Stator inter-turn short-circuit	280km/h	A-phase current
S3	Motor	Serious	Stator inter-turn short-circuit	280km/h	A-phase current
A1	Motor	Minor	Air-gap eccentricity	280km/h	A-phase current
A2	Motor	Medium	Air-gap eccentricity	280km/h	A-phase current
A3	Motor	Serious	Air-gap eccentricity	280km/h	A-phase current

TABLE III
CLIENT-SPECIFIC TASK ASSIGNMENT.

Client	Classification task										Output number
	N	B1	B2	B3	S1	S2	S3	A1	A2	A3	
Client 1	✓	✓	✓	✓	✓	✓	✓	✓	✓		9
Client 2	✓	✓	✓	✓	✓	✓	✓	✓			8
Client 3	✓	✓	✓	✓	✓	✓		✓	✓	✓	9
Client 4	✓	✓	✓	✓	✓			✓	✓	✓	8
Client 5	✓	✓	✓		✓	✓	✓	✓	✓	✓	9
Client 6	✓	✓			✓	✓	✓	✓	✓	✓	8
Client 7	✓	✓	✓	✓	✓	✓		✓	✓		8
Client 8	✓	✓	✓		✓	✓		✓	✓	✓	8
Client 9	✓	✓	✓		✓	✓	✓	✓	✓		8

classification tasks for 9 clients in the experiment. Considering the operation of high-speed trains across varied geographical regions, certain clients exhibit relatively low level of fault extent to simulate different real-world scenarios. The dataset of each client is segmented into training, validation, and testing subsets. Specifically, the training datasets range from 4,401 to 4,951 samples, each structured as [1, 1024], where "1" represents a single row (or a single time series) and "1024" denotes the length of the time series or the number of features it contains. Each sample can thus be regarded as a time series of length 1024, typically representing signals collected by sensors over a specific time period. The validation datasets contain about 943 to 1,061 samples, and the testing datasets include approximately 944 to 1,062 samples, both maintaining the same structural format as the training data.

ResNet1D is selected as the backbone of the federated learning model, specifically tailored for handling one-dimensional signal data. The weights of the model are initialized using Kaiming Normal initialization for convolutional layers and constant initialization for batch normalization layers, and all parameters will be optimized during the training process. The local training process involved adjusting the weights of the convolutional layers, batch normalization parameters, and fully connected layers used for classification. Synchronization points are established after a predetermined number of local training epochs to effectively align the aggregation cycles. The training process for each client is standardized, including specific epochs, batch sizes, and learning rates, all optimized based on preliminary trials to maximize the performance of the local model.

The experimental framework simulates a federated learning environment using a single GPU of RTX 2080 Ti(11GB), 12 vCPU Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz, 40-GB main memory to ensure a fair comparison between clients. Concerning the hyperparameter choices in Section II, cluster number m is set at 2 considering the full size of 9 clients in the experiment, where the relatively small cluster is suspected to be a Byzantine cluster. Cluster size threshold τ is set at 3 to identify misleading clients and ensure aggregation efficiency. Weight adjustment rate κ is set at 0.01 to guarantee smooth adjustment. The accuracy threshold α is set at 98.5%, 98% and 95% in the ablation, comparison studies and robustness evaluation, respectively. The following ablation, comparison studies and robustness evaluation are carried out under the configuration mentioned above.

B. Ablation Study

1) Ablation Study Settings: As demonstrated in Section II, the BR-MTFL framework consists of three pivotal components, including adapted multi-task learning (A), Byzantine resilience enhancement (B), and weighted aggregation optimization (C). In the ablation study, the three components of BR-MTFL are incrementally incorporated to validate their effectiveness. The number of cycles for the ablation study is set to 11, as BR-MTFL achieves exceptionally high accuracy after 11 cycles. Thus, the results of each component after 11 cycles are taken as the outcomes of the ablation study. This ablation study aims to explore the contributions of these components within the proposed BR-MTFL framework, demonstrating the potential advantages of each component.

2) Ablation Study Results & Analysis: As a baseline, each client trains its model independently without any collaboration,

TABLE IV
ABLATION STUDY OF FINAL ITERATION ACCURACIES ACROSS DIFFERENT FRAMEWORKS.

Algorithm	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Average
Local training	34.75%	59.75%	**100.00%**	80.72%	**100.00%**	89.09%	78.81%	87.61%	**100.00%**	81.19%
BR-MTFL(A)	88.61%	87.92%	81.73%	78.28%	89.64%	89.83%	87.50%	88.67%	88.14%	86.70%
BR-MTFL(A+B)	98.02%	97.78%	**100.00%**	99.68%	**100.00%**	99.89%	**100.00%**	**100.00%**	**100.00%**	99.49%
BR-MTFL(A+B+C)	**98.78%**	**99.05%**	**100.00%**	**99.79%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**99.74%**

resulting in an average accuracy of 81.19%. While some clients, such as Clients 3, 5, and 9, attain perfect accuracies of 100.00%, others show lower performance. In particular, Client 1 and Client 2 achieve accuracies of only 34.75% and 59.75%, respectively. This disparity highlights the limitations of independent local training, especially for clients with limited data. Table IV presents the final experimental results of the ablation study conducted in this paper.

Integrating adapted multi-task learning (A) into the framework increases the average accuracy to 86.70%, with Client 1 and Client 2 achieving final accuracies of 88.61% and 87.92%, respectively. However, some clients, such as Client 3 and Client 4, experience a decrease in accuracy compared to their local training results. This indicates that while multi-task learning facilitates knowledge sharing across related tasks, it may not fully address the challenges posed by data heterogeneity and task customization inherent in the fault diagnosis tasks.

The incorporation of Byzantine resilience enhancement (B) alongside multi-task learning (A) leads to a significant improvement in performance, with the average accuracy rising to 99.49%. Clients 1 and 2, which struggle in other experiments, achieve accuracies of 98.02% and 97.78%, respectively. This accuracy boost is attributed to the framework's ability to effectively relieve the misleading influence of non-malicious Byzantine threats, validating the importance of addressing Byzantine threats in federated learning environments with heterogeneous data.

With the full integration of all three components, the average accuracy further improves to 99.74%, with Client 1 and Client 2 enhancing their accuracies to 98.78% and 99.05%, and most clients achieving 100.00%. This performance is attributed to Component C, which allows the aggregated model to be refined and optimized through continuous local feedback and adaptive weight adjustments. The final model demonstrates exceptional generalization across different operational environments, achieving the highest average accuracy and the most consistent client performances. The complete BR-MTFL framework effectively addresses the challenges posed by data heterogeneity and task customization.

The ablation study underscores the critical contributions of each component in the BR-MTFL framework. While adapted multi-task learning lays the foundation for improved collaboration among clients, the addition of Byzantine resilience enhancement significantly augments the framework's ability to handle data heterogeneity and task customization. The final weighted aggregation optimization fine-tunes the model, maximizing its diagnostic precision. Through the gradual integration of each component, we have demonstrated how the framework overcomes the limitations of local training and standard multi-task learning approaches.

C. Comparison Study

In this study, the BR-MTFL framework is compared with 8 related methods to demonstrate its advantages in dealing with Byzantine threats in MTFL of high-speed trains.

1) Comparison Study Settings: Current federated learning approaches and the handling of Byzantine threats primarily consider scenarios with identical output quantities. However, in the context of high-speed train fault diagnosis, the final model of each client needs to adapt to the specific output requirements of its model. Therefore, all comparative experiments discussed here have adapted the MTFL framework for high-speed train fault diagnosis, enabling adaptive outputs of the relevant fault diagnosis categories according to practical scenarios. The comparison experiment is conducted over 9 cycles, as the BR-MTFL framework achieves convergence within these cycles, attaining an accuracy of over 98% across all client models. Therefore, the results after 9 cycles are selected as the final outcomes for the comparative analysis of the eight experiments. The methods selected for the comparative experiments include:

- **MTFL(BN) [36]:** This method incorporates model personalization for clients in MTFL by utilizing private batch normalization (BN) layers to improve model convergence speed and accuracy of user-specific models.
- **MTFL(MCA) [37]:** This method introduces the maximum correntropy aggregation (MCA) to derive a central value from parameter distributions, enhancing robustness against Byzantine threats.
- **MTFL(Trimmed Mean) [38]:** This method utilizes a coordinate-wise trimmed mean to aggregate gradients, ensuring robustness against data corruption.
- **MTFL(FedSuper) [17]:** This method introduces Byzantine-robust federated learning under supervision (FedSuper), which injects a shadow dataset into local training to supervise and filter out poisoned methods, ensuring robust aggregation under adversarial conditions.
- **MTFL(MOCHA) [39]:** This method proposes a systems-aware optimization of stragglers and fault tolerance. The optimization manages system heterogeneities and maintains efficient computation and communication.
- **MTFL(Multi-Krum) [40]:** This method introduces the Krum to select the most reliable updates by minimizing the sum of squared distances to the closest vectors, and filtering out updates from Byzantine clients.
- **MTFL(FedRadar) [41]:** This method introduces FedRadar, a federated multi-task transfer learning framework for radar-based heartbeat rate and activity monitor-

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

TABLE V
COMPARISON OF FINAL ITERATION ACCURACIES ACROSS DIFFERENT FRAMEWORKS.

Algorithm	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Average
MTFL(BN)	74.95%	24.26%	38.04%	36.65%	31.92%	22.78%	41.95%	73.52%	52.22%	44.03%
MTFL(MCA)	49.53%	52.01%	63.37%	61.55%	59.32%	58.58%	53.50%	53.28%	52.33%	55.94%
MTFL(Trimmed Mean)	89.08%	87.71%	78.72%	75.53%	89.55%	87.82%	85.91%	85.91%	85.91%	85.13%
MTFL(FedSuper)	87.38%	89.09%	96.33%	98.31%	94.44%	96.72%	96.61%	95.23%	92.58%	94.08%
MTFL(MOCHA)	88.14%	85.91%	98.59%	97.67%	99.44%	98.41%	96.29%	95.13%	92.90%	94.72%
MTFL(Multi-Krum)	89.64%	87.29%	**100.00%**	96.72%	**100.00%**	99.89%	**100.00%**	99.47%	99.05%	96.90%
MTFL(FedRadar)	88.78%	88.65%	**100.00%**	99.79%	**100.00%**	99.89%	**100.00%**	99.89%	99.89%	97.43%
MTFL(Median-Krum)	90.87%	89.19%	**100.00%**	**99.89%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	97.77%
BR-MTFL	**99.06%**	**98.83%**	99.81%	99.15%	99.34%	98.41%	99.89%	99.79%	99.26%	**99.28%**

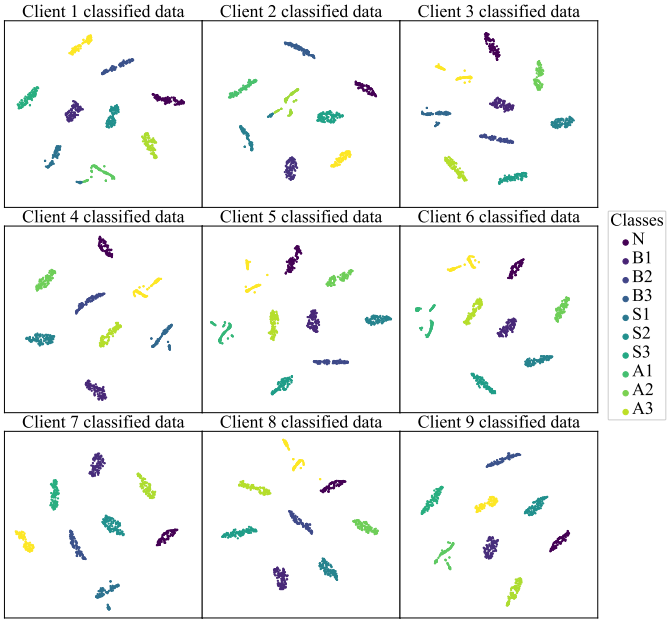

Fig. 4. Visualization of classified data in different clients.

ing (FedRadar) to handle heterogeneous and non-IID data across different devices.

- **MTFL(Median-Krum) [32]:** This method introduces the Median-Krum algorithm, enhancing Byzantine-robustness by combining distance-based statistical strategies to score and select model updates accurately.

2) Comparison Study Results & Analysis: The performance comparison of 8 federated learning frameworks provides critical insights into the challenges posed by data heterogeneity and task specialization in MTFL for high-speed train fault diagnosis. These methods can be broadly categorized into three groups: baseline MTFL without Byzantine resilience enhancement, MTFL with general Byzantine resilience enhancement aggregation methods, and MTFL with specialized optimization algorithms. The T-SNE algorithm is adopted to visualize the classified data in each client after utilizing the BR-MTFL framework in Fig. 4. The comparative experimental results are shown in Table V, and the corresponding metrics of precision, recall, and F1-scores are recorded in Table VI. The split-side violin diagrams shown in Fig. 5 visualize the accuracy distribution of each client.

As the baseline MTFL without Byzantine resilience enhancement, the results indicate that the MTFL(BN) algorithm struggles to cope with the high degree of data heterogeneity and task specialization inherent in fault diagnosis tasks. The average accuracy achieved is 44.03%, with some clients, such as Client 6 and Client 2, recording notably low accuracies of 22.78% and 24.26%, respectively. The poor performance of certain clients highlights the detrimental impact of simplistic backbone models and unfiltered aggregation. Retaining batch normalization layers for each client amplifies information loss during aggregation, further worsening these issues and leaving the model vulnerable to Byzantine threats posed by the misleading updates.

In contrast, the MTFL with general Byzantine resilience enhancement aggregation methods, such as MTFL(Trimmed Mean), MTFL(Multi-Krum), and MTFL(Median-Krum), show improvements in average accuracy. Implementing the MTFL(Trimmed Mean) method slightly raises the average accuracy to 85.13%, yet disparities persist, indicating residual effects of data heterogeneity in this aggregation. The MTFL(Multi-Krum) method attains a higher average accuracy of 96.90%, with Clients 3, 5, and 7 reaching a perfect accuracy of 100%. Despite this, Clients 1 and 2 continue to lag behind, with accuracies of 89.64% and 87.29%. MTFL(Median-Krum) aggregation increases the average accuracy to 97.77%, still with some inconsistencies observed in Clients 1 and 2 achieving lower accuracies of 90.87% and 89.19%. Although these methods enhance robustness by reducing the impact of outlier updates, they primarily focus on reducing malicious attacks rather than addressing the inherent misleading effects introduced by data heterogeneity and task customization, leading to suboptimal global models vulnerable to non-malicious Byzantine threats.

By employing specialized optimization algorithms, namely MTFL(MCA), MTFL(FedSuper), MTFL(MOCHA), and MTFL(FedRadar), the average accuracies also see further enhancements compared with the basic MTFL(BN) framework. The MTFL(MCA) method yields an average accuracy of 55.94%, while MTFL(FedSuper) increases the average accuracy to 94.08%, with most clients above 90%. However, Clients 1 and 2 still record lower accuracies of 87.38% and 89.09%. The MTFL(MOCHA) method increases the average accuracy to 94.72%, with higher accuracies overall, yet Client 2 remains suboptimal at

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

TABLE VI
COMPARISON OF FINAL ITERATION PRECISIONS/RECALLS/F1-SCORES ACROSS DIFFERENT FRAMEWORKS.

Algorithm	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Average
MTFL(BN)	0.82/0.75/0.71	0.23/0.24/0.14	0.46/0.38/0.33	0.28/0.37/0.29	0.23/0.32/0.24	0.15/0.23/0.17	0.32/0.42/0.32	0.73/0.74/0.67	0.48/0.52/0.42	0.41/0.44/0.37
MTFL(MCA)	0.48/0.50/0.42	0.51/0.52/0.43	0.78/0.63/0.60	0.67/0.62/0.56	0.65/0.59/0.56	0.63/0.59/0.53	0.61/0.53/0.48	0.61/0.53/0.48	0.60/0.52/0.46	0.62/0.56/0.50
MTFL(Trimmed Mean)	0.94/0.89/0.86	0.81/0.88/0.84	0.77/0.79/0.73	0.75/0.76/0.69	0.95/0.90/0.87	0.81/0.88/0.84	0.80/0.86/0.82	0.80/0.86/0.82	0.79/0.86/0.81	0.82/0.85/0.81
MTFL(FedSuper)	0.92/0.87/0.85	0.93/0.89/0.87	0.97/0.96/0.96	0.98/0.98/0.98	0.95/0.94/0.94	0.97/0.97/0.97	0.97/0.97/0.97	0.96/0.95/0.95	0.94/0.93/0.92	0.95/0.94/0.93
MTFL(MOCHA)	0.93/0.88/0.85	0.92/0.86/0.82	0.99/0.99/0.99	0.98/0.98/0.98	0.99/0.99/0.99	0.99/0.98/0.98	0.97/0.96/0.96	0.96/0.95/0.95	0.95/0.93/0.93	0.96/0.95/0.94
MTFL(Multi-Krum)	0.94/0.90/0.87	0.93/0.87/0.84	**1.00/1.00/1.00**	0.97/0.97/0.97	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	0.99/0.99/0.99	0.99/0.99/0.99	0.98/0.97/0.96
MTFL(FedRadar)	0.95/0.89/0.85	0.94/0.89/0.85	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**0.99**/0.98/0.97
MTFL(Median-Krum)	0.95/0.91/0.88	0.94/0.89/0.86	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**1.00/1.00/1.00**	**0.99**/0.98/0.97
BR-MTFL	**0.99/0.99/0.99**	**0.99/0.99/0.99**	**1.00/1.00/1.00**	0.99/0.99/0.99	0.99/0.99/0.99	0.99/0.98/0.98	**1.00/1.00/1.00**	**1.00/1.00/1.00**	0.99/0.99/0.99	**0.99/0.99/0.99**

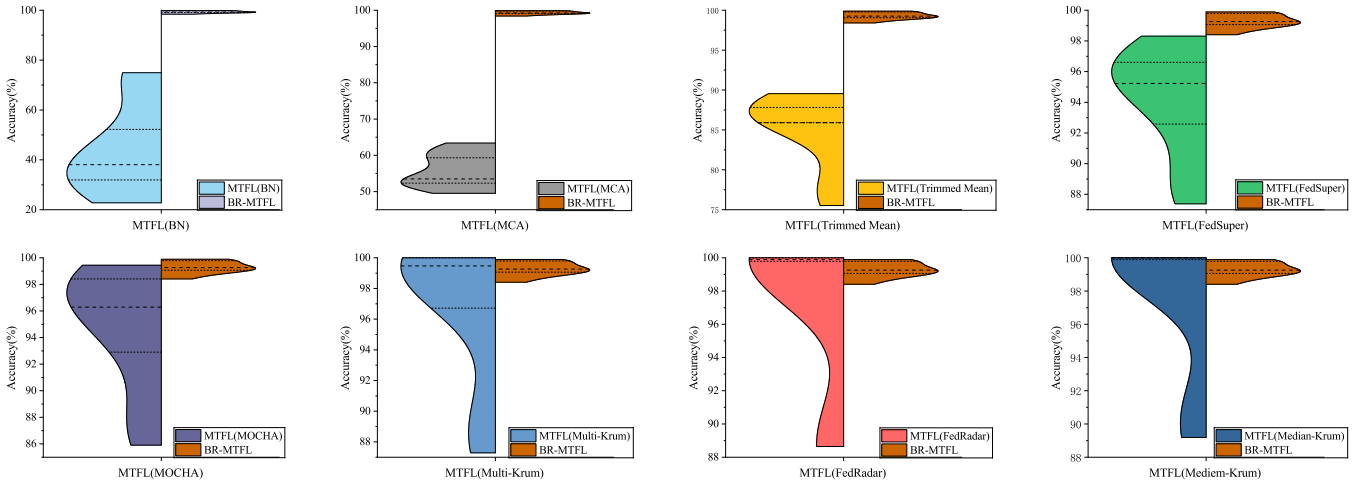

Fig. 5. Split-side violin diagrams of the comprehensive comparison study. The split-side violin diagrams visualize the distribution and statistical characteristics of categorical data using density diagrams. The middle dashed lines represent the median, while the upper and lower dashed lines indicate the third quartile and first quartile, respectively. These features highlight the data's central tendency and variability. The left side corresponds to the accuracy distributions of compared methods, and the right side corresponds to the proposed BR-MTFL framework.

85.91%. The utilization of specialized optimization algorithm MTFL(FedRadar) elevates the average accuracy to 97.43%, with most clients achieving above 99%. Despite this, Clients 1 and 2 attain accuracies of 88.78% and 88.65% respectively, which are still suboptimal. These observations underline the challenge of fully mitigating the misleading effects of task-specific local updates and data heterogeneity, even with advanced optimization methods.

A recurring issue across all compared methods is the inconsistent performance among clients, particularly noticeable in Clients 1 and 2. This inconsistency can be attributed to factors such as data heterogeneity and task specialization biases. Clients possess non-identically distributed data with varying fault types and numbers, leading to highly specialized local models that may not align well during global aggregation. The comparative analysis validates our conjecture that in MTFL applied to high-speed train fault diagnosis, data heterogeneity and task specialization can lead to non-malicious Byzantine threats. Existing methods, while improving robustness to some extent, require further enhancement to fully address these intrinsic challenges.

The BR-MTFL framework demonstrates a remarkable improvement, achieving an average accuracy of 99.28%, with most clients reaching 99% accuracy. The previously underperforming clients, such as Clients 1 and 2, achieve 99.06% and 98.83%, respectively. The superior performance is attributed to the incorporation of a Byzantine resilience enhancement specifically designed to handle the biases arising from client task specialization. This effectively reduces the misleading effects of specialized local updates. By accounting for the non-identical data distributions across clients, the framework ensures that the aggregated global model retains the essential features applicable to all fault types and demonstrates its potential as a robust solution for fault diagnosis systems, offering practical implications for general fault detection tasks in heterogeneous environments.

D. Robustness Evaluation

TABLE VII
SUMMARY OF FOUR SIGNAL AUGMENTATION TECHNIQUES.

Augmentation	Description
Gaussian SNR noise	Add Gaussian noise to the signal by randomly choosing an SNR value.
Mask	Mask a segment of the signal with a length selected randomly to obscure part of the data.
Shift	Shift signal forward or backward by random steps to vary temporal position.
Vertical flip	Vertically flip the signal to reverse amplitude values.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TIM.2025.3550635, IEEE Transactions on Instrumentation and Measurement

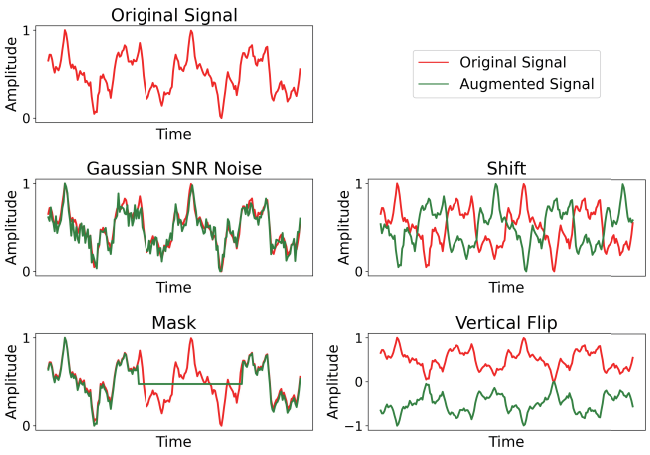

Fig. 6. Visualization of four augmentation techniques.

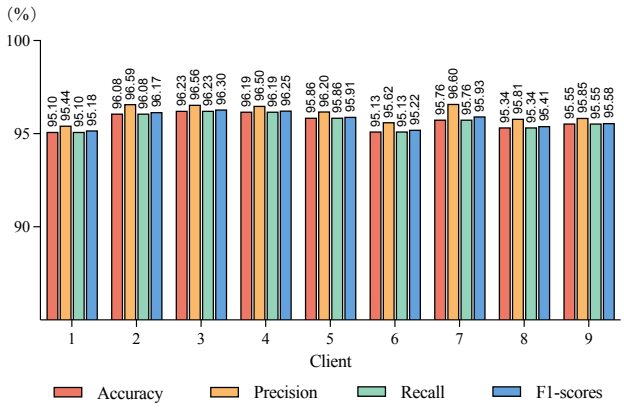

Fig. 7. Visualization of signal augmentation experiments results.

To assess the robustness of BR-MTFL, this experiment applies four signal augmentation techniques to perturb the input data [42]. The model's robustness is evaluated by analyzing its performance under these perturbations, including accuracy, precision, recall, and F1-score. Details of the augmentation methods used are presented in Table VII and Fig. 6.

The results from the signal augmentation experiments, visualized in Fig. 7, demonstrate that while adding noise to the raw signals causes a decrease in overall performance, the proposed BR-MTFL method still achieves strong results across all clients. Specifically, the final test accuracy ranges from 95.10% to 96.23%, with precision, recall, and F1 scores consistently above 95% for each client. While the performance drops compared to the noise-free situation (with accuracy values nearing 99%), it is clear that the method remains robust and performs reliably in noisy conditions.

The method's ability to maintain balanced performance across clients highlights its resilience to perturbations and its effectiveness in mitigating the impact of misleading Byzantine updates. These results confirm that BR-MTFL is capable of providing reliable fault diagnosis models for each client, even under challenging, noisy scenarios. This demonstrates the robustness of the method and its practical applicability in real-world settings, where data noise and imperfections are common.

IV. Conclusion

In this paper, the BR-MTFL framework is introduced to tackle the challenges posed by data heterogeneity and task customization in high-speed train fault diagnosis. The research highlights that non-malicious but misleading updates, stemming from clients' specialized models due to diverse operational environments, can act as Byzantine threats in MTFL. Through comprehensive experiments, we demonstrate that the BR-MTFL framework significantly outperforms existing methods. The ablation study confirms the critical contributions of each component. While the BR-MTFL framework demonstrates significant improvements in safety and reliability, it has limitations in real-time performance due to communication delays. Several avenues for future research remain in high-speed trains and other transportation systems. Incorporating privacy-preserving techniques like differential privacy can enhance the framework's ability to protect sensitive data. Developing real-time implementation strategies in high-speed train systems, including optimizing communication efficiency and reducing computational overhead, would be a significant step forward. The BR-MTFL framework sets a new precedent for developing robust and effective high-speed train systems capable of handling the complexities inherent in real-world applications.

References

[1] Y. Xue, R. Yang, X. Chen, B. Song, and Z. Wang, "Separable convolutional network-based fault diagnosis for high-speed train: A gossip strategy-based optimization approach," *IEEE Transactions on Industrial Informatics*, 2024.

[2] Y. Zhang, N. Qin, D. Huang, and J. Du, "Precise diagnosis of unknown fault of high-speed train bogie using novel FBM-Net," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[3] G. Mirzaeva and K. I. Saad, "Advanced diagnosis of stator turn-to-turn faults and static eccentricity in induction motors based on internal flux measurement," *IEEE Transactions on Industry Applications*, vol. 54, no. 4, pp. 3961–3970, 2018.

[4] M. Ojaghi, M. Sabouri, and J. Faiz, "Performance analysis of squirrel-cage induction motors under broken rotor bar and stator inter-turn fault conditions using analytical modeling," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.

[5] H. Shao, X. Zhou, J. Lin, and B. Liu, "Few-shot cross-domain fault diagnosis of bearing driven by task-supervised ANIL," *IEEE Internet of Things Journal*, 2024.

[6] Z. Huo, M. Martínez-García, Y. Zhang, and L. Shu, "A multisensor information fusion method for high-reliability fault diagnosis of rotating machinery," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2021.

[7] J. Zhang, Y. Wang, K. Zhu, Y. Zhang, and Y. Li, "Diagnosis of interturn short-circuit faults in permanent magnet synchronous motors based on few-shot learning under a federated learning framework," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8495–8504, 2021.

[8] S. Sun, H. Huang, T. Peng, C. Shen, and D. Wang, "A data privacy protection diagnosis framework for multiple machines vibration signals based on a swarm learning algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–9, 2023.

[9] Y. Xiao, H. Shao, J. Lin, Z. Huo, and B. Liu, "BCE-FL: A secure and privacy-preserving federated learning system for device fault diagnosis under non-IID condition in IIoT," *IEEE Internet of Things Journal*, vol. 11, no. 8, pp. 14241–14252, 2024.

[10] J. Chen, J. Li, R. Huang, K. Yue, Z. Chen, and W. Li, "Federated transfer learning for bearing fault diagnosis with discrepancy-based weighted federated averaging," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[11] B. Li and W. Li, "Distillation-based user selection for heterogeneous federated learning," *International Journal of Network Dynamics and Intelligence*, pp. 100 007–100 007, 2024.

[12] J. Du, N. Qin, D. Huang, X. Jia, and Y. Zhang, "Lightweight FL: A low-cost federated learning framework for mechanical fault diagnosis with training optimization and model pruning," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[13] J. Cui, J. Li, Z. Mei, K. Wei, S. Wei, M. Ding, W. Chen, and S. Guo, "Federated meta-learning for few-shot fault diagnosis with representation encoding," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[14] J. Lin, J. Ma, and J. Zhu, "Hierarchical federated learning for power transformer fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[15] J. Mao, F. Chen, B. Jiang, and L. Wang, "Composite fault diagnosis of rotor broken bar and air gap eccentricity based on park vector module and decision tree algorithm," in *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*. IEEE, 2019, pp. 701–706.

[16] H. Chen, B. Jiang, S. X. Ding, and B. Huang, "Data-driven fault diagnosis for traction systems in high-speed trains: A survey, challenges, and perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1700–1716, 2020.

[17] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] Z. Li, H. Wu, Y. Lu, B. Ai, Z. Zhong, and Y. Zhang, "Matching game for multi-task federated learning in Internet of Vehicles," *IEEE Transactions on Vehicular Technology*, 2023.

[19] B.-L. Ye, S. Zhu, L. Li, and W. Wu, "Short-term traffic flow prediction at isolated intersections based on parallel multi-task learning," *Systems Science & Control Engineering*, vol. 12, no. 1, p. 2316160, 2024.

[20] Z.-A. Huang, Y. Hu, R. Liu, X. Xue, Z. Zhu, L. Song, and K. C. Tan, "Federated multi-task learning for joint diagnosis of multiple mental disorders on MRI scans," *IEEE Transactions on Biomedical Engineering*, vol. 70, no. 4, pp. 1137–1149, 2022.

[21] H. Guan, P.-T. Yap, A. Bozoki, and M. Liu, "Federated learning for medical image analysis: A survey," *Pattern Recognition*, p. 110424, 2024.

[22] Q. Yang, J. Zhou, and Z. Wei, "Time perspective-enhanced suicidal ideation detection using multi-task learning," *International Journal of Network Dynamics and Intelligence*, pp. 100 011–100 011, 2024.

[23] R. Jin, Y. Liu, Y. Huang, X. He, T. Wu, and H. Dai, "Sign-based gradient descent with heterogeneous data: Convergence and Byzantine resilience," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[24] Y. Liu, H. He, W. Li, and J. Song, "Edge-based event-triggered consensus control of multi-agent systems against DoS attacks," *International Journal of Systems Science*, pp. 1–11, 2024.

[25] L. Sun, T. Wu, and Y. Zhang, "A defense strategy for false data injection attacks in multi-agent systems," *International Journal of Systems Science*, vol. 54, no. 16, pp. 3071–3084, 2023.

[26] E. C. Ferrer, E. Jiménez, J. L. Lopez-Presa, and J. Martín-Rueda, "Following leaders in Byzantine multirobot systems by using blockchain technology," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1101–1117, 2021.

[27] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, "TDFL: Truth discovery based Byzantine robust federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4835–4848, 2022.

[28] J. T. Mawanza, J. T. Agee, and E. Bhero, "Fault-tolerant dynamic formation control of the heterogeneous multi-agent system for cooperative wildfire tracking," *Systems Science & Control Engineering*, vol. 12, no. 1, p. 2294991, 2024.

[29] C. Berger, H. P. Reiser, J. Sousa, and A. Bessani, "AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1605–1620, 2020.

[30] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 436–447, 2014.

[31] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[32] F. Colosimo and F. De Rango, "Median-Krum: A joint distance-statistical based Byzantine-robust algorithm in federated learning," in *Proceedings of the Int'l ACM Symposium on Mobility Management and Wireless Access*, 2023, pp. 61–68.

[33] S. Li, E. C.-H. Ngai, and T. Voigt, "An experimental study of Byzantine-robust aggregation schemes in federated learning," *IEEE Transactions on Big Data*, 2023.

[34] X. Yang, C. Yang, T. Peng, Z. Chen, B. Liu, and W. Gui, "Hardware-in-the-loop fault injection for traction control system," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 6, no. 2, pp. 696–706, 2018.

[35] X. Chen, R. Yang, Y. Xue, B. Song, and Z. Wang, "TFPred: Learning discriminative representations from unlabeled data for few-label rotating machinery fault diagnosis," *Control Engineering Practice*, vol. 146, p. 105900, 2024.

[36] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630–641, 2021.

[37] Z. Luan, W. Li, M. Liu, and B. Chen, "Robust federated learning: Maximum correntropy aggregation against Byzantine attacks," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[38] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[39] P. Zhao, J. Jiang, and G. Zhang, "FedSuper: A Byzantine-robust federated learning under supervision," *ACM Transactions on Sensor Networks*, vol. 20, no. 2, pp. 1–29, 2024.

[40] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[41] X. Jiang, J. Zhang, and L. Zhang, "FedRadar: Federated multi-task transfer learning for radar-based Internet of Medical Things," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1459–1469, 2023.

[42] Y. Zhan, R. Yang, Y. Zhang, and Z. Wang, "Mitigating catastrophic forgetting in cross-domain fault diagnosis: An unsupervised class incremental learning network approach," *IEEE Transactions on Instrumentation and Measurement*, 2024.