Scalable Machine Learning for Real-Time Fault Diagnosis in Industrial IoT Cooling Roller Systems

Dandan Zhao[a], Karthick Sharma[b], Yuxin Qi[c], Qixun Liu[d], Shuhao Zhang[a]
[a] *National Engineering Research Center for Big DataTechnology and System*
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
[b]*Department of Computer Engineering, University of Sri Jayewardenepura*
[c]*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University*
[d]*CISDI Information Technology (Chongqing) Co., Ltd.*

***Abstract*—The CISDI Hot Rolled Strip Cooling Roller Health Monitoring System (HRSCR-HMS) is a platform widely deployed in hot rolled steel manufacturing, providing real-time monitoring, fault diagnosis, and health management for cooling rollers. Operating in industrial IoT environments with high data velocity and volume, the system faces challenges in scaling real-time fault diagnosis due to evolving fault patterns, severe data imbalance, and the need for both diagnostic accuracy and efficiency. Evaluations of state-of-the-art fault diagnosis (FD) methods, including online continual learning (OCL) algorithms like Camel, reveal their limitations in meeting the real-time adaptability and data processing demands of HRSCR-HMS. To address these challenges, we propose `SRTFD`, a scalable framework tailored for real-time fault diagnosis in industrial IoT systems. `SRTFD` processes high-velocity data streams using three core innovations: *Retrospect Coreset Selection (RCS)* for optimizing training efficiency by reducing redundant data, *Global Balance Technique (GBT)* for robust performance with imbalanced data streams, and *Confidence and Uncertainty-driven Pseudo-label Learning (CUPL)* for adaptive updates with unlabeled data. Experiments on industrial datasets demonstrate that `SRTFD` outperforms competing methods, addressing challenges of imbalance, redundancy, and scalability. Integration within HRSCR-HMS validates `SRTFD` as a practical, scalable solution aligned with the stringent demands of stream data processing in industrial IoT environments.**

I. Introduction

The CISDI Hot Rolled Strip Cooling Roll Health Monitoring System (HRSCR-HMS) revolutionizes operational efficiency and safety in hot rolling processes by providing advanced fault diagnosis (FD) and health management tailored to the complexities of modern steel manufacturing. The system facilitates the early detection and resolution of faults in hot rolled strip cooling rolls, reducing unplanned downtime, extending the lifespan of critical equipment, and minimizing maintenance costs. By processing continuous streams of data from sensors deployed across the manufacturing line, HRSCR-HMS enhances productivity, cost-effectiveness, and operational reliability in industrial processes. CISDI HRSCR-HMS has been deployed in key projects [1], including Panzhihua Steel's 1450 hot rolling line, WISCO's 1580 hot rolling line, Yongfeng Steel's 2250 hot rolling line, and Rizhao Steel's 1 ESP line. Its application in these facilities demonstrates its effectiveness in optimizing equipment performance, ensuring product quality, and meeting the stringent requirements of large-scale production. As an important advancement in industrial monitoring and maintenance, HRSCR-HMS highlights the importance of intelligent fault diagnosis and health management systems in improving the efficiency and sustainability of steel manufacturing.

While HRSCR-HMS has excelled in optimizing performance and product quality, the complexity of modern industrial systems makes it challenging to scale real-time FD for streaming data. Industrial IoT environments supported by HRSCR-HMS produce massive, fast-moving data streams that need scalable, adaptive processing. Traditional FD methods, which use periodic or offline batch analysis, cannot keep up with the continuous and dynamic nature of these streams. This leads to delays or inaccuracies in fault detection, causing costly downtimes and maintenance issues. To overcome this, scalable and adaptive FD methods for streaming data must be integrated into HRSCR-HMS. Stream processing is essential for real-time analysis while preserving diagnostic accuracy and operational efficiency. Additionally, modern FD systems must tackle evolving fault patterns, data imbalance, and limited labeled samples common in continuous industrial processes. Without effective streaming solutions, current FD systems struggle to maintain performance, affecting the reliability of HRSCR-HMS in modern manufacturing.

FD have evolved from simple limit checks in the 19th century to model-based methods that use physical system models and statistical tools such as trend analysis and parameter estimation [13]. They then advanced to knowledge-based FD with signal models and spectral analysis [14]. In modern industrial environments like those supported by HRSCR-HMS, FD must handle new fault types, adapt to changing conditions, process large-scale data in real time, and operate

The first author conducted this work as a visiting researcher at Zhang's Lab. Correspondence: shuhao_zhang@hust.edu.cn.

TABLE I: The existing representative FD methods and the key challenges in modern FD tasks.

Methods	Year	Key Challenges				
		Low-training cost	**High-data efficiency**	**Large-scale data**	**Imbalance data**	**Limited labeled data**
OSELM [2]	2020	×	×	×	✓	×
D-CART [3]	2020	✓	×	×	×	×
TCNN [4]	2020	✓	×	×	×	×
OSSBLS [5]	2021	×	×	×	×	✓
AMPNet [6]	2022	×	×	✓	×	×
1D-CNN [7]	2022	×	×	×	×	✓
TVOTL [8]	2023	×	×	×	×	✓
ODDFD [9]	2023	×	×	×	×	✓
ODCT [10]	2024	×	×	×	✓	×
OLFA [11]	2024	×	×	×	×	✓
MPOS-RVFL [12]	2024	✓	×	×	✓	✓
SRTFD **(Ours)**	2024	✓	✓	✓	✓	✓

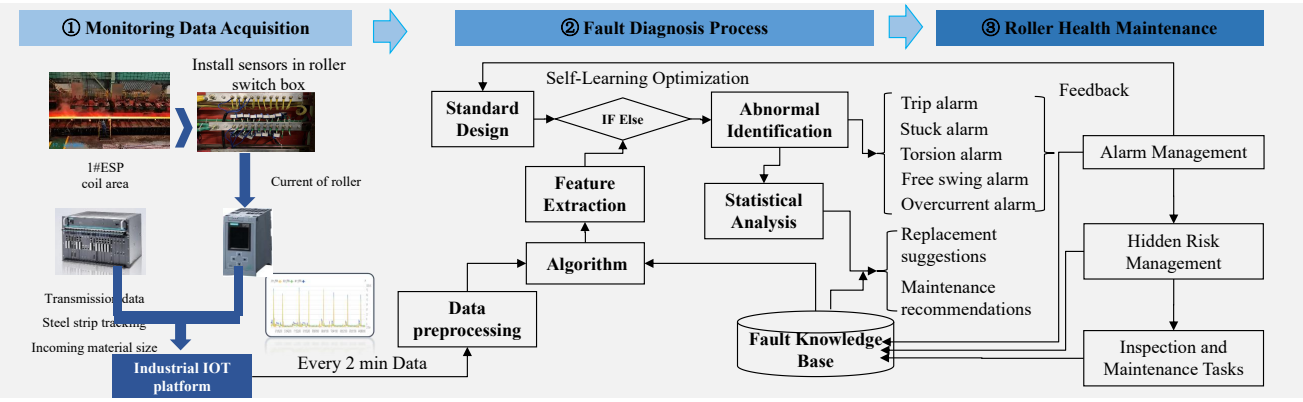


Fig. 1: The HRSCR-HMS framework, which comprises three stages: (1) Data acquisition via sensors and IoT integration; (2) Fault diagnosis through self-learning, feature extraction, and anomaly detection; (3) Maintenance with alarm management and risk handling.

with minimal prior information [12]. This introduces challenges such as managing training costs, ensuring data efficiency, dealing with data imbalance, and working with limited labeled data. Continuous industrial processes also require streaming data handling and maintaining model performance over time.However, existing DL-based FD methods have limitations, as shown in Table I. For example, OSELM [2], D-CART [3], and TCNN [4] struggle with data efficiency and scalability. OSSBLS [5] and ODDFD [10] handle limited labeled data but not data imbalance. AMPNet [6] processes large-scale data but lacks adaptability to new fault types. 1D-CNN [7] faces scalability challenges. TVOTL [8] and ODDFD [9] handle limited labeled data but miss other key challenges. ODCT [10] addresses data imbalance but not scalability or data efficiency. OLFA [11] manages limited labeled data but lacks real-time response and high accuracy.Real-time response is crucial, but only TCNN [4] offers low training costs suitable for quick reactions, unlike AMPNet [6] and OLFA [11]. Most methods struggle with accurate diagnosis using minimal labeled samples. High performance and low training costs are necessary but are often overlooked, limiting their practical usage.

The advent of artificial intelligence and neural networks introduced data-driven FD [15], [16]. Today, deep learning (DL)-driven FD methods dominate due to their ability to leverage large datasets and adapt to complex fault patterns, making them suitable for enhancing HRSCR-HMS functionality [17]. Specifically, online continual learning (OCL) [18] offers a potential solution for real-time fault diagnosis by enabling models to continuously learn from streaming data.Although OCL conceptually fits HRSCR-HMS's needs by supporting real-time updates and adaptation to evolving fault patterns, current algorithms like CAMEL [19] and GoodCore [20] were designed for general-purpose tasks, not the specific needs of FD in industrial settings. Consequently, they struggle with challenges such as redundant monitoring data, severe data imbalance, and scarce labeled samples, leading to suboptimal performance in real-world scenarios. Key limitations include ineffective handling of highly redundant system monitoring data, extreme data imbalance, and the scarcity of labeled samples, which are common in FD tasks in factory environments.Therefore, while OCL provides a promising framework, significant adaptations are needed to tailor it for the unique and demanding requirements of real-time fault diagnosis in industrial settings. This work aims to design a tailored framework to bridge these gaps and meet the specific demands of HRSCR-HMS in modern steel manufacture.

To overcome the limitations of traditional and OCL-based fault diagnosis methods, we propose *SRTFD*, a scalable real-

time fault diagnosis framework designed for high-velocity data streams in industrial IoT systems like HRSCR-HMS. *SRTFD* has three main components: Retrospect Coreset Selection (RCS), Global Balance Technique (GBT), and Confidence and Uncertainty-driven Pseudo-label Learning (CUPL).RCS improves data efficiency by selecting only the most relevant data for model updates, reducing redundant training and lowering training costs. GBT ensures balanced coreset selection to handle data imbalance, maintaining robust model performance across different fault categories. CUPL enables continuous model adaptation by leveraging unlabeled data, addressing the scarcity of labeled monitoring data in industrial processes.Together, these components allow *SRTFD* to handle high data velocity, extreme data imbalance, and minimal labeled samples, ensuring scalability and efficiency in modern industrial environments with large-scale streaming data.

We validated SRTFD through experiments on one real-world industrial process (Hot Roll of Steel, HRS) and two simulation processes (Tennessee Eastman Process, TEP, and CAR Learning to Act, CARLA). The HRS dataset, sourced from HRSCR-HMS monitoring data, includes motor current readings from 294 rollers across five fault categories, featuring severe data imbalance and limited labeled samples. The TEP dataset covers 21 fault categories, while the CARLA dataset simulates single-sensor and multi-sensor faults under three weather conditions, reflecting diverse real-world scenarios. We compared SRTFD against five methods: baseline experience replay (ER) [21], CAMEL [19], ASER [22], AGEM [23], and MPOS-RVFL [12], focusing on class-incremental learning and varying working conditions. SRTFD outperformed these methods, achieving a 2.73% improvement in recall, 1.43% in precision, 1.76% in F1 score, 1.81% in G-means, and a 55.83% reduction in training time compared to state-of-the-art FD and OCL methods.

The main contributions of this work are as follows:

- We propose SRTFD, a novel and scalable real-time fault diagnosis framework tailored to enhance online continual learning (OCL) for industrial fault diagnosis. SRTFD effectively addresses the limitations of existing methods, particularly in handling large-scale, streaming, and imbalanced data with minimal labeled samples.
- SRTFD integrates three innovative components: Retrospect Coreset Selection (RCS) to enhance data efficiency and reduce training costs, Global Balance Technique (GBT) to address data imbalance and maintain robust model performance, and Confidence and Uncertainty-driven Pseudo-label Learning (CUPL) to enable continuous model updates using unlabeled data.
- We validate SRTFD through extensive experiments on real-world and simulated datasets, demonstrating its superior performance and efficiency compared to state-of-the-art methods. Our approach achieved significant improvements across key metrics such as recall, precision, F1 score, G-means, and training time, underscoring its effectiveness and cost-efficiency for real-time fault diagnosis in modern industrial environments.

In addition, we have integrated SRTFD into HRSCR (HRSCR-HMS-SRTFD) in collaboration with HRSCR developers, significantly enhancing fault diagnosis at Panzhihua Steel's factory. While the original HRSCR-HMS had a minimum latency of 5 minutes in fault diagnosis, HRSCR-HMS-SRTFD achieved real-time online diagnosis with comparable accuracy, eliminating this delay. These results demonstrate the practical, scalable, and cost-efficient design of SRTFD, validating its effectiveness for industrial fault diagnosis in modern manufacturing environments. While HRSCR-HMS is proprietary software, the standalone prototype of SRTFD and associated datasets are publicly available at: https://github.com/intellistream/SRTFD.

II. Background and Motivation

A. HRSCR-HMS

As shown in Figure 1, the HRSCR-HMS framework integrates data acquisition, fault diagnosis, and maintenance strategies for hot roll strip cooling roller monitoring and anomaly detection of roller operations. Below is a detailed explanation of its components and functionalities:

Monitoring Data Acquisition: The system employs Hall sensors installed at the V-phase current output of each roller motor within the roller switch box. These sensors collect motor current data in a 4-20 mA format, scaled to a 0-50 A range. The collected data is transmitted to an industrial IoT platform via a Siemens S7-1513 PLC cabinet. The data includes roller current, steel strip tracking, and material size measurements, updated every two minutes.

Fault Diagnosis Process: The system begins with data preprocessing. Signal processing techniques are then employed to extract features associated with different fault types. To accelerate feature extraction, the system utilizes Spark big data tools. Fault detection is performed by analyzing the extracted features, and if a fault is detected, an alert is issued based on predefined operational rules. This step serves as the cornerstone of the entire HRSCR-HMS, as the accuracy and real-time performance of fault diagnosis are pivotal to its overall effectiveness. The proposed SRTFD is meticulously designed to improve diagnostic precision and enhance responsiveness, enabling the system to accurately detect, alert, and respond to roller faults in a timely manner.

Roller Health Maintenance: Once faults are identified, the system transitions to maintenance management by categorizing and prioritizing alerts based on severity, providing temporary replacement and maintenance recommendations to enhance operational efficiency. Feedback from alarm management is used to schedule inspections and ensure timely resolutions, while the fault knowledge base automatically suggests solutions for recurring anomalies, reducing downtime and supporting efficient decision-making.

B. Traditional Data-driven FD

To diagnose potential cooling roller faults, HRSCR-HMS employs several types of traditional data-driven FD models, which typically involve three steps: data collection, model

training, and fault prediction. Let $X^{tr} = \{x_1^{tr}, x_2^{tr}, \ldots, x_{n_{tr}}^{tr}\}$ and $X^{te} = \{x_1^{te}, x_2^{te}, \ldots, x_{n_{te}}^{te}\}$ be the training and testing samples, respectively. Corresponding labels are $Y^{tr} = \{y_1^{tr}, y_2^{tr}, \ldots, y_{n_{tr}}^{tr}\}$ and $Y^{te} = \{y_1^{te}, y_2^{te}, \ldots, y_{n_{te}}^{te}\}$, with n_{tr} and n_{te} being the number of samples. Labels range from 0 to c, where 0 indicates normal samples and 1 to c denotes different fault categories, with c being the total number of fault categories.

Features of collected samples are extracted by $\phi(\cdot)$ before model training. The FD model is then trained using the following loss function:

$$\min_{\theta_t} \sum_{i=1}^{n_{tr}} \mathcal{L}(f(\phi(x_i^{tr}); \theta), y_i^{tr}), \tag{1}$$

where, θ denotes the model parameters, x_i^{tr} is the i-th training sample ($i = 1, 2, \ldots, n_{tr}$). For a new test sample x_j^{te} ($j = 1, 2, \ldots, n_{te}$), the corresponding label y_j^{te} can be predicted by the trained FD model as follows:

$$\hat{y}_j^{te} = f(\phi(x_j^{te}); \theta). \tag{2}$$

Equations (1) and (2) show that fault diagnosis performance depends on the dataset quality. As monitoring data evolves, the model's performance degrades, necessitating data recollection and model retraining. This highlights a key limitation of traditional fault diagnosis methods in meeting real-world demands.

C. Online Continuous Learning

OCL [18] may address several limitations of traditional FD by enabling models to learn incrementally from streaming data. In OCL, monitoring data is collected over time as $X^t \in \mathbb{R}^{d \times n} = \{x_1^t, x_2^t, \ldots, x_n^t\}$ and $Y^t \in \mathbb{R}^{1 \times n} = \{y_1^t, y_2^t, \ldots, y_n^t\}$, where d is the sample dimension, n is the number of samples, and t is the collection time. Thus, the loss function equation (1) becomes:

$$\min_{\theta_t} \sum_{i=1}^{n} \mathcal{L}(f(\phi(x_i^t); \theta_t), y_i^t), \tag{3}$$

where, $x_i^t \in X^t$, $y_i^t \in Y^t$, and θ_t are updated continuously as new data arrives. To update the model at time t without losing previous information, θ_t is adjusted based on θ_{t-1} and the loss function gradient. For models optimized by stochastic gradient descent (SGD) [24], the update rule is:

$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{1}{n} \sum_{i=1}^{n} w_i^t \nabla \mathcal{L}(f(\phi(x_i^t); \theta_{t-1}), y_i^t), \tag{4}$$

where, η is the learning rate and w_i is the weight of data (x_i^t, y_i^t). Typically, $w_i^t = 1$ for all $i \in [n]$, assuming all samples are equally important. This continuous learning process allows the model to adapt to new fault classes and variable working conditions without retraining from scratch.

Although OCL addresses evolving monitoring data, it has a critical drawback: high model update costs when the data volume at time t is large. *Coreset selection* [19] mitigates this by selecting a smaller representative subset from the current batch, reducing data volume and computational requirements. This approach lowers training complexity, allowing fault diagnosis systems to efficiently manage large datasets and enhancing the scalability and performance of OCL models.

D. Motivation

Developing a robust FD framework using OCL and coreset selection is promising but faces challenges due to the unique characteristics of industrial monitoring data.

- **Redundancy:** Monitoring data is highly redundant, making OCL methods inefficient for model updates. Coreset selection within each batch overlooks global information, leading to unnecessary updates and extra training costs when consecutive batches are similar.
- **Data Imbalance:** FD data is imbalanced, with more normal operating data than fault data, and varying frequencies of different faults. This imbalance results in an unbalanced coreset, reducing FD system performance and reliability.
- **Labeled Data Scarcity:** Collecting labeled monitoring data is challenging due to the time-consuming and labor-intensive nature of manual labeling. The lack of labeled data makes model updating difficult.

To address these challenges, we propose `SRTFD`, a scalable real-time fault diagnosis method through OCL. The details of this approach will be introduced in the next section.

III. Problem Statement

Achieving a real-time fault diagnosis framework in complex systems is challenging due to several key issues mentioned in the previous section. These problems are formulated in this section.

Let $D^0 = (X^0, Y^0)$ represent the data used for model pre-training, and X_u^t denote the arriving unlabeled data. The pseudo-labels Y_u^t for X_u^t can be predicted by well-trained model $f_{\theta_t}(\cdot)$. To handle the large-scale data that accumulates over time t, a small subset $S^t = (Z^t, V^t)$ is selected from the pseudo-labeled samples $U_u^t = (X_u^t, Y_u^t)$. The semi-supervised learning strategy is employed by incorporating a small amount of labeled data $D^t = (X^t, Y^t)$ to update the model along with the pseudo-labeled samples. Thus, the loss function at time t becomes:

$$\min_{\theta_t} \sum_{i=1}^{n} \mathcal{L}(f(\phi(x_i^t); \theta_t), y_i^t) + \sum_{j=1}^{s} \mathcal{L}(f(\phi(z_j^t); \theta_t), v_j^t), \tag{5}$$

where $z_j^t \in Z^t$, $v_j^t \in V^t$, and s denotes the total number of samples in the subset S^t. The model parameter update rule becomes:

$$\begin{aligned} \theta_t &\leftarrow \theta_{t-1} - \eta(A + B), \\ A &= \frac{1}{n} \sum_{i=1}^{n} w_i^t \nabla \mathcal{L}(f(\phi(x_i^t); \theta_{t-1}), y_i^t), \\ B &= \frac{1}{s} \sum_{j=1}^{s} w_j^t \nabla \mathcal{L}(f(\phi(z_j^t); \theta_{t-1}), v_j^t), \end{aligned} \tag{6}$$

where, w_i^t and w_j^t are the weights of the labeled and pseudo-labeled samples, respectively, and η is the learning rate. The buffer B^t is crucial in OCL to avoid catastrophic forgetting. It contains data from the pseudo-labeled dataset S^t and the labeled dataset D^t, formulated as $B_t \subseteq (S^t \cup D^t)$. This allows the model to review and learn from past data while updating with new information.

The goal of SRTFD is to select an effective coreset from the current batch of data while considering historical data. First, the coreset must effectively represent the current batch of data. Thus, we have:

$$\min_{\theta_t} \left[\sum_{j=1}^{s} \mathcal{L}(f(\phi(z_j^t); \theta_t), v_j^t) - \sum_{k=1}^{u} \mathcal{L}(f(\phi(x_u^t); \theta_t), y_u^t) \right]. \tag{7}$$

However, the coreset from the above objective function only considers the current batch. To account for historical data, the selected coreset at each time step should not overlap with previous coresets. For the entire coreset $S^T, T = 1, 2, \ldots, t-1$, it should meet the condition $S^t \cap S^T = \varnothing$. Additionally, to address the internal imbalance, the proportion of samples from each class in the coreset should be equal. If there are c fault categories and the coreset size is s, the ideal probability for each class is $\frac{1}{c}$. Thus, we have:

$$\min_{p_l^t} \lambda \left(\frac{1}{c} \sum_{l=1}^{c} \left| p_l^t - \frac{1}{c} \right| \right), \tag{8}$$

where λ is a regularization parameter that balances the importance of the class proportion term. p_l^t is the proportion of samples from class l in the selected coreset S_u^t.

Moreover, to handle model updates with insufficient labeled data, a pseudo-labeling strategy can leverage the abundance of unlabeled data. Directly using the model's predicted labels as pseudo-labels is intuitive but can result in mislabeled data, degrading performance. Therefore, generating accurate pseudo-labels is crucial. Let $v_j^t \leftarrow f(\phi(z_j^t); \theta_{t-1})$ denote the process of generating pseudo-labels. The entire problem of achieving a robust and realistic SRTFD can be formulated as follows:

$$\begin{aligned} \min_{\theta_t, p_l^t} & \left[\sum_{j=1}^{s} \mathcal{L}(f(\phi(z_j^t); \theta_t), v_j^t) - \sum_{k=1}^{u} \mathcal{L}(f(\phi(x_u^t); \theta_t), y_u^t) \right] \\ & + \lambda \left(\frac{1}{c} \sum_{l=1}^{c} \left| p_l^t - \frac{1}{c} \right| \right), \\ \text{s.t.} \quad & S^t \cap S^T = \varnothing, \quad T = 1, 2, \ldots, t-1, \\ & v_j^t \leftarrow f(\phi(z_j^t); \theta_{t-1}). \end{aligned} \tag{9}$$

IV. Methodology

This section introduces the SRTFD framework, illustrated in Figure 2. The prediction and training processes are synchronized, with most data for model updating coming from previously unlabeled data and only a small portion being labeled. This approach is realistic for modern complex systems. The framework consists of three main components: Retrospect Coreset Selection (RCS), Global Balance Technique (GBT), and Confidence and Uncertainty-driven Pseudo-label Learning (CUPL). Each component will be detailed in the following subsections.

A. Retrospect Coreset Selection (RCS)

To optimize fault diagnosis (FD) tasks, we introduce Retrospect Coreset Selection (RCS). RCS selects a representative coreset by considering all historical data and addressing memory constraints through efficient implementation. Specifically, we use the buffer in the system to store samples that encapsulate all historical data, enabling effective RCS. By assessing the similarity between incoming data and buffer data, we eliminate redundancy. When new data arrives, the buffer filters out redundant information, ensuring that coreset selection focuses on novel, non-redundant data. This approach also reduces computational load by bypassing updates when new batches closely resemble existing data.

Given the large-scale nature of the data, directly computing Euclidean distances between new samples and buffer entries is computationally intensive. To mitigate this, we employ batch-wise metrics and cluster the buffer data. Let $B^t = \{(x_{b1}^t, y_{b1}^t), \cdots, (x_{bn}^t, y_{bn}^t)\}$ represent the buffer at time t, where $x_{bi}^t \in (X^t \cup X_u^t)$ and $y_{bi}^t \in (Y^t \cup Y_u^t)$, with $i = 1, 2, \cdots, bn$, and bn being the total number of buffer samples at time t. Labels y_{bi}^t fall within $\{0, 1, ..., bc\}$, where bc is the number of classes at time t. We partition buffer data into bc clusters based on these labels, denoted as $\{X_{b1}^t, X_{b2}^t, \ldots, X_{bc}^t\}$. By clustering the buffer data, we reduce the computational burden, enabling batch-wise comparison instead of individual distance calculations.

For the newly arrived data X_u^t, we employ the MiniBatchKMeans algorithm [25], a highly efficient clustering method designed for large-scale datasets, to partition the data into different clusters. This algorithm accelerates computation and reduces memory usage by processing data in small batches. By clustering X_u^t into uc clusters, denoted as $\{X_{u1}^t, X_{u2}^t, \ldots, X_{uc}^t\}$, where X_{ui}^t represents the ui-th cluster and $ui = 0, 1, \ldots, uc$. The number of clusters uc must be less than or equal to the number of samples in the arriving batch un, and there is no clustering performed if uc equals un. Then, the similarity between X_{ui}^t and B_{bi}^t can be calculated by Kullback-Leibler (KL) divergence [26], which is denoted as:

$$d_{bi,ui}^t = KL(X_{bi}^t, X_{ui}^t). \tag{10}$$

If $d_{bi,ui}^t$ is less than a threshold τ, the cluster X_{ui}^t is considered redundant. Then, we have

$$S^t = \{ \bigcup_{ui=1}^{uc} X_{ui}^t \mid \bigcup_{bi=1,ui=1}^{bc,uc} d_{bi,ui}^t \leq \tau \} \tag{11}$$

This buffer retrospective process ensures that only the most informative and non-redundant data points are retained in the coreset. Thus, the goal of $S_u^t \cap S_u^T = \varnothing$, for $T = 1, 2, \ldots, t-1$, is achieved.

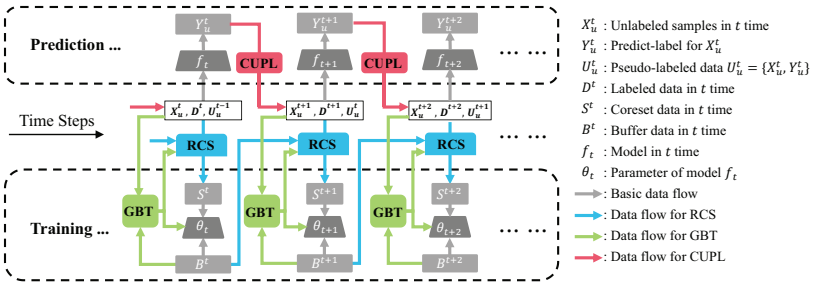


Fig. 2: SRTFD framework conducts fault diagnosis in two stages: Prediction and Training. In the Prediction stage, unlabeled samples are pseudo-labeled and combined with labeled data for model updates via CUPL. In the Training stage, the model is iteratively trained using RSC and GBT to ensure effective learning despite class imbalances.

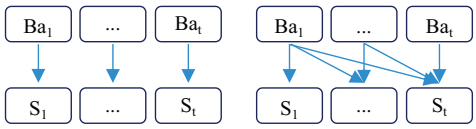


Fig. 3: Comparison of existing [19] and our coreset selection approaches. The $\mathrm{Ba_t}$ and $\mathrm{S_t}$ denote the batch data and selected coreset in t time, respectively.

Selecting the most representative samples is recognized as an NP-hard problem. However, in the recent OCL technique Camel [19]. Transformed this challenge into a submodular maximization problem. Building on their work, we iteratively select two samples with the largest Euclidean distance until the coreset size requirement is satisfied. The selection of the coreset S^t in RCS is defined as follows:

$$S^t = \{Z^t \subseteq \bigcup_{ui=1}^{uc} X_{ui}^t \mid \bigcup_{bi=1,ui=1}^{bc,uc} d_{bi,ui}^t \leq \tau, d_{\max}\}, \quad (12)$$

here, $d_{\max} = \max_{i,j\in[s]} ||z_i - z_j||$, with $z_i, z_j \in Z^t$ and s being the size of the coreset. Z^t is a subset of $\bigcup_{ui=1}^{uc} X_{ui}^t$. The distance metric $d_{bi,ui}^t$ ensures that only data points with a similarity measure above a threshold τ are selected. $d_{\max}$ represents the maximum Euclidean distance between any two samples within Z^t.

B. Global Balance Technique (GBT)

As highlighted in the introduction, the imbalance problem significantly impacts the performance of fault diagnosis systems. Current online continual learning (OCL) methods rarely address this issue adequately. They typically attempt to mitigate it by selecting a coreset that appears balanced from the current batch data, prioritizing samples from less-represented classes. However, this often results in a pseudo-balanced coreset that does not reflect a truly balanced distribution. For instance, if a class has many samples in the buffer but few in the current batch, existing OCL methods will select more samples from this class, neglecting genuinely underrepresented classes. Furthermore, in cases of extreme imbalance, selecting a balanced coreset alone cannot resolve the problem, as the selected coreset often remains unbalanced. Therefore, we propose GBT, which addresses this issue based on two critical factors affecting fault diagnosis performance: the quality of training data and the frequency of model updates.

From the perspective of training data, feeding a balanced dataset into the model will undoubtedly enhance its performance. However, the FD model is trained on both the current batch of data and the data stored in the buffer. Focusing solely on the current batch data makes it easy to obtain a pseudo-balanced coreset. Thus, it is essential to consider both the buffer samples and the categories, and the corresponding objective function (8) becomes:

$$\min \left(\frac{1}{c} \sum_{l=1}^{c} \left| ps_l^t - \frac{1}{c} \right| \right) + \left(\frac{1}{bc} \sum_{l=1}^{bc} \left| pb_l^t - \frac{1}{bc} \right| \right) \quad (13)$$

where ps_l^t and pb_l^t represent the true probabilities of the l-th classes in the current coreset and buffer, respectively. bn represents the total sample size in the buffer, and bc_t denotes the number of classes in the buffer at t time. This approach ensures a genuinely balanced dataset by selecting samples from the least represented classes in both the buffer and the current coreset.

From the perspective of model updates, considering the scenario where certain classes have a significantly low number of samples, with $ps_l^t \ll \frac{1}{c}$ and $pb_l^t \ll \frac{1}{bc}$, it becomes evident that selecting a balanced coreset is unfeasible. Therefore, we introduce the following loss function:

$$\mathcal{L} = -\sum_{i=1}^{b} (1-p_i)^\gamma \log(p_i) - \sum_{j=1}^{s} \alpha (1-p_j)^\gamma \log(p_j) \quad (14)$$

where $p_i = f_{\theta_t}(x_{bi})$ and $p_j = f_{\theta_t}(z_j)$ are the predicted probabilities for data from the buffer and coreset, respectively. α represents the data weights in coreset. The modulating factor γ reduces the loss from easy-to-classify examples, encouraging the model to focus on harder cases. Equation (14) directs the model's attention towards underrepresented and difficult classes, effectively mitigating the class imbalance issue.

C. Confidence and Uncertainty-driven Pseudo-label Learning (CUPL)

The proposed CUPL module enhances the SRTFD training process by utilizing both labeled and unlabeled data. The critical components of this method include generating pseudo-labels and employing a selection strategy to incorporate these labels into the training set. Here, the pseudo-labels v_j are determined directly by the maximum probability, expressed as $v_j = \Gamma[\max(p_j)]$. For the selection strategy, the conventional approach for selecting pseudo-labeled samples is to choose those with high confidence. However, since most newly arriving data are normal data, the model shows high confidence in predicting these normal instances but low confidence in identifying rare faults. Thus, relying solely on high-confidence selections can degrade overall performance in FD tasks.

Inspired by the work of Mamshad et al. [27], we introduce both positive and negative label selection. Specifically, samples are selected as positive pseudo-labels when the model's predicted probability is high, and as negative pseudo-labels when the probability is extremely low, which can be denoted as follows:

$$g_j = \mathbf{1}[p_j \geq \tau_p] + \mathbf{1}[p_j \leq \tau_n], \tag{15}$$

where g_j is an indicator that determines whether the sample j is selected. The indicator function $\mathbf{1}[\cdot]$ returns 1 if the condition inside the brackets is true and 0 otherwise. The thresholds τ_p and τ_n are used to select positive and negative pseudo-labels, respectively.

Additionally, it has been demonstrated that predictions with low uncertainty are more likely to result in correct pseudo-labels, as shown by Mamshad's research. Therefore, we incorporate an uncertainty constraint into the positive and negative pseudo-label selection process. The uncertainty of model predictions can be estimated using Monte Carlo dropout (MCdropout) [28]. MCdropout estimates uncertainty by incorporating dropout during the inference phase. The process involves performing multiple stochastic forward passes through the network, each time applying dropout with a certain probability p. For a layer with output $\mathbf{h}$, the output with dropout is $\mathbf{h}' = \mathbf{h} \odot \mathbf{d}$, where $\mathbf{d} \sim \text{Bernoulli}(p)$. Repeating this process T times results in a set of predictions $\{\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, \ldots, \hat{\mathbf{y}}^{(T)}\}$. The mean prediction is calculated as $\hat{\mathbf{y}}_{\text{mean}} = \frac{1}{T}\sum_{t=1}^{T} \hat{\mathbf{y}}^{(t)}$, and the uncertainty is estimated using the variance $\hat{\sigma}^2 = \frac{1}{T}\ \sum_{t=1}^{T}(\hat{\mathbf{y}}^{(t)} - \hat{\mathbf{y}}_{\text{mean}})^2$. This approach is simple to implement, requiring no complex modifications to the model architecture, and provides valuable insights into the model's confidence in its predictions, especially with uncertain or complex input data.

By utilizing both the confidence and uncertainty of network predictions, a more accurate subset of pseudo-labels can be selected. The selection function becomes:

$$g_i = \mathbf{1}\left[\hat{\sigma}^2 \leq \kappa \text{ and } p_j \geq \tau_p\right] + \mathbf{1}\left[\hat{\sigma}^2 \leq \kappa \text{ and } p_j \leq \tau_n\right], \tag{16}$$

where κ is the uncertainty thresholds.

V. Experiments

A. Experimental Setup

Experiments were conducted on an Intel(R) Xeon(R) w7-3455 system with an NVIDIA RTX 6000 Ada Generation GPU (48GB GDDR6) and 512GB of RAM. The software used includes Python 3.12.4 and PyTorch 2.3.0 with CUDA 12.1.

Datasets: Table II summarizes the datasets used in our experiments. (1) Hot Roll of Steel (HRS): The hot rolling process in steel manufacturing requires monitoring conveyor rollers to prevent billet deformation, surface defects, downtime, and increased costs. This dataset, from an actual hot rolling steel industry, includes data from 294 rollers recorded via motor current signals and converted to digital data. It encompasses five fault categories: roller swing, roller stuck, overcurrent, squeaking, and base deformation. Due to the random occurrence of these faults, the dataset is highly unbalanced and contains limited samples. (2) Tennessee-Eastman process (TEP): It is widely used to validate fault diagnosis methods. It includes 21 distinct fault categories with training and testing sets. In this study, 4320 normal samples and 800 samples for each fault are used. (3) CAR Learning to Act (CARLA): Yan et al. [11] collected this dataset using Dosovitskiy's [29] autonomous driving simulator. It features single-sensor and multi-sensor faults across three maps: rainy, cloudy, and sunny conditions. There are 9 categories of single-sensor faults and 4 categories of multi-sensor faults. Each map had a 30-minute simulation with sensor data sampled at 60Hz.

Testing Scenarios: We conduct experiments on three datasets under two scenarios: class-incremental and variable working conditions. The proposed method supports online continuous learning, eliminating the need to divide data into training and testing sets. We first initialize the network using 1000 normal samples. As monitoring data arrives, labeled samples and reliable samples with predicted labels from previous tasks are used for model training while predicting the new data. For class-incremental scenarios, normal samples are randomly divided equally into each task, and each fault sample appears in each task in turn. For varying working conditions, only the CARLA dataset includes three conditions, while the HRS and TEP datasets each have one, as shown in Table II. We gradually introduce noise into the HRS and TEP samples to simulate different working conditions. In the CARLA dataset, monitoring data evolve randomly from condition 1 to condition 2, and then to condition 3.

Competing Methods: We selected five methods, including state-of-the-art FD and four advanced OCL approaches, as benchmark algorithms for performance comparison: 1) Baseline: A basic model trained using experience replay (ER) [21], achieving online continuous learning by replaying examples from previous tasks. 2) Adversarial Shapley Value Experience Replay (ASER) [22]: Maintains learning stability and optimizes new class boundaries in the online class-incremental setting. 3) Camel [19]: An advanced OCL method that accelerates model training and improves data efficiency through coreset selection. 4) Averaged Gradient Episodic

TABLE II: The description of datasets

Dataset	Normal samples	Number of fault classes	Number of each fault	Dimension	Number of working cond.	Total samples
HRS	36333	5	1783/8114/1533/83/83	120	1	47929
TEP	4320	21	800	52	1	31200
CARLS (single-sensor)	89166	9	2404/1803/2404/2404 /2404/1604/2004/1803/2004	10	3	108000
CARLS (multi-sensor)	100885	4	1604/1604/1503/2404	10	3	108000

Memory (AGEM) [30]: Evaluates OCL efficiency in terms of sample complexity, computational cost, and memory usage. 5) MPOS-RVFL [12]: An advanced ML-based FD method focused on real-time fault diagnosis with imbalanced data.

Implementations: The transformer network structure is used for all DL-based benchmarks. The basic neural network consists of an encoder, decoder, transformer encoder as the feature extractor, and a fully connected layer as the predictor. The encoder's hidden layer dimensions are 500, 500, and 2000, respectively, and the decoder's dimensions are 2000, 500, and 500. Model training used a learning rate of 0.0001 with the SGD optimizer, a maximum of 200 epochs, and a batch size of 100.

B. Performance Comparison

Given the imbalanced nature of the experimental datasets, accuracy is not reliable. Therefore, we used metrics specifically designed for imbalanced datasets: Recall, Precision, G-mean, and F1 score. Recall measures the proportion of actual positives correctly identified, while Precision measures the proportion of predicted positives that are correct. G-mean balances sensitivity and specificity, and the F1 score is the harmonic mean of Precision and Recall. We also compared model training time. Our results are averaged across all tasks after the final update, denoted as Avg-End-Rel for Recall, Avg-End-Pre for Precision, Avg-End-Gmean for G-mean, and Avg-End-F1 for F1 score.

Class-incremental: The comparison results for class-incremental methods are shown in the left part of Table IV. Our method requires the least training time on all datasets and outperforms other DL-based methods. Although the ML-based method MPOS-RVFL requires less training time, its performance is significantly inferior to other DL methods. The ER method is competitive, outperforming our method by 2.3% in the four metrics on the CARLA-S dataset. However, ER's training time is 4.56 times longer, at 222.44 seconds compared to our 48.40 seconds. These results highlight the effectiveness of `SRTFD` in balancing high performance with reasonable training times across various datasets and conditions.

Variable working conditions: The right part of Table IV shows the performance comparison under variable working conditions across three datasets. `SRTFD` consistently outperforms other approaches, achieving the highest scores in most performance metrics. While the ER method outperforms our method on the CARLS-S dataset, it requires much more training time (ER: 456.87 seconds vs. `SRTFD`: 95.99 seconds). The Camel method maintains the second shortest training time across all datasets, leading on the TEP dataset with 105.18 seconds, 20.26 seconds shorter than `SRTFD` (125.44 seconds). However, Camel's overall performance is 9.7% lower than `SRTFD` in other metrics. These results highlight the effectiveness of `SRTFD` in balancing high performance with reasonable training times across various datasets and conditions.

Moreover, the distinct challenges of class-incremental and variable working condition scenarios are reflected in both performance and computational efficiency. For class-incremental scenarios, the primary difficulty lies in learning new fault classes while avoiding catastrophic forgetting of previous classes. Traditional methods like ER rely on large replay buffers to retain old-class samples, but severe class imbalance persists as new classes are introduced with limited data. This forces our RSC module to select a minimal yet balanced coreset by aligning the smallest class size, resulting in faster training times but potential underrepresentation of data distributions. In contrast, variable working condition scenarios face shifting fault distributions under abrupt condition changes. While all methods exhibit longer training times, this stems from RSC selecting larger core sets to capture multi-modal distributions, which improves performance but increases computational load. The performance gap also reveals a limitation: enforcing strict class balance in class-incremental scenarios may overly restrict core set size, leaving insufficient samples to represent complex new-class features.

C. Ablation Study

To evaluate the contribution of each component in `SRTFD`, we separately removed the RSC, GBT, and CUFL components and conducted experiments on all datasets. We compared the performance before and after removing each component under class-incremental and variable working conditions. From Table 4, the following three conclusions can be drawn.

First, the RSC module effectively reduces model training time. When comparing *SRTFD* with *SRTFD w/o RSC*, the training duration of *SRTFD* is significantly shortened across all datasets while maintaining comparable performance across four metrics. For instance, in the HRS dataset within a class-incremental setting, the training time is reduced by 80.66%, and the Avg-End-rel improves by 13.22%.

Second, the GBT module effectively balances training time and FD performance. The GBT mechanism ensures that the coreset is balanced at the cost of discarding some data, as it enforces class balance by aligning the sample quantities across all classes with the least-represented category in the current coreset. As demonstrated in Table IV, *SRTFD* consistently outperforms *SRTFD w/o GBT* across various scenarios, including HRS (class-incremental), TEP (variable working condition) and CARLS-M (variable working condition), while

TABLE III: Performance comparison of the SRTFD and other approaches across three datasets in two scenarios.

Dataset	DL/ML-based	Methods	Class-incremental					Variable working condition				
			Avg-End-Rel	Avg-End-Pre	Avg-End-F1	Avg-End-Gmean	Training-Time (s)	Avg-End-Rel	Avg-End-Pre	Avg-End-F1	Avg-End-Gmean	Training-Time (s)
HRS	ML-based	MPOS-RVFL	0.3533	0.4083	0.3760	0.3784	9.04	0.1269	0.1667	0.1441	0.1454	6.73
	DL-based	ASER	0.3169	0.3750	0.2986	0.3397	355.35	0.1937	**0.5990**	0.2320	0.3373	225.72
		Camel	0.5795	0.4926	0.5082	0.5327	114.64	0.1520	0.5025	0.2185	0.2762	91.29
		AGEM	0.6032	0.5211	0.5382	0.5592	176.79	0.1468	0.5597	0.2142	0.2866	117.98
		ER	0.5331	0.4594	0.4659	0.4935	144.48	0.1660	0.5468	0.1803	0.2937	93.75
		SRTFD	**0.6831**	**0.5663**	**0.5881**	**0.6187**	**56.17**	**0.5914**	0.5215	**0.5365**	**0.5552**	**59.42**
TEP	ML-based	MPOS-RVFL	0.0992	0.1678	0.1151	0.1228	14.33	0.0093	0.0455	0.0154	0.0205	4.27
	DL-based	ASER	0.1524	0.0737	0.0857	0.1042	348.72	0.3259	0.3258	0.2985	0.3258	350.97
		Camel	0.1619	0.1173	0.1263	0.1359	116.29	0.2364	0.2773	0.2289	0.2559	**105.18**
		AGEM	0.1534	0.0993	0.1029	0.1208	139.47	0.2975	0.2643	0.2453	0.2803	131.21
		ER	0.1721	0.1471	0.1491	0.1567	121.97	0.3351	0.3509	0.3219	0.3428	127.91
		SRTFD	**0.1950**	**0.1606**	**0.1572**	**0.1742**	**77.10**	**0.3450**	**0.3569**	**0.3358**	**0.3508**	125.44
CARLS-S	ML-based	MPOS-RVFL	0.2694	0.2929	0.2787	0.2804	20.57	0.3755	0.4183	0.3942	0.3955	29.80
	DL-based	ASER	0.2749	0.2642	0.2179	0.2579	563.18	0.5155	0.4378	0.4366	0.4718	1181.92
		Camel	0.3896	0.3657	0.3274	0.3711	156.65	0.5733	0.5025	0.4925	0.5319	357.42
		AGEM	0.4706	0.4501	0.4178	0.4569	271.62	**0.6249**	0.5031	0.4997	0.5543	560.76
		ER	0.5295	**0.5916**	**0.5077**	**0.5587**	222.44	0.5907	**0.5482**	**0.5550**	**0.5682**	456.87
		SRTFD	**0.5519**	0.5160	0.4972	0.5306	**48.04**	0.5982	0.5115	0.5117	0.5521	**95.99**
CARLS-M	ML-based	MPOS-RVFL	0.4410	0.4567	0.4484	0.4486	29.89	0.4608	0.4833	0.4715	0.4718	28.92
	DL-based	ASER	0.3701	0.4042	0.2625	0.3723	525.43	0.6985	0.6428	0.6488	0.6679	621.28
		Camel	0.5610	0.5303	0.5021	0.5388	148.50	0.5775	0.4836	0.4720	0.5221	230.10
		AGEM	0.5503	0.5183	0.5166	0.5326	262.09	0.6062	0.4068	0.3983	0.4784	355.06
		ER	**0.6269**	0.6403	0.6172	0.6332	229.13	0.7856	0.6886	0.7109	0.7330	281.76
		SRTFD	0.6108	**0.7133**	**0.6400**	**0.6568**	**45.57**	**0.8223**	**0.7089**	**0.7314**	**0.7617**	**37.35**

TABLE IV: Performance of SRTFD compared to its variants without each component across three datasets.

Dataset	Methods	Class-incremental					Variable working condition				
		Avg-End-Rel	Avg-End-Pre	Avg-End-F1	Avg-End-Gmean	Training-Time (s)	Avg-End-Rel	Avg-End-Pre	Avg-End-F1	Avg-End-Gmean	Training-Time (s)
HRS	SRTFD w/o RSC	0.6034	0.5619	0.5725	0.5815	362.29	**0.7196**	0.6808	**0.6862**	**0.6998**	307.35
	SRTFD w/o GBT	0.6437	0.5464	0.5699	0.5919	218.17	0.6750	**0.7005**	0.6858	0.6875	174.93
	SRTFD w/o CUPL	0.5897	0.5003	0.5127	0.5405	**56.00**	0.4109	0.4311	0.4019	0.4198	**18.08**
	SRTFD	**0.6831**	**0.5663**	**0.5881**	**0.6187**	56.17	0.5914	0.5215	0.5365	0.5552	59.42
TEP	SRTFD w/o RSC	**0.2174**	0.1642	0.1743	**0.1880**	168.05	0.3203	0.3469	0.3119	0.3332	159.60
	SRTFD w/o GBT	0.2053	**0.1889**	**0.1793**	0.1857	140.25	0.2036	0.2105	0.1919	0.2070	115.17
	SRTFD w/o CUPL	0.1055	0.0824	0.0803	0.0915	87.94	0.2343	0.2615	0.2221	0.2474	**65.86**
	SRTFD	0.1950	0.1606	0.1572	0.1742	**77.10**	**0.3450**	**0.3569**	**0.3358**	**0.3508**	125.44
CARLS-S	SRTFD w/o RSC	0.5133	0.5169	0.4769	0.5135	334.64	0.6878	**0.5972**	**0.6133**	0.6390	986.64
	SRTFD w/o GBT	**0.5922**	**0.5518**	**0.5410**	**0.5702**	222.49	**0.7340**	0.5878	0.6098	**0.6526**	689.73
	SRTFD w/o CUPL	0.3907	0.4732	0.3684	0.4265	61.42	0.3620	0.3302	0.3221	0.3450	**34.19**
	SRTFD	0.5519	0.5160	0.4972	0.5306	**48.04**	0.5982	0.5115	0.5117	0.5521	95.99
CARLS-M	SRTFD w/o RSC	0.6222	0.6360	0.6121	0.6262	673.35	0.7905	0.6881	**0.7134**	0.7363	300.65
	SRTFD w/o GBT	**0.7389**	**0.7149**	**0.7176**	**0.7255**	190.73	**0.8236**	0.6199	0.6379	0.7044	99.53
	SRTFD w/o CUPL	0.5104	0.4865	0.4446	0.4939	55.39	0.7764	0.6644	0.6906	0.7168	44.34
	SRTFD	0.6108	0.7133	0.6400	0.6568	**45.57**	0.8223	**0.7089**	**0.7314**	**0.7617**	**37.35**

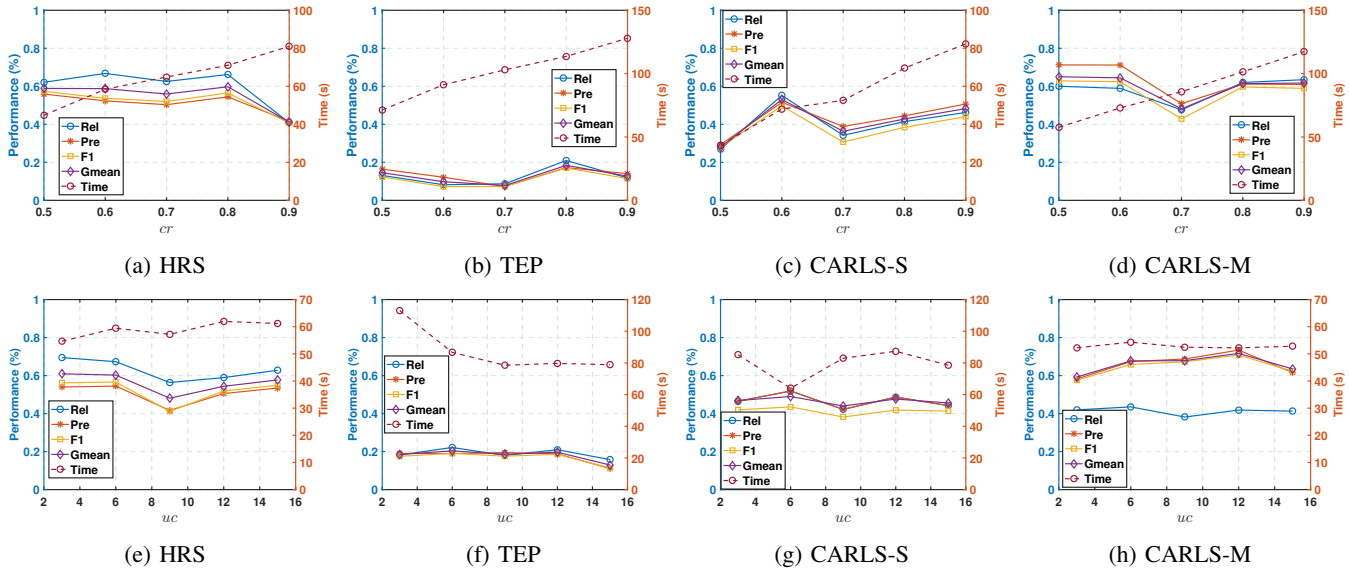
significantly reducing training time. For example, in HRS class-incremental tasks, SRTFD achieves a 74.3% reduction in training time alongside a 3. 2% improvement in the F1 score. In cases where *SRTFD w/o* GBT exhibits marginal performance gains (e.g., 0.01–0.04 increases in metrics such as Avg-End-Rel or Gmean), it incurs a substantial computational cost, with training times 2–6 times longer. This is particularly evident in TEP class-incremental scenarios (140.25 s vs 77.10 s) and CARLS-S variable working conditions (689.73 s vs 95.99 s), where minor performance improvements do not justify a significant increase in computational overhead.

Lastly, the CUPL module significantly improves FD performance. While the training time for *RSTFD w/o CUPL* may be shorter in certain cases, it is evident that its performance is markedly inferior to that of *RSTFD*. A comparison between *RSTFD* and *RSTFD w/o CUPL* reveals that the extensive use of pseudo-labeled samples for model training substantially improves FD performance. For instance, the Avg-End-Rel of *RSTFD* increased by 9.34% in the HRS dataset within a class-incremental setting, while the training times are similar (56.17 vs. 56.00 seconds).

SRTFD excels in balancing training efficiency and FD performance. The RSC module drastically reduces training time (e.g., 80.66% reduction on HRS) by dynamically selecting representative samples. The GBT module enhances robustness under class imbalance, improving F1 scores by 3% on HRS, while the CUPL module boosts generalization using pseudo-labels. However, limitations persist: (1) Pseudo-labeling risks noise sensitivity if initial models are poorly calibrated; (2) GBT's class-balancing discards excess majority-class data, potentially underutilizing information; (3) Efficiency-performance trade-offs remain context-dependent—e.g., marginal gains (0.04 Gmean) without GBT in CARLS-M incur 6× longer training. Future work could refine pseudo-label thresholds or hybrid sampling to address these issues.

D. Analysis and Discussion

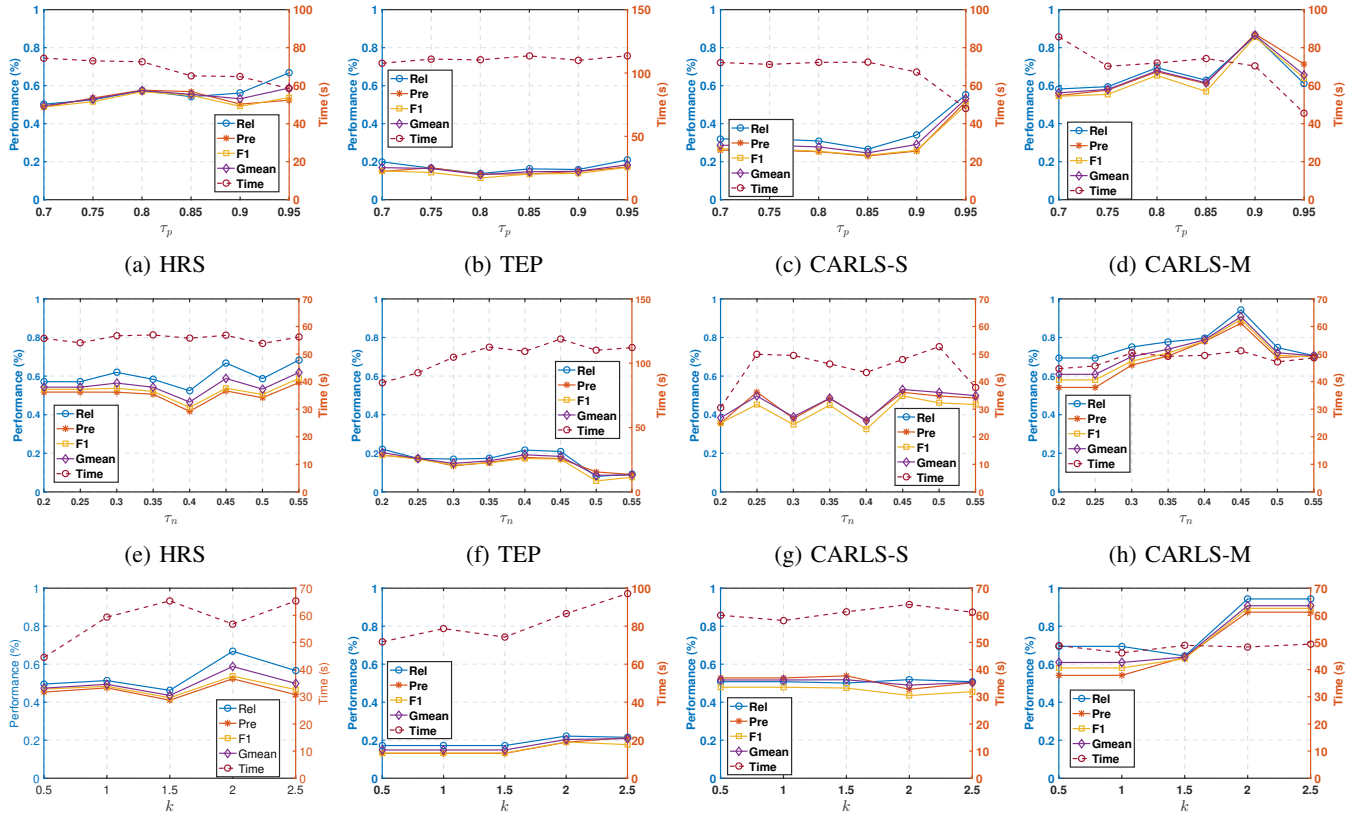
1) Effects of coreset ratio cr and cluster number uc: The impact of these two parameters is illustrated in Figure 4. Panels (a-d) in Figure 4 show the performance across all datasets in the class-incremental scenario for coreset ratios of [0.5, 0.6, 0.7, 0.8, 0.9]. From this figure, it can be observed that as the coreset ratio increases, the number of samples in the coreset grows, leading to longer training times for the model. Additionally, when the coreset ratio is 0.6, the performance of the HRS, CARLS-S, and CARLS-M models is relatively better, and the training time is shorter. Additionally, Figures 4 (e-h) demonstrate the model performance with different numbers of clusters, uc, set to [3, 6, 9, 12, 15]. From the figures, it can be observed that although the training performance of the models is relatively stable across different uc values, the choice of uc is related to the number of categories in the dataset. For



(a) HRS (b) TEP (c) CARLS-S (d) CARLS-M

(e) HRS (f) TEP (g) CARLS-S (h) CARLS-M

Fig. 4: Performance of different coreset ratios and cluster numbers uc on all datasets within class-incremental.



(a) HRS (b) TEP (c) CARLS-S (d) CARLS-M

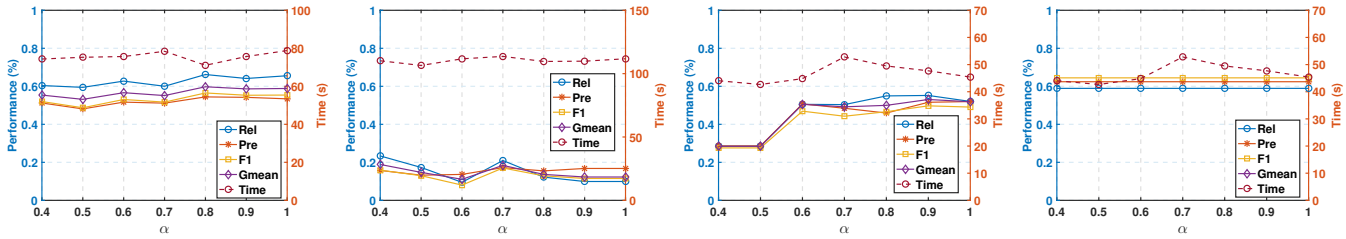
(e) HRS (f) TEP (g) CARLS-S (h) CARLS-M

(i) HRS (j) TEP (k) CARLS-S (l) CARLS-M

Fig. 5: Performance of different thresholds on all datasets within class-incremental setting.

instance, the HRS dataset has 5 categories, while the TEP dataset has 22 categories. Therefore, on the HRS dataset, a uc value of 3 achieves a trade-off in performance across the four metrics and training time, whereas on the TEP dataset, a uc value of 12 achieves a balance.

2) Effects of parameters for pseudo-label samples selection: For the selection of pseudo-label samples, there are three thresholds, positive label threshold τ_p, negative label threshold τ_n, and uncertainty threshold of model prediction k. Figure 5 demonstrates the performance of these three thresholds on



(a) HRS (b) TEP (c) CARLS-S (d) CARLS-M

Fig. 6: Performance of different weights α of pseudo-labeled data on all datasets in class-incremental setting.

different datasets. Regarding the positive label threshold, as shown in Figures 5 (a-d), higher values lead to better model performance due to using pseudo-labels with higher confidence. For the negative label threshold, as shown in Figure 5 (e-h), a value of 0.45 results in the best performance across all datasets. When the threshold exceeds 0.45, the model performance tends to decline, as illustrated in Figure 5 (f) for the TEP dataset. The impact of the uncertainty threshold on model performance is shown in Figures 5 (i-l). When the value is set to 2, the model performs better across all datasets.

3) Effects of pseudo-labeled sample weights α: Figure 6 shows the performance of different pseudo-labeled sample weights α on all datasets within class-incremental setting. From Figure 6, it can be observed that changes in the value of alpha do not significantly impact the model training time. However, there are minor effects on the model's Recall, Precision, F1, and Gmean metrics. The optimal performance is achieved with an $alpha$ value of 0.7 on the HRS, TEP, and CARLS-M datasets, while an $alpha$ value of 0.6 performs best on the CARLS-S dataset.

4) The latency, throughput and memory of `SRTFD`*:* Figure 7 presents the performance of the `SRTFD` system in 100 micro-batches in the TEP dataset with class-incremental scenario, focusing on latency, throughput, and memory usage. Latency remains consistently low across all tasks, demonstrating the system's efficiency in processing real-time data streams. As network communication is omitted from the evaluation, the throughput is calculated as the inverse of latency, maintaining a consistently high level throughout the tasks. This result validates the system's capability for high-speed streaming data processing. Memory consumption increases progressively as new classes are introduced, reaching a peak of approximately 4,000 MB. Notably, memory usage stabilizes after this peak, indicating that the system's memory demand levels off beyond a certain point. This behavior suggests efficient memory management within `SRTFD`, ensuring scalability for long-term deployment.

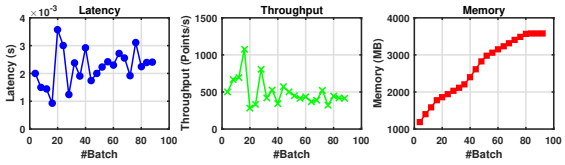


Fig. 7: The latency, throughput and memory of `SRTFD`.

VI. Integrating with HRSCR-HMS

As described in Section II-A, HRSCR-HMS is a fault diagnosis (FD) and health management system designed to monitor cooling rolls in hot rolled steel manufacturing. In its original implementation, fault diagnosis relied on simple, traditional algorithms such as K-Nearest Neighbors (KNN), statistical analysis, and signal processing techniques like exponential smoothing. While these methods were moderately effective, they heavily depended on expert-defined rules tailored to specific operational contexts, requiring significant domain knowledge for fault identification. This reliance on manually crafted rules limited the system's generalizability and necessitated extensive customization for deployment in different factories or under varying operating conditions. Additionally, the original design introduced a fixed latency of at least 5 to 10 minutes for fault diagnosis, making it ill-suited for handling the dynamic, high-velocity demands of modern industrial environments.

To overcome these limitations, we integrated `SRTFD` into HRSCR-HMS, creating an enhanced framework termed HRSCR-HMS-`SRTFD`. This integration replaced the periodic, rule-based fault-checking mechanism with `SRTFD` 's real-time and adaptive fault diagnosis capabilities, transforming the system into a dynamic, data-driven solution. By incorporating advanced techniques such as Retrospect Coreset Selection (RCS) to improve data efficiency, Global Balance Technique (GBT) to address data imbalance, and Confidence and Uncertainty-driven Pseudo-label Learning (CUPL) to enable adaptive model updates with unlabeled data, `SRTFD` eliminates reliance on expert-defined rules and enables the system to dynamically learn from evolving fault patterns. The system architecture was modified to support continuous data ingestion and real-time analysis, replacing traditional batch processing workflows with streaming data pipelines. These enhancements ensured compatibility with existing hardware and data collection systems while enabling the framework to efficiently handle imbalanced, high-velocity, and large-scale data streams.

The deployment of HRSCR-HMS-`SRTFD` at Panzhihua Steel's factory validated its effectiveness in a real-world

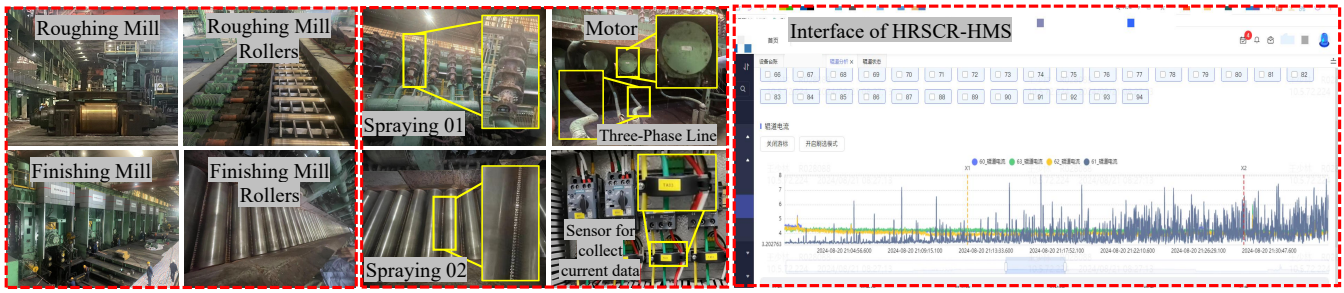


Fig. 8: Deployment of the proposed SRTFD framework integrated into the HRSCR-HMS at Panzhihua Steel's factory. The subfigure (a) shows the key stages in the steel rolling production process, including the roughing and finishing mills, along with the two-stage conveyor rollers. The subfigure (b) illustrates the environment and devices used to collect the data, specifically showing a single roller's spraying systems and motor components, as well as the three-phase line and Hall sensors used to collect current data. The subfigure (c) displays the system interface, where the time series data, which is the current data collected from subfigure (b), is shown. The interface of HRSCR-HMS displays live operational data and analysis results, showcasing the system's ability to provide real-time monitoring and diagnostics.

production environment, as illustrated in Figure 8. Unlike the original HRSCR-HMS, which incurred a latency of up to 10 minutes for fault diagnosis, HRSCR-HMS-SRTFD provided real-time fault detection, completely eliminating this delay. Moreover, it achieved a recall of 68.31% on real-world datasets collected from Panzhihua Steel, showcasing significantly improved fault detection capabilities and responsiveness. The system also removed the reliance on manual rule updates and expert interventions, drastically reducing operational overhead while improving scalability. Its ability to adapt to new and unseen fault patterns further enhanced robustness and flexibility, enabling seamless deployment across diverse industrial scenarios without additional customization. Through the integration of SRTFD, HRSCR-HMS-SRTFD addresses the critical challenges of real-time fault diagnosis in industrial manufacturing. The transformation not only eliminates latency and reduces dependency on domain expertise but also ensures adaptability to evolving fault patterns and industrial demands.

VII. Related Work

a) Fault Diagnosis: Fault diagnosis (FD) ensures reliability, safety, and efficiency in engineering systems by detecting faults that could degrade performance or cause failures [31]. In steel hot rolling, undetected faults lead to defective products, scrap losses, and safety risks. FD focuses on process-level faults requiring real-time detection to maintain operational continuity. However, industrial IoT environments introduce challenges like high-velocity streaming data, evolving fault patterns, severe data imbalance, and limited labeled samples. Traditional FD methods relying on offline models or periodic batch processing cannot meet these demands (see Section I). Scalable, adaptive FD techniques designed for real-time conditions are essential to address these limitations.

b) Online Continual Learning: Online continual learning (OCL) enables models to learn from streaming data without forgetting previous knowledge [32]. Methods include replay-based approaches like experience replay (ER) [21] and ASER [22], regularization techniques such as GEM [23] and AGEM [30], and parameter isolation approaches like progressive neural networks [33] and PackNet [34]. Coreset selection methods like GoodCore [20] and Camel [19] improve scalability by focusing on representative data subsets. Despite their strengths, most OCL methods are general-purpose and not optimized for FD. Challenges such as redundant monitoring data, imbalanced streams, and scarce labels require significant adaptations for effective application in industrial IoT. Tailored OCL solutions are necessary to meet the real-time, domain-specific demands of FD and motivate our design of SRTFD.

VIII. Conclusion

This paper presents SRTFD, a scalable framework designed for real-time fault diagnosis in industrial IoT environments, addressing key challenges such as high training costs, limited labeled samples, severe data imbalance, and the complexities of processing high-velocity data streams. By integrating three innovative components—*Retrospect Coreset Selection (RCS)*, *Global Balance Technique (GBT)*, and *Confidence and Uncertainty-driven Pseudo-label Learning (CUPL)*—SRTFD delivers significant advancements in model performance, scalability, and adaptability. SRTFD addresses the unique demands of dynamic, high-velocity streaming environments by efficiently processing large-scale data streams, mitigating redundancy, and maintaining robust performance in the presence of imbalanced data distributions. Its integration and deployment within HRSCR-HMS validate its practical scalability and industrial relevance, eliminating fault diagnosis latency and achieving a recall improvement of 68.31% in real-world applications. These results highlight SRTFD's ability to enable real-time, adaptive fault diagnosis, ensuring reliability and cost efficiency in modern steel manufacturing and other industrial IoT contexts.

ACKNOWLEDGMENT

This research project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research & Development Programme FCP-SUTD-RG-2022-005. The corresponding author is Shuhao Zhang.

REFERENCES

[1] "中冶赛迪新一代数智热轧工厂——攀钢钒1450热轧智慧产线投用," https://www.cisdi.com.cn/html/mtzx/gsdt/23/03/3412.html, accessed: 2024-11-26.

[2] W. Hao and F. Liu, "Imbalanced data fault diagnosis based on an evolutionary online sequential extreme learning machine," *Symmetry*, vol. 12, no. 8, p. 1204, 2020.

[3] H. Deng, Y. Diao, W. Wu, J. Zhang, M. Ma, and X. Zhong, "A high-speed d-cart online fault diagnosis algorithm for rotor systems," *Applied Intelligence*, vol. 50, pp. 29–41, 2020.

[4] G. Xu, M. Liu, Z. Jiang, W. Shen, and C. Huang, "Online fault diagnosis method based on transfer convolutional neural networks," *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 69, no. 2, pp. 509–520, FEB 2020.

[5] X. Pu and C. Li, "Online semisupervised broad learning system for industrial fault diagnosis," *IEEE transactions on industrial informatics*, vol. 17, no. 10, pp. 6644–6654, 2021.

[6] D. Fedorishin, J. Birgiolas, D. D. Mohan, L. Forte, P. Schneider, S. Setlur, and V. Govindaraju, "Large-scale acoustic automobile fault detection: Diagnosing engines through sound," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2871–2881.

[7] M. Elsisi, M.-Q. Tran, K. Mahmoud, D.-E. A. Mansour, M. Lehtonen, and M. M. Darwish, "Effective iot-based deep learning platform for online fault diagnosis of power transformers against cyberattacks and data uncertainties," *Measurement*, vol. 190, p. 110686, 2022.

[8] Y. Zhou, Y. Dong, and G. Tang, "Time-varying online transfer learning for intelligent bearing fault diagnosis with incomplete unlabeled target data," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 6, pp. 7733–7741, 2022.

[9] L. Jin, Y. Mao, X. Wang, L. Lu, and Z. Wang, "Online data-driven fault diagnosis of dual three-phase pmsm drives considering limited labeled samples," *IEEE Transactions on Industrial Electronics*, 2023.

[10] J. Lin, M. Guo, Q. Hong, and R. Jiang, "An earth fault diagnosis method based on online dynamically calculated thresholds for resonant ground systems," *IEEE Transactions on Smart Grid*, 2024.

[11] X. Yan, M. Sarkar, B. Lartey, B. Gebru, A. Homaifar, A. Karimoddini, and E. Tunstel, "An online learning framework for sensor fault diagnosis analysis in autonomous cars," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[12] P. Han, S. Chen, Z. Liu, and X. He, "Imbalanced real-time fault diagnosis based on minority-prioritized online semi-supervised random vector functional link network," *IEEE Transactions on Instrumentation and Measurement*, 2024.

[13] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[14] C. Nan, F. Khan, and M. T. Iqbal, "Real-time fault diagnosis using knowledge-based expert system," *Process safety and environmental protection*, vol. 86, no. 1, pp. 55–71, 2008.

[15] 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.

[16] P. Wu, S. Lou, X. Zhang, J. He, Y. Liu, and J. Gao, "Data-driven fault diagnosis using deep canonical variate analysis and fisher discriminant analysis," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3324–3334, 2020.

[17] X. Chen, R. Yang, Y. Xue, M. Huang, R. Ferrero, and Z. Wang, "Deep transfer learning for bearing fault diagnosis: A systematic review since 2016," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–21, 2023.

[18] Y. Ghunaim, A. Bibi, K. Alhamoud, M. Alfarra, H. A. Al Kader Hammoud, A. Prabhu, P. H. Torr, and B. Ghanem, "Real-time evaluation in online continual learning: A new hope," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 888–11 897.

[19] Y. Li, Y. Shen, and L. Chen, "Camel: Managing data for efficient stream learning," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1271–1285.

[20] C. Chai, J. Liu, N. Tang, J. Fan, D. Miao, J. Wang, Y. Luo, and G. Li, "Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–27, 2023.

[21] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, 2019.

[22] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Online class-incremental continual learning with adversarial shapley value," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, 2021, pp. 9630–9638.

[23] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[25] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177–1178.

[26] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[27] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah, "In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning," *arXiv preprint arXiv:2101.06329*, 2021.

[28] L. Ma and J. Kaewell, "Fast monte carlo dropout and error correction for radio transmitter classification," in *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2020, pp. 1–5.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[30] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[31] Y. Ren, A. Wang, and H. Wang, "Fault diagnosis and tolerant control for discrete stochastic distribution collaborative control systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 462–471, 2014.

[32] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," *Advances in neural information processing systems*, vol. 32, 2019.

[33] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[34] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.