A Combined Approach of Industrial Edge Computing and Machine Learning for Predictive Maintenance

Abhishek Hazra
ECE department
National University of Singapore
Singapore, 119077
a.hazra@ieee.org

Alakesh Kalita
ECE department
National University of Singapore
Singapore, 119077
alakesh@nus.edu.sg

Mohan Gurusamy
ECE department
National University of Singapore
Singapore, 119077
gmohan@nus.edu.sg

2023 IEEE Globecom Workshops (GC Wkshps) | 979-8-3503-7021-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/GCWKSHPS58843.2023.10464488

***Abstract*—Recent advances in edge computing have enabled latency-critical industrial applications to increase their capabilities by transferring computation data to the network edge, where data processing and analysis occur. However, the major challenge of a smart industry is to minimize downtime and maximize machine lifetime by intelligently estimating predictive maintenance. These demands can be solved by combining the benefits of Industrial Edge Computing (IEC) and Machine Learning (ML) paradigms. Therefore, in this work, we propose an IEC-enabled industrial task offloading and predictive maintenance framework for estimating downtime and future failures of industrial machines. At first, we classify industry-generated tasks into IEC executables and cloud executables using a heuristic technique. Then, tasks are offloaded to suitable computing devices using a queue allocation strategy. The remaining life of industrial machines is also accurately estimated with the help of several ML-based techniques. We carry out performance study with a practical dataset *NASA Turbofan Jet Engine Dataset* collected from 3 different types of industrial machines. Numerical results demonstrate that our proposed strategy is effective in reducing the end-to-end delay and energy consumption rates while estimating scheduled maintenance at the network edge, generating near-optimal solutions.**

***Index Terms*—Industrial Edge Computing, Industrial Internet of Things, Offloading, Predictive Maintenance.**

I. Introduction

In the Industrial Internet of Things (IIoT) era, applications related to product lifetime, predictive maintenance, downtime estimation, etc., are becoming increasingly popular [1]. These applications are authorised to offer safety observation, preemptive disciplinary measures, and improve machines' lifetime. However, owing to limited resource capability and heavy data traffic, industrial devices cannot meet the execution requirements of these applications [2]. To overcome these challenges, industries started incorporating the advantages of Industrial Edge Computing (IEC), which focuses on local processing of latency-critical tasks rather than connecting them to nearby edge devices to reduce latency, increase reliability, and improve security at the edge. [3]. As IEC servers are also restricted in terms of storage and processing capabilities, deploying these applications to IEC servers and delivering the execution requirements of IIoT applications poses several new challenges and dependencies for industrial operations [4]. These challenges include high availability and fault tolerance, data transfer and synchronization optimization, data security, and interoperability. Consequently, it is essential to incorporate task classification, best-match device selection, and efficient task offloading strategy for industrial processes to maintain quicker decision-making capabilities.

Industrial predictive maintenance is an anticipatory approach that uses data collected from sensors and IoT devices. This data is then subjected to advanced analytical techniques and machine learning algorithms to forecast potential equipment malfunctions in manufacturing environments. This methodology allows businesses to strategically plan maintenance activities, mitigating unexpected operational interruptions, minimizing maintenance expenses, and optimizing equipment performance. Industrial predictive maintenance is significant for its capacity to mitigate disruptions, strengthen safety measures, save expenses, and enhance overall operational efficiency. Nevertheless, several problems must be addressed to effectively analyze data in order. These challenges encompass the assurance of data quality, the integration of data from various sources, the ability to handle substantial amounts of data, and the initial financial commitment necessary for implementation. Despite the aforementioned problems, predictive maintenance has emerged as an essential component of contemporary industrial practices. It provides notable benefits in terms of enhanced performance and cost-efficiency.

A. Motivation

A growing number of pharmaceutical companies, industrial manufacturing facilities, oil refineries, and mining operations use heavy machinery to increase productivity, prevent machine failures, and achieve several other benefits [5]. Even though such developments can be advantageous for industries, they pose challenges, such as periodic maintenance spans, increasing device lifetimes, and optimizing operational

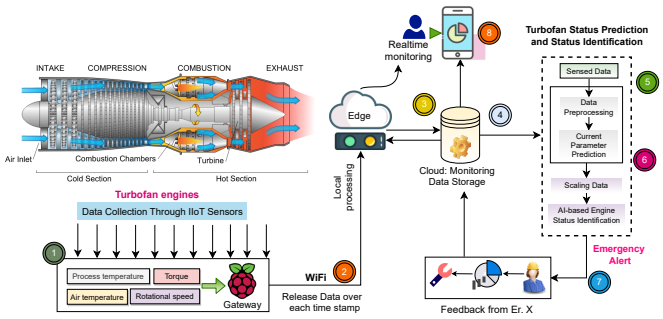

Fig. 1: Illustration of industrial edge networks for predictive maintenance.

costs. These challenges present a borderline opportunity for research about industrial automation processes. Therefore, an IEC framework is necessary for time-critical industrial applications involving the placement of IEC servers to collect data and intelligently identify the useful lifecycle of industrial devices. However, the advantages of ML-based data analytics and estimating the next predictive maintenance cannot be neglected. Further, ML models can be advantageous for predicting analysis, enabling proactive maintenance that can reduce downtime and increase efficiency.

B. Related Work

Edge computing has already indicated outstanding attainment in manufacturing industries for fault detection and predictive maintenance with the help of near-edge computing functionalities [6], [7]. Numerous studies have been accomplished to evaluate machine downtime and handle failure in edge-enabled industrial networks. In this regard, a straightforward method could allocate industry-generated tasks among IEC servers. For example, Teoh *et al.* [8] have presented a fog computing-based innovative predictive maintenance model for industrial applications. De *et al.* [9] have developed an ML-based fault diagnosis and condition monitoring system in edge computing networks. Similarly, Liu *et al.* [10] have proposed an IIoT fault detection system for sustainable Industry 4.0 applications. These approaches improve industrial functionality and the lifespan of industrial machines, but there is still a need for efficient industrial frameworks that can further enhance downtime and reduce future failures.

Only a few research activities focus on industrial fault diagnosis and predictive maintenance capabilities through edge networks. Existing studies focus mainly on predicting fault rates with ML or using edge computing separately [11]. Additionally, few research initiatives examine how data analytics and industrial edge networks can work together to accelerate manufacturing processes. Although edge-enabled industrial networks optimize costs and maintain a sustainable industrial environment, research in this domain requires further advancement. Several ML techniques can be applied to industrial environments to improve decision-making capabilities. Additionally, ML-based strategies can make intelligent decisions when the environment is unknown [12]. This paper bridges research gaps in integrating edge computing, implementing ML-based predictive maintenance, and enabling real-time decision-making in latency-critical industrial edge networks. Fig. 1 illustrates various components and the sequence of activities carried out for predictive maintenance in industrial edge networks.

C. Contribution

To address these issues, we design a unique predictive maintenance framework that utilises ML-enabled data analytics to predict machine downtime with IEC resources. The primary objective of this work is to estimate subsequent downtime and schedule maintenance for complex industrial machines while performing industrial operations. The key contributions of this study are summarized as follows.

- We exemplify our optimization function by integrating weighted latency and energy for industrial task execution and considering several important constraints regarding task offloading. To achieve this goal, we have developed a threshold-based task classification technique to classify tasks into IEC executables and cloud executables.
- To optimize the end-to-end execution delay, we transform our problem into a binary task offloading decision-making problem and develop a queue allocation strategy for obtaining the best possible devices for task execution. Further, to make next-scheduled maintenance and estimate the remaining lifetime, we leverage the capabilities of ML techniques.
- We carry out a comprehensive performance study, and the numerical results show that our proposed framework archives relatively close to optimal solutions while estimating the next maintenance scheduled time while preserving energy and latency as low as possible.
- Our performance study uses a practical dataset "*NASA Turbofan Jet Engine Dataset*" collected from 3 different types of industrial machines.

The rest of the paper is organized as below. The network design and objective function are presented in Section II. Section III describes our IEC-enabled predictive maintenance framework. Performance analysis of the proposed strategy is illustrated in Section IV. Section V concludes our discussion.

II. System Model and Problem Formulation

Considering an industrial edge network scenario with $\mathcal{K} = \{1, 2, \ldots, K\}, \forall k \in \mathcal{K}$ IIoT sensors and a set of $\mathcal{N} = \{1, 2, ..., N\}, \forall n \in \mathcal{N}$ IEC servers, where industrial sensors are connected with one default IEC server. Let $\mathcal{S} = (\mathcal{M} \cup \mathcal{N})$ denotes the set of computing devices, where $\mathcal{M} = \{1, 2, \ldots, M\}, \forall m \in \mathcal{M}$ be the set of cloud servers used to handle excessive data generated from IIoT sensors. In this network, IIoT sensors $\mathcal{K}$ generate $\mathcal{I} = \{1, 2, \ldots, I\}, \forall i \in \mathcal{I}$ number of time-dependent tasks, defined by $3 - tuples$, $\mathcal{I}_k \triangleq \langle \mathcal{I}_i^{\text{IN}}, \mathcal{I}_i^{\text{EB}}, \mathcal{I}_i^{\text{CPU}} \rangle$, where $\mathcal{I}_i^{\text{IN}}$ indicates the input size, $\mathcal{I}_k^{\text{EB}}$ defines the execution bound and $\mathcal{I}_i^{\text{CPU}}$ signifies the CPU

requirement (in cycles). Further, denote $\pi \in \mathbb{R}^{\mathcal{I}\times|\mathcal{M}\cup\mathcal{N}\cup\mathcal{K}|}$ be the task offloading decision profile, where each entry $\pi(i,j) \in \{0,1\}, i \in \mathcal{I}, j \in (\mathcal{M}\cup\mathcal{N}\cup\mathcal{K})$ in the decision vector can be represented by.

$$\pi(i,j)=\begin{cases}1 & \text{if } i\text{th task is offloaded to the } j\text{th device,}\\ 0 & \text{otherwise.}\end{cases}$$

A. IIoT Execution

Initially, IIoT sensors try to execute their tasks locally using available computation frequency $\mathscr{F}_j^{\text{CPU}}$. Given the fixed task execution processing density $\mathcal{I}_i^{\text{PD}}$, the IIoT level execution delay can be expressed as follows.

$$\mathbb{T}_{i,j}^{\text{IIoT}} = \frac{\sum_{i=1}^{\mathcal{I}} \pi(i,j)\mathcal{I}_i^{\text{IN}}\mathcal{I}_i^{\text{PD}}}{\mathscr{F}_j^{\text{CPU}}},\ \forall i \in \mathcal{I}, j \in \mathcal{K}\,. \tag{1}$$

B. Remote Execution

Due to the confined storage and processing capacity of IIoT sensors, most tasks are offloaded to nearby IEC servers or cloud data centres. Let hp_i be the task offloading power gain between ith IIoT sensor jth computing servers, $\forall j \in \mathcal{S}$. Then, the task offloading rate can be defined as $\mathbb{R}_{i,j}^{\text{UP}} = \mathscr{B}_{i,j}\log_2\left(1+\frac{\mathbb{P}_j^{\text{UP}}hp_i}{\xi_j^2}\right), \forall i \in \mathcal{I}, j \in \mathcal{S}$, where, $\mathbb{P}_j^{\text{UP}}$ signifies the transmission power and $\mathscr{B}_{i,j}$ denotes the transmission bandwidth between ith IIoT sensors and jth computing server. Thus the task offloading delay from ith IIoT sensor can be defined as follows.

$$\mathbb{T}_{i,j}^{\text{UP}} = \frac{\sum_{j=1}^{\mathcal{S}} \pi(i,j)\mathcal{I}_i^{\text{IN}}}{\mathbb{R}_{i,j}^{\text{UP}}}, \forall i \in \mathcal{I}, j \in \mathcal{S} \tag{2}$$

Once the tasks are offloaded from industrial devices, computing devices process tasks immediately, and task execution delay on various computing devices can be defined as follows.

$$\mathbb{T}_{i,j}^{\text{EXE}} = \frac{\sum_{j=1}^{\mathcal{S}} \pi(i,j)\mathcal{I}_i^{\text{IN}}\mathcal{I}_i^{\text{PD}}}{\mathscr{F}_j^{\text{CPU}}},\ \forall i \in \mathcal{I}, j \in \mathcal{S} \tag{3}$$

Similarly, the result-downloading time to the ith computing device can be expressed as.

$$\mathbb{T}_{j,i}^{\text{DOWN}} = \frac{\sum_{j=1}^{\mathcal{S}} \pi(j,i)\mathcal{I}_i^{\text{IN}}}{\mathbb{R}_{j,i}^{\text{DOWN}}} \tag{4}$$

Thus, the overall delay in executing ith task on jth executing device can be defined as follows.

$$\mathbb{T}_{i,j}^{\text{Total}} = \begin{cases}\mathbb{T}_{i,j}^{\text{IIoT}} & \forall i \in \mathcal{I}, j \in \mathcal{K}\\ \mathbb{T}_{i,j}^{\text{UP}} + \mathbb{T}_{i,j}^{\text{EXE}} + \mathbb{T}_{j,i}^{\text{DOWN}} & \forall i \in \mathcal{I}, j \in \mathcal{S}\end{cases} \tag{5}$$

Similarly, by adding $\mathbb{P}^{\text{IIoT}}$ and $\mathbb{P}^{\text{Offload}}$ amount of power consumption on various computing devices, we can also calculate the overall energy consumption rate as.

$$\mathbb{E}_{i,j}^{\text{Total}} = \begin{cases}\mathbb{E}_{i,j}^{\text{IIoT}} & \forall i \in \mathcal{I}, j \in \mathcal{K}\\ \mathbb{E}_{i,j}^{\text{UP}} + \mathbb{E}_{i,j}^{\text{EXE}} + \mathbb{E}_{j,i}^{\text{DOWN}} & \forall i \in \mathcal{I}, j \in \mathcal{S}\end{cases} \tag{6}$$

C. Problem Formulation

A task can be executed on industrial machines or remote computing servers based on task offloading decision profile $\pi(i,j)$. Similarly, we can define our objective function as the weighted energy-delay optimization problem for all industry-generated tasks in IEC networks.

$$\text{minimize} \quad \sum_{i=1}^{\mathcal{I}}\left(\mathbb{E}_{i,j}^{\text{Total}} + \delta\mathbb{T}_{i,j}^{\text{Total}}\right) \tag{7a}$$

$$\text{subject to} \quad \sum_{i=1}^{|\mathcal{I}|}\sum_{j=1}^{|\mathcal{S}|}\pi(i,j) \le |\mathcal{S}|, \tag{7b}$$

$$\sum_{i=1}^{|\mathcal{I}|}\pi(i,j) = 1, \tag{7c}$$

$$\pi(i,j) \in \{0,1\}, \tag{7d}$$

$$\mathbb{T}_{i,j}^{\text{UP}} \ge 0 \text{ and } \mathbb{T}_{j,i}^{\text{DOWN}} \ge 0, \tag{7e}$$

Where the weight factor δ controls the energy-delay tradeoff for task execution, *i.e.,* when $\delta = 0$, we only examine energy consumption rate rather than delay. The case of $\delta = 1$, means that energy and delay are considered with equal weightage. In the above formulation, constraints (7b), (7c) and (7d) represent the binary task offloading decision profile on various computing devices. Additionally, a positive uplink and downlink transmission rate is ensured by the constraint (7e).

III. Proposed Methodology

The proposed predictive maintenance framework is separated into multiple phases. At first, a heuristic-based task classification strategy is considered to identify the delay-sensitive and computation-intensive. In the next phase, we adopt a task queueing strategy to offload industrial tasks on appropriate devices. Finally, ML techniques are used to make accurate estimations of the remaining lifetime of industrial machines. A summary of these techniques is presented below.

A. Task Classification

The task classification strategy mainly identifies the delay-sensitive and computation-intensive tasks using input size ($\mathcal{I}_i^{\text{IN}}$) and execution deadline ($\mathcal{I}_i^{\text{EB}}$) of each task. Besides, this strategy assumes two threshold parameters for defining priority, namely input bound (ϕ^{CPU}) and execution bound (ψ^{EB}), respectively. Then, based on threshold bounds, industrial machines classify tasks as follows.

- **Delay-sensitive Tasks ($\mathcal{I}^D$):** *A set of tasks $\mathcal{I}$ in the industrial network is called Delay-sensitive $\mathcal{I}^D$, if $\mathcal{I}_i^{IN} < \psi^{IN}$ and $\mathcal{I}_i^{EB} < \phi^{EB}$, $\forall i \in \mathcal{I}$. This represents that these tasks are more suitable for IEC sever execution.*
- **Computation-intensive Tasks ($\mathcal{I}^C$):** *A set of tasks $\mathcal{I}$ in the industrial network is called Computation-intensive $\mathcal{I}^C$, if $\mathcal{I}_i^{IN} \ge \psi^{IN}$ or $\mathcal{I}_i^{EB} \ge \phi^{EB}$, or both $\forall i \in \mathcal{I}$. This represents that these tasks are more suitable for cloud server execution.*

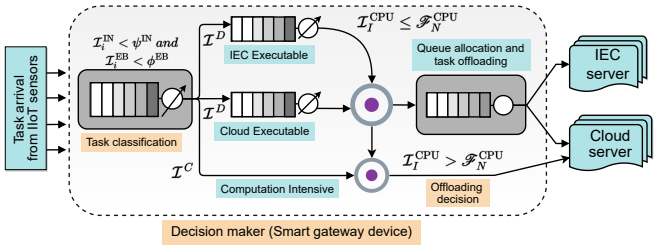

Fig. 2: Illustration of task offloading and queue allocation strategy.

After task classification, industrial machines offload delay-sensitive tasks to the nearby default IEC server and computation-intensive tasks to the cloud server. In the next section, we describe how IEC servers further offload excessive workloads to various cloud servers to control task execution delays and energy consumption rates.

B. Queue Allocation and Task Offloading

Once tasks are received, IEC servers maintain two separate queues: IEC executable (Q^D) and cloud executable (Q^C) for handling computation demands. Tasks that meet the available CPU requirements are kept in the IEC executable queue in IEC servers. The remaining tasks are moved to the cloud executable queue for further offloading to the cloud server. To determine the allocation of queues, the following rules must be followed.

- **IEC Executable:** *A set of tasks is kept in IEC executable queue if and only if the IEC server has the computational capacity to execute those tasks, i.e.,* $Q^D = \{\mathcal{I}_i \in I \mid \mathcal{I}_i^{CPU} \in \left(\mathcal{I}_1^{CPU}, \mathcal{I}_2^{CPU}, \ldots, \mathcal{I}_I^{CPU}\right) \cap \mathscr{F}_j^{CPU} \in \left(\mathscr{F}_1^{CPU}, \mathscr{F}_2^{CPU}, \ldots, \mathscr{F}_N^{CPU}\right) \cap (\mathcal{I}_I^{CPU} \leq \mathscr{F}_N^{CPU})\}$.
- **Cloud Executable:** *A set of tasks are kept in the cloud executable queue if IEC servers don't have sufficient computation capacity during execution. i.e.,* $Q^C = \{\mathcal{I}_i \in I \mid \mathcal{I}_i^{CPU} \in \left(\mathcal{I}_1^{CPU}, \mathcal{I}_2^{CPU}, \ldots, \mathcal{I}_I^{CPU}\right) \cap \mathscr{F}_j^{CPU} \in \left(\mathscr{F}_1^{CPU}, \mathscr{F}_2^{CPU}, \ldots, \mathscr{F}_N^{CPU}\right) \cap (\mathcal{I}_I^{CPU} > \mathscr{F}_N^{CPU})\}$.

After queue allocation, IEC servers execute tasks and return results to the requesting industrial machines. On the other hand, cloud executable tasks are offloaded to the cloud server to meet execution requirements. This strategy helps to redistribute tasks with lower computational complexity. The queue allocation and task offloading strategy are depicted in Fig. 2.

C. ML-Based Downtime Estimation

Finally, computing devices start using supervised ML-based techniques (such as random forest, bagging, boosting, and ensemble technique) to estimate the next maintenance schedule accurately [13]. For predictive analysis, supervised ML involves training a model on labelled datasets, where each data point represents an expected outcome. This enables the algorithm to learn patterns and relationships between input features and target variables, making predictions about industrial data and estimating the next maintenance schedule. Moreover, supervised ML algorithms can be trained on a variety of different types of data, such as unstructured data (e.g. images or text), structured data (e.g. from a database), and time-series data (e.g. sensor readings or stock prices). With its versatile nature, supervised ML is an effective tool for analyzing industrial data [14]. The detailed steps are also presented in the *Algorithm* 1. It can be observed from *Algorithm* 1 that our heuristic strategy takes $\mathcal{O}(\mathcal{K})$ time to classify tasks on the industrial network. On the other hand, the queue allocation strategy takes $\mathcal{O}(\mathcal{KS})$ time to allocate tasks to multiple queues and simultaneously offload them to randomly selected suitable computing devices. Thus, the overall time complexity of our proposed methodology is $\mathcal{O}(\mathcal{K}) + \mathcal{O}(\mathcal{KS})$.

Algorithm 1 Task Offloading

INPUT: $\mathcal{I}$, $\mathcal{K}$, $\mathcal{N}$, $\mathcal{M}$, $\mathcal{I}^{\text{IN}}$, $\mathcal{I}^{\text{CPU}}$, ϕ^{EB}, ψ^{IN}, $\mathcal{I}^{\text{EB}}$
OUTPUT: $\pi(i, j)$
for $i \leftarrow 1$ to $\mathcal{I}$ **do**
  if $\mathcal{I}_i^{\text{IN}} < \psi^{\text{IN}}$ and $\mathcal{I}_i^{\text{EB}} < \phi^{\text{EB}}$ **then**
    if $\mathcal{I}_I^{\text{CPU}} \leq \mathscr{F}_N^{\text{CPU}}$ **then**
      Execute $\mathcal{I}^D$ tasks on IEC server
    end if
    if $\mathcal{I}_I^{\text{CPU}} > \mathscr{F}_N^{\text{CPU}}$ **then**
      Offload $\mathcal{I}^D$ tasks to cloud server servers
    end if
    Offload $\mathcal{I}^C$ tasks to cloud server servers
  end if
  Return offloading decision $\pi(i, j)$
  Use ML for downtime estimation
end for

IV. Numerical Analysis

In this section, we examine the implementation of our task offloading and downtime prediction strategy by evaluating different performance matrices, such as a) end-to-end execution delay, b) energy consumption and c) downtime estimation. For further demonstration, we compare our method with three standard techniques, namely `Wang's` strategy, `Mao's` strategy, and `Din's` strategy, as defined in [15].

A. Simulation Setup

For the numerical analysis, we assume $\mathcal{I} = \{50, 100, 150, 200, 250, 300\}$, $\mathcal{K} = 20$, $\mathcal{N} = 10$, $\mathcal{M} = 2$ and $\mathscr{B}_{i,j} = 20$MBps. We set $\phi^{\text{CPU}} = 15$, $\psi^{\text{EB}} = 15$, $\mathbb{P}_n^{\text{UP}} = 0.5$mW $\mathbb{P}_m^{\text{UP}} = 10$mW and $\delta = 0.5$. Throughout the experiment, we consider $\mathscr{F}_i^{\text{CPU}} << \mathscr{F}_n^{\text{CPU}}$ and $\mathscr{F}_n^{\text{CPU}} << \mathscr{F}_m^{\text{CPU}}$ for performing ML-based techniques on computing devices. In this work, tasks are generated through a random process for making offloading decisions. Additionally, we obtained NASA Turbofan Jet Engine Dataset[1] for estimating the best possible lifetime and next approximate maintenance of industrial machines. The complete experiment is done using `sklearn` and `imblearn` on the Google Colab environment.

[1]https://www.kaggle.com/datasets/behrad3d/nasa-cmaps

TABLE I: Machine types and their failure rates

Machine	Failure	No Failure
Low type	96.14%	3.86%
Medium type	97.36%	2.64%
High type	98.00%	2.00%

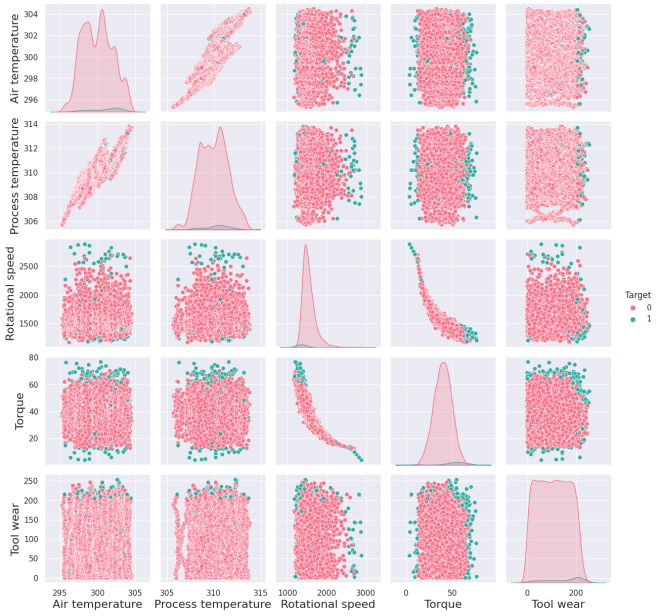

Fig. 3: Correlation matrix between industrial attributes.

B. Dataset Description

The NASA Turbofan Jet Engine Dataset is collected from 3 different types of industrial machines and contains 10000 entries. Among the industrial machines, 60% of the engines are large, 30% of the engines are medium, and 10% of the engines are small in size. The target results also contain 96.7% of no failure and 3.3% of failure labels. Statistical observation also reveals that 1.1% and 1.0% of failures are the reasons for power and toolware. Additionally, 0.8% of failure comes from overstrain failure, and 0.5% of failure is due to high heat dissipation failure. A summary of machine types and failure rates is also illustrated in Table I. The dataset has six key features (such as type, air temperature [K], process temperature [K], rotational speed [rpm], torque [Nm], and tool wear [min]) and two output levels (failure and no failure). We have also evaluated the types of failure among the industrial machines. A correlation of various failure types is shown in Fig. 3.

C. Performance Analysis

To analyze the performance of the proposed framework, we consider various metrics while varying the number of incoming tasks $\mathcal{I}$ upto 300. Fig. 4(a) shows the execution rate (in %) at which tasks are completed their execution on various computing devices $\mathcal{S}$. Based on Fig. 4(a), it is easy to observe that with increasing task numbers $\mathcal{I}$, the execution rate for IEC servers $\mathcal{N}$ gradually decreases. However, the execution rate on cloud servers $\mathcal{M}$ increases. The reason is that IEC servers $\mathcal{M}$ are limited in terms of storage and processing capabilities. IEC servers $\mathcal{N}$ can only execute a certain number of tasks $\mathcal{I}$, and beyond that level, IEC servers $\mathcal{N}$ fail to execute those tasks and start transferring them to the cloud servers. As the cloud server has an extensive processing capability, it can complete the remaining unsatisfied talks coming from industrial sensors, thus balancing the task execution rate on different devices.

If we look closely, we can observe that the task completion rate directly affects how long different computing devices take and how much energy they use. As illustrated in Fig 4(b), the delay on IEC and cloud servers varies with the number of tasks $\mathcal{I}$. Additionally, Fig. 4(b) indicates that IEC servers $\mathcal{M}$ have the potential to regulate the overall end-to-end execution delay $\mathbb{T}_{i,j}^{\text{Total}}$ within a certain limit. However, as the number of tasks increases, our proposed offloading strategy involves transferring computation-intensive tasks to the cloud server. Cloud server execution encompasses transmission delay $\mathbb{T}_{i,j}^{\text{UP}}$, processing delay $\mathbb{T}_{i,j}^{\text{EXE}}$, and end-result downloading delays $\mathbb{T}_{j,i}^{\text{DOWN}}$, leading to an increase in overall execution delay.

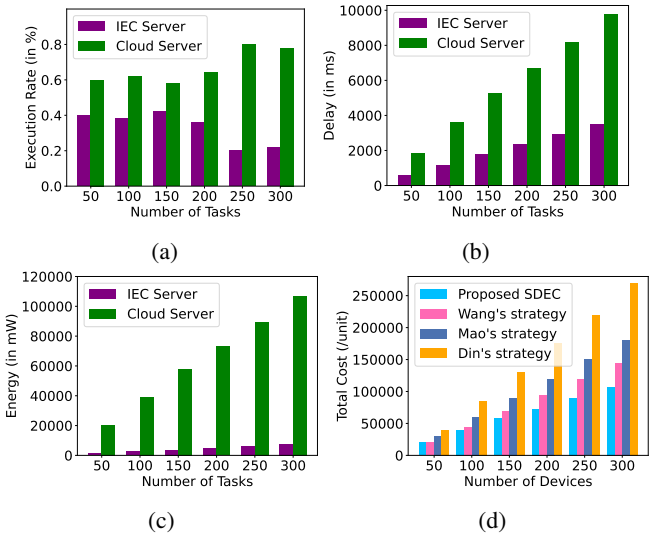

Fig. 4: Performance Analysis in terms of (a) Task execution rate, (b) End-to-end execution delay, (c) Energy consumption rate, (d) Overall system cost.

Similarly, energy consumption rates on various computing devices follow a similar trend as illustrated in Fig. 4(c). IIoT sensors initially try to execute tasks $\mathcal{I}$ on devices, but due to limited storage and processing capabilities, they offload tasks to other computing devices $\mathcal{S}$. As a result of our proposed strategy, delay-sensitive tasks $\mathcal{I}^D$ are offloaded to the IEC servers $\mathcal{N}$ rather than transferred to the cloud servers $\mathcal{M}$. In contrast, computation-intensive tasks $\mathcal{I}^C$ are typically assigned to cloud servers $\mathcal{M}$ for long-term storage and processing. Therefore, tasks sent to cloud-based servers require more storage and CPU resources, leading to higher energy consumption. Besides transferring a significant number of tasks from IEC servers $\mathcal{N}$ to cloud servers $\mathcal{M}$, the proposed strategy reduces computation overhead from IEC servers $\mathcal{N}$, enabling delay-sensitive tasks to be performed on

IEC devices. Thus, the overall energy consumption rate on cloud servers also increases with increased tasks.

Fig. 4(d) illustrates the total computation cost of IIoT, IEC, and Cloud executions of our proposed strategy, along with a comparative analysis of existing strategies (`Wang's`, `Mao's`, and `Din's`). Our proposed strategy significantly improves performance with a comparatively lower overall computation cost. The analysis considers end-to-end costs and the execution of costs on various computing devices. As the number of IIoT sensors increases, the overall computation cost increases due to the data transmission from IIoT sensors to remote computing devices. However, this cost is less than existing techniques.

D. Predictive Analysis and Downtime Estimation

Intelligent estimation for predictive maintenance was implemented using ML techniques on various computing devices to increase machine longevity and reduce downtime. An optimal outcome and device estimation were achieved using balanced models such as random forests, bagging, boosting, and ensemble learning techniques. A total of 1000 iterations were considered to train the model to achieve a balanced prediction. The dataset is divided into training samples (80%) and test samples (20%) in order to test how well the model works. Additionally, we have shuffled our dataset in such a way that both the training samples and the testing samples came from the same distribution. Random forest, bagging, boosting, and ensemble techniques produced results with an accuracy of 97.8%, 97.33%, 86.7%, and 96.15%, respectively. Various standard scaling and sampling techniques were employed to enhance estimation accuracy further. Our cross-validation experiments have shown that the bagging technique is able to achieve a high level of performance improvement compared to other ML algorithms. Although these techniques improved the aggregate prediction accuracy, the average prediction accuracy decreased from 83% to 87%. Various hyperparameters were learned from the training and test samples, which resulted in a 94% training accuracy and 96% test accuracy increase in model performance. Nevertheless, this accuracy can be further improved by incorporating distributed learning techniques.

V. Conclusion

This work developed an efficient downtime prediction strategy for industrial machines in edge networks. To obtain this goal, we developed a problem formulation with the objective function as a sum of weighted energy-delay minimization problems under limited delay constraints. At first, a low-complexity heuristic task classification strategy is considered to separate IEC executable tasks from IIoT executable tasks. Further, to control execution delay and offload tasks to suitable computing devices, we proposed a queue allocation strategy where IEC servers collect real-time industrial data and estimate the subsequent maintenance time using ML techniques. Experimental analysis with a real dataset indicates that our proposed approach outperforms the existing algorithms while executing industrial tasks at the edge. Our future work will incorporate distributed ML techniques into the existing predictive maintenance framework to improve sustainability in industrial networks.

Acknowledgement

This work is supported in part by the National University of Singapore under Grant No.: MoE AcRF Tier-1 FRC Grant, NUS WBS No. A-0005142-01-00.

References

[1] S. Zeb, A. Mahmood, S. A. Khowaja, K. Dev, S. A. Hassan, N. M. F. Qureshi, M. Gidlund, and P. Bellavista, "Industry 5.0 is coming: A survey on intelligent nextg wireless networks as technological enablers," *arXiv preprint arXiv:2205.09084*, 2022.

[2] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Fog Computing for Energy-Efficient Data Offloading of IoT Applications in Industrial Sensor Networks," *IEEE Sensors Journal*, vol. 22, no. 9, pp. 8663–8671, 2022.

[3] H. R. Chi, C. K. Wu, N.-F. Huang, K. F. Tsang, and A. Radwan, "A survey of network automation for industrial internet-of-things towards industry 5.0," *IEEE Transactions on Industrial Informatics*, 2022.

[4] A. Hazra, "Promising Role of Visual IoT: Challenges and Future Research Directions," *IEEE Engineering Management Review*, pp. 1–7, 2023.

[5] A. Raja Santhi and P. Muthuswamy, "Industry 5.0 or industry 4.0 s? introduction to industry 4.0 and a peek into the prospective industry 5.0 technologies," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pp. 1–33, 2023.

[6] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[7] A. Hazra, M. Adhikari, and T. Amgoth, "Dynamic Service Deployment Strategy using Reinforcement Learning in Edge Networks," in *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, 2022, pp. 1–6.

[8] Y. K. Teoh, S. S. Gill, and A. K. Parlikad, "IoT and Fog-Computing-Based Predictive Maintenance Model for Effective Asset Management in Industry 4.0 Using Machine Learning," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2087–2094, 2023.

[9] J. De las Morenas and F. Moya-Fernández, "Predictive Maintenance in Electrical Machines: An Edge Computing Approach," in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2022, pp. 01–06.

[10] Y. Liu, W. Yu, T. Dillon, W. Rahayu, and M. Li, "Empowering IoT Predictive Maintenance Solutions With AI: A Distributed System for Manufacturing Plant-Wide Monitoring," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1345–1354, 2022.

[11] A. Adel, "Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1–15, 2022.

[12] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A survey of predictive maintenance: Systems, purposes and approaches," *arXiv preprint arXiv:1912.07383*, 2019.

[13] S. Schwendemann, Z. Amjad, and A. Sikora, "A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines," *Computers in Industry*, vol. 125, p. 103380, 2021.

[14] A. Bousdekis, K. Lepenioti, D. Apostolou, and G. Mentzas, "Decision making in predictive maintenance: literature review and research agenda for industry 4.0," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 607–612, 2019.

[15] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustainable Computing: Informatics and Systems*, vol. 21, pp. 154–164, 2019.