AUT Journal of Electrical Engineering
10.22060/EEJ.2026.24815.5769

A Deep Reinforcement Learning Approach for Predictive Maintenance in Edge-Enabled Sensor Systems

Ladan Zahmatkeshan[1], Alireza Javadi Noshabadi [1, *], Aliakbar Abdollahzadeh[2], Sajjad Talesh Hosseini [3]

[1] Department of Mining Engineering, University of Kashan, Kashan, Iran
[2] Department of Mining Engineering, Amirkabir University of Technology, Tehran, Iran
[3] Department of Mining Engineering, Faculty of Engineering, Imam Khomeini International University, Qazvin, Iran
* Correspondence: Alireza.javadi@kashanu.ac.ir

Abstract Unexpected failures in essential industrial systems can cause operational disruptions and financial losses. To mitigate unplanned downtime and maintain safe, efficient functioning of critical assets, predictive maintenance strategies are essential. However, with the rapid increase in sensor-equipped machinery, the overwhelming volume of generated data has outpaced the capabilities of traditional machine learning models to provide accurate, real-time diagnostics. This research introduces a model-free deep reinforcement learning (DRL) approach tailored for predictive maintenance within sensor-integrated equipment networks. Each machine is equipped with a sensor module that captures real-time data and detects anomalies. Unlike conventional opaque regression-based methods, the proposed framework autonomously determines optimal maintenance policies and delivers actionable insights for each individual device. Experimental evaluations indicate the potential of this adaptive learning method to extend across diverse maintenance scenarios.

Keywords:

Deep Reinforcement Learning, Predictive Maintenance, Edge Computing, Sensor Networks, Model-Free Learning

AUT Journal of Electrical Engineering
10.22060/EEJ.2026.24815.5769

1. Introduction

In industrial settings, any pause in the production process results in what is commonly termed as equipment downtime, with current industry standards typically aiming to keep it below ≤10% [1]. Despite advancements in technology and increasingly sophisticated machinery, breakdowns still occur frequently and often incur substantial financial losses. Maintenance teams, operating under limited budgets, face mounting pressure to ensure continuous production. When unexpected failures arise, corrective maintenance is employed to quickly repair faulty systems and meet delivery schedules. Alternatively, proactive companies implement preventive maintenance strategies—scheduling regular upkeep to reduce the likelihood of unplanned interruptions. While this approach can slightly improve productivity, it also comes with the trade-off of higher maintenance expenditures, even though the overarching goal is to reduce long-term maintenance costs. Predictive maintenance, which relies on real-time assessment of equipment conditions, is widely regarded as a more cost-effective and labor-efficient strategy. By intervening only, when necessary, it helps minimize downtime and optimizes resource usage.

As industries around the world embrace the principles of Industry 4.0 to boost manufacturing efficiency, modern machinery has grown increasingly complex and demands more sophisticated maintenance strategies. A major challenge in this productivity-centric and labor-constrained environment is simplifying the interpretation of sensor data for predictive maintenance purposes. Traditional black-box regression approaches often rely heavily on hand-crafted features tailored to specific domains, making them costly and inflexible to adapt or scale to related systems. In response, recent research has explored deep learning (DL) as an alternative for predicting remaining useful life (RUL) in various components, including general equipment [2]– [4] and ball bearings [5], [6]. Additionally, studies like [7] have applied Temporal Difference (TD) learning to derive health indicators for estimating the RUL of turbofan engines. However, these techniques primarily focus on accurate RUL predictions and fail to provide actionable insights to guide maintenance teams. A common limitation of such methods is their

10.22060/EEJ.2026.24815.5769

reliance on explicit models, which are often too complex or poorly understood to represent real-world systems effectively. To address this, our study proposes a model-free Deep Reinforcement Learning (DRL) approach that autonomously derives optimal maintenance policies based on equipment health status, while also delivering practical, interpretable recommendations.

Rising consumer demands, tighter profit margins, and the decreasing cost of cloud infrastructure are key motivators pushing manufacturers to modernize in order to maintain a competitive edge. However, companies in the manufacturing and industrial equipment sectors remain hesitant to transmit sensitive operational data to the cloud, especially over unreliable internet connections. As a result, full-scale cloud adoption tends to be limited to larger enterprises with significant resources [8]. An alternative approach involves leveraging locally connected sensor networks. In the proposed system architecture, each machine is equipped with edge computing devices integrated with sensors [9], enabling peer-to-peer data exchange among edge units. This decentralized setup offers several benefits, including accelerated decision-making, more responsive and data-driven maintenance support, and reduced data load on the manufacturing facility's IT infrastructure.

The proposed learning framework builds upon the Double Deep Q-Network (Double DQN) algorithm and employs prioritized experience replay to improve sample efficiency and training stability [10, 11].

Recent advances in deep reinforcement learning have significantly improved the stability and efficiency of value-based methods. One notable contribution is the Double Deep Q-Network (Double DQN), originally introduced by van Hasselt et al. [10], which addresses the overestimation bias inherent in standard DQN by decoupling action selection from action evaluation. In parallel, Li et al. [11] proposed the Prioritized Experience Replay mechanism, which enhances learning efficiency by sampling more informative transitions based on their temporal-difference error. These foundational techniques have since become standard components in modern DRL architectures and have been widely adopted in sequential decision-making problems, including maintenance optimization, resource allocation, and industrial control systems.

10.22060/EEJ.2026.24815.5769

The core contribution of this work is the development of a model-free DRL-based maintenance decision-making framework that directly derives optimal maintenance actions from multi-sensor data collected at the edge. Unlike conventional RUL estimation methods, the proposed PDDQN-PN algorithm is designed to learn actionable maintenance policies under sparse reward conditions, enhanced through prioritized experience replay and parameter noise. This allows the system to provide practical, interpretable, and equipment-specific maintenance recommendations. Unlike prior DRL-based predictive maintenance studies, our work introduces a cost-aware, model-free action-selection framework (repair/replace/hold) optimized for edge/fog architectures. The proposed PDDQN-PN method integrates prioritized replay with parameter noise to manage sparse rewards and accelerate convergence, while an unsupervised health indicator enables operation without ground-truth labels. These design choices distinguish our contribution from existing literature on DRL-based sensor analytics and edge-predictive maintenance.

In summary, this research presents the following key contributions:

1. A novel problem formulation aimed at maximizing equipment operational time by leveraging multiple sensor inputs. The proposed solution employs a model-free Deep Reinforcement Learning (DRL) framework to derive optimal decision-making policies.

2. The DRL approach incorporates an Equipment Health Indicator to deliver actionable guidance on replacement strategies, offering transparent, data-informed recommendations.

3. The developed DRL method effectively tackles the challenge of sparse reward signals in maintenance scenarios, maintaining robust and consistent policy performance across various datasets and equipment types—even when faced with uncertain initial conditions and the absence of labeled ground truth data.

2. System Model

This section explores the deployment of a sensor network for real-time health monitoring of industrial equipment, either through direct installation or retrofitting. At the device level, the proposed architecture

10.22060/EEJ.2026.24815.5769

adopts a Star Topology configuration (see Figure 1), where individual equipment units are connected to a centralized node. For advanced data processing and broader system integration, collected sensor data can be further transmitted to a higher-level sensor infrastructure organized using mesh or tree topology schemes. The core elements of such a network include Sensor Nodes (SNs) and a Sensor Gateway (SG), which is also commonly referred to as the Base Station. It is important to note that the terms "Sensor Gateway" and "Base Station" are used interchangeably throughout this work. In the proposed architecture, communication between Sensor Nodes (SNs) and the Sensor Gateway (SG) is established through a wireless link using low-power industrial wireless communication protocols. This design enables flexible deployment and avoids the need for wired cabling in distributed equipment environments.

Each Sensor Node (SN) functions as an autonomous measurement unit and does not communicate directly with other SNs. Instead, interaction occurs indirectly through the Sensor Gateway (SG), which aggregates multi-sensor data and coordinates the maintenance decision-making process. This design combines local autonomy with system-level cooperation.

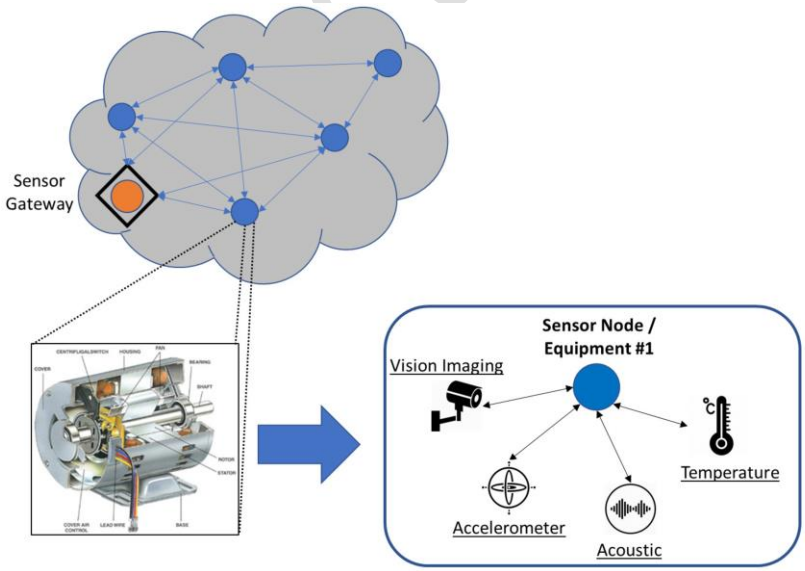


Fig. 1. Equipment-level architecture of the sensor network system.

10.22060/EEJ.2026.24815.5769

As illustrated in Figure 1, Sensor Node (SN) devices represent physical sensors responsible for transmitting various types of data, such as temperature, acceleration, angular velocity (gyroscope), and acoustic signals. Continuous transmission of raw sensor streams is known to consume significant network bandwidth, potentially degrading overall network performance. To mitigate network congestion, the model incorporates data aggregation and initial processing at the Sensor Gateway (SG) node. It is assumed that each SG possesses sufficient computational and storage resources to handle this workload. For efficiency in both memory usage and processing time, the system performs sensor data analysis using a batch-processing approach, based on fixed time intervals. For simplicity of explanation—and without loss of generality—the model assumes a one-to-one pairing between SG and SN in the system architecture.

Based on the analysis of sensor data over a time window comprising K timesteps, the system predicts one of three potential maintenance actions: repair, replace, or hold. When the Sensor Gateway (SG) assesses a low probability of imminent failure, the default recommendation is to hold, indicating no immediate intervention is necessary.

In cases where the SG detects abnormal sensor readings with a high degree of certainty, it must decide between recommending either a repair or a complete replacement—mirroring real-world maintenance decision-making. However, due to time constraints and limited maintenance budgets, repair is typically the preferred and most feasible action in practice.

The proposed architecture follows the fog computing paradigm, where Sensor Nodes operate at the edge as simple data-gathering units, and the Sensor Gateway functions as a fog node performing local computation, health estimation, and DRL-based decision-making. This design reduces latency and bandwidth usage while maintaining compatibility with optional cloud services for long-term analytics.

While the proposed edge-enabled maintenance framework reduces reliance on continuous cloud connectivity for real-time decision-making, it does not eliminate the role of cloud data centers. Cloud resources may still be utilized for long-term data storage, large-scale analytics, and periodic model

10.22060/EEJ.2026.24815.5769

retraining. The aim of the proposed method is to shift time-critical inference tasks to the edge rather than replacing cloud infrastructures.

3. Problem Formulation

The primary objective of the proposed system model is to maximize the cumulative operational time of equipment while remaining within defined maintenance budget limitations. The optimization target, referred to as Maximum Equipment Uptime (ρ), is mathematically described in Eq. (1) and is framed within the context of a Markov Decision Process (MDP), assuming full observability of system states.

An MDP is formally characterized by a 5-tuple:

- S : the set of states,
- A : the set of possible actions,
- R : the reward function,
- γ : the discount factor, and
- P : the state transition probability distribution.

This formulation provides a structured framework for learning optimal decision policies through interaction with the environment. The 5-tuple can be succinctly represented as (S, A, R, γ, P).

$$\rho = \sum_{Node=1}^{N} RunTime \tag{1}$$

The raw sensor data generated within the sensor network is denoted as X_t^i , where $i \in \{0,1,...,Z\}$ represents the sensor index at each discrete time step t. As outlined in Eq. (1), the system considers N Sensor Nodes (SN) in total, with this study assuming $N = 1$ for simplicity. Eq. (2) introduces a discretization step to reduce the complexity of the continuous sensor readings, facilitating more efficient

analysis. Following this, the state space corresponding to each individual sensor node is expanded in Eq. (3), allowing the system to capture a broader representation of the equipment's operational context.

$$q_x \leftarrow Discretize\,(X_t^i) \tag{2}$$

$$S^i = \{q_z\} \tag{3}$$

Manufacturers typically specify the Mean Time Between Failures (MTBF) and recommended operating temperature range for each device or sensor within its technical documentation. However, due to fluctuating environmental temperatures, equipment degradation and condition changes often follow a non-linear and non-sequential pattern. In this study, we initially model the sensor state transitions as a sequential process exhibiting a gradual degradation trend across discrete timesteps. For example, the probability of transitioning between states is assumed to increase with rising operational temperature. Over time, as described in Eq. (4), a mode shift in temperature behavior becomes evident—leading to an exponential decrease in the rate of deterioration as equipment adapts or stabilizes in the new thermal environment. This adaptation is reflected by the sensor skipping intermediate states to represent equivalent shifts in temperature under the new mode. Empirical observations indicate that sensor state transitions relative to equipment runtime follow a concave exponential decay curve.

Figure 3 visualizes the generation of a pseudo health indicator, which adjusts dynamically as the failure risk increases. This behavior is modeled by considering the transition probability of each state-action pair. Importantly, the observed pattern of degradation aligns with the system model and exhibits an inverse correlation between equipment health and increasing sensor state values.

$$F(i) = e^{-\lambda t} \tag{4}$$

As illustrated in Eq. (5), the operational temperature conditions, denoted by τ, are incorporated into the model by reducing them to a binary representation under a set of conditional constraints, thereby

characterizing the ambient state (S^{τ}). Specifically, the operating temperature—measured in degrees Celsius—is discretized into two categorical states, representing low and high thermal conditions. This binary encoding simplifies the modeling of environmental influences on equipment behavior while retaining the essential impact of temperature fluctuations on system performance.

$$S^{\tau}=\begin{cases}0, & if \quad \tau \in [25,60] \\ 1, & if \quad \tau > 60\end{cases} \tag{5}$$

The Cartesian product of all relevant state variables provides a comprehensive representation of the resulting state space in the proposed system model.

$$S = S^{i} \times S^{\tau} \tag{6}$$

The system's action space is defined as a vector comprising three scalar actions: Replace (ξ), Repair (η), and Hold (κ). The maintenance agent operates under a budget constraint, represented by a maintenance credit account β. By default, the agent selects the Hold action κ, which incurs no maintenance cost. As equipment degrades over time—reflected by worsening sensor health readings—the cost and frequency of repair actions begin to rise gradually, eventually following an exponential growth trend.

Let $C_{\xi}(n,t)$ and $C_{\eta}(n,t)$ denote the cost of performing a replacement and a repair, respectively, on node n at time t. Define indicator functions

$$\begin{aligned}\mathbb{I}_{\xi}(n,t)&=\begin{cases}1, & \text{if action at node } n \text{ and time } t \text{ is Replace } (\xi), \\ 0, & \text{otherwise}\end{cases} \\ \mathbb{I}_{\eta}(n,t)&=\begin{cases}1, & \text{if action at node } n \text{ and time } t \text{ is Repair } (\eta), \\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{7}$$

Then the total maintenance cost over the decision horizon (for all nodes) is

10.22060/EEJ.2026.24815.5769

$$C_{\text{total}} = \sum_{n=1}^{N}\sum_{t=1}^{T}\left(C_{\xi}(n,t)\mathbb{I}_{\xi}(n,t)+C_{\eta}(n,t)\mathbb{I}_{\eta}(n,t)\right),$$

The natural and non-contradictory budget constraint is therefore

$$C_{\text{total}} = \sum_{n=1}^{N}\sum_{t=1}^{T}\left(C_{\xi}(n,t)\mathbb{I}_{\xi}(n,t)+C_{\eta}(n,t)\mathbb{I}_{\eta}(n,t)\right)\le \beta, \tag{8}$$

where β denotes the available maintenance credit (budget) for the considered horizon.

In many practical settings a single replacement is more expensive than a repair; to capture this typical cost relationship we optionally impose a per-action cost ordering

$$C_{\xi}(n,t) \ge \kappa C_{\eta}(n,t), \quad \kappa \ge 1, \tag{9}$$

where κ is a design parameter reflecting that replacement usually costs at least twice a repair. Note that Eq. (9) is an assumption or modeling choice (and not a budget constraint) that can be tuned to reflect the application-specific cost ratio; it does not introduce any contradiction with Eq. (8).

Remark 1: In the proposed Markov decision process (MDP), the DRL agent selects only among the high-level actions (Repair, Replace, Hold). The repair type ψ is defined as an environmental parameter that reflects the effectiveness of the repair process. When the agent selects a Repair action, the health state transitions according to $H(t+1) = H(t) + \psi\Delta_{\eta}$, where ψ represents the fraction of recoverable degradation restored by the repair, Δ_{η} represents the standardized repair improvement factor, and $H(t)$ is the equipment's health indicator at time t.

To simplify the explanation, we consider a scenario involving a single piece of equipment connected to one sensor ($N = 1$). In this setup, the maintenance agent selects actions based on observable changes in

system state over time. The Repair action (η) attempts to revert the sensor's condition by moving it backward by states y_{Rrpair}, ideally restoring it to a previously known healthy state. In contrast, the Replace action (ξ) nearly resets the sensor state to a condition approximating that of a brand-new unit. Importantly, not all repairs yield the same effect. The resulting state change $\phi(S)$ depends on the type of repair applied, denoted as ψ, where $\psi \in \{0,1,...,M\}$. Each value of ψ corresponds to a different repair type, influencing the magnitude of recovery. As illustrated in Figure 2, when η is triggered at a state $S_t = y - 1$, the selected repair type causes a state transition to $S_t^{'} = y - |\phi(S)|$. This behavior is formally summarized in Eq. (8)**,** reflecting the system's dynamic response to different maintenance strategies.

$$y_{\text{Re}pair} \in \{\phi(S) \mid S \in \psi\} \tag{10}$$

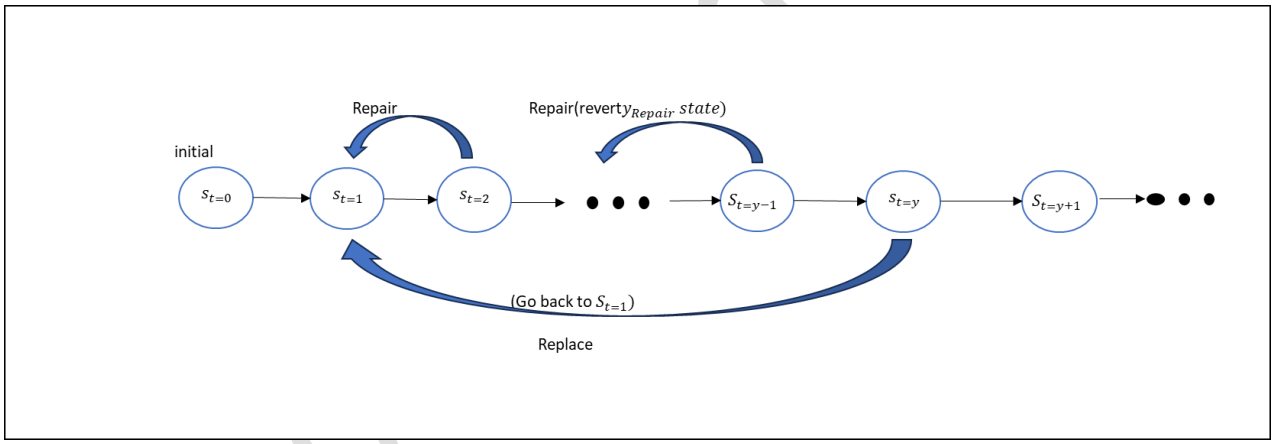


Fig. 2. Illustrative example of equipment state transitions triggered by replacement and repair operations.

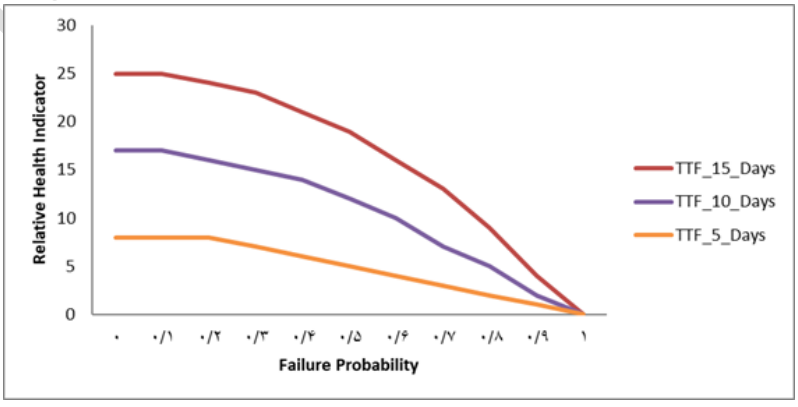


Fig. 3. Visualization of progressive health deterioration versus the risk of system failure.

The reward function, denoted as $R(s_t, a_t, s_{t+1})$, formally captures the benefit associated with transitioning from one state to another under a specific action. Based on the prior formulations, this reward can alternatively be interpreted as the aggregate operational time of the sensor, evaluated with respect to the current state S and selected action A. To guide the agent's decision-making process, Eq. (11) introduces an instantaneous scalar reward, $R_t \in R$, which quantifies the immediate outcome of each action taken within a given state context.

$$R_t = \begin{cases} R_{Rpl}, & \text{if } S_t^i > 0, \beta > 0 \\ R_{Rpa}, & \text{if } S_t^i > 0, \beta > 0 \\ R_{\text{Exp}}, & \text{if } S_t^i > 0 \\ R_{\text{Frug}}, & \text{if } S_t^i > 0, \beta > 0 \\ R_{\text{Pen}}, & \text{otherwise} \end{cases} \tag{11}$$

As an example, when the agent performs either the Replace (R_{Rpl}) or (R_{Rpa}) actions within the constraints of budget β, it is rewarded with significantly high health scores. Conversely, if the sensor condition drops to a critical level denoted as (R_{Pen}) or if the hold action (κ) is repeatedly selected throughout the episode, the agent incurs a heavy penalty. In many real-world applications, reward signals are sparse, making it challenging to drive exploration effectively. To address this, the study introduces a randomly defined exploratory reward (R_{Exp}), which serves to intrinsically motivate the agent to explore its environment. To further enhance the agent's likelihood of visiting underrepresented states—where potentially valuable rewards may exist—a ranking-based prioritization strategy is proposed and detailed in Section IV. Additionally, to mimic more human-like decision tendencies, a frugality-driven reward, (R_{Frug}), is implemented to encourage the agent to execute selected actions in an efficient and resource-conscious manner.

10.22060/EEJ.2026.24815.5769

The purpose of R_{Exp} is to discourage repetitive actions and prevent the agent from falling into local optima (e.g., selecting "Hold" at every step). The function is defined as:

$$R_{\text{Exp}}(t) = \begin{cases} +\lambda_{\text{exp}}, & \text{if } a(t) \neq a(t-1) \\ 0, & \text{otherwise} \end{cases}$$
$$\lambda_{\text{exp}} = 0.1$$

This value was selected empirically to encourage behavioral diversity while ensuring that exploration does not dominate the main maintenance-related rewards.

R_{Frug} promotes cost-efficient policies by rewarding the agent for preserving the remaining maintenance budget. Its precise formulation is:

$$R_{\text{Frug}}(t) = \lambda_{\text{frug}}\left(1 - \frac{C_{\text{used}}(t)}{\beta}\right),$$
$$\lambda_{\text{frug}} = 0.2$$

where $C_{\text{used}}(t)$ is the cumulative maintenance cost up to time t, and β is the total available budget. Thus, frugality is maximally rewarded early in the episode or when costs are low. As the agent consumes more budget, this reward gradually diminishes, creating a natural incentive to avoid unnecessary repairs or replacements.

The corresponding state-value function, ($V^{\pi}(s)$), is subsequently defined as follows:

$$V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma R_{t+1} \mid S_t = s] \tag{12}$$

The next step involves deriving the optimal policy (π^*) by employing the standard reinforcement learning paradigm. When Eq. (12) is analyzed under the assumptions of the Markov property, the associated value function can be re-expressed in a simplified and recursive form, as shown below:

$$V^{\pi^*}(s) = \sum_a \pi(s,a) \sum_{s' \in s} P_{\pi(s)}\left(s,s'\right)\left[R\left(s,a+\gamma V^{\pi}\left(s'\right)\right)\right] \tag{13}$$

Eq. (13) identifies the optimal action corresponding to the current policy function. Consequently, the Bellman optimality equation can be utilized to iteratively update the action-value function, $Q^*(s,a)$, which is formally expressed as:

$$Q^*(s,a) = (1-a) \quad Q(s,a) \quad + \alpha\left[R(s,a) + \gamma \max_{a' \in A} Q'\left(s',a'\right)\right]$$

Here, $\gamma \in [0,1]$ denotes the discount factor, which allows the agent to balance the trade-off between immediate rewards and potential long-term gains. The parameter α represents the learning rate, determining how significantly new information affects existing knowledge. To identify the most favorable action at each state, the max operator is used, selecting the action with the highest expected return. However, in many real-world applications, the state space is prohibitively large, making it infeasible to compute exact values for all state-action transitions. To overcome this limitation, Temporal Difference (TD) learning is adopted, offering a practical means of estimating the action-value function. The updated Q-value, denoted as $Q(S_i^i, A_t, \theta_t)$, gradually moves toward a target value y_t^Q, as shown in Eq. (14). This process closely resembles stochastic gradient descent in its iterative nature.

$$y_t^Q \equiv R_{i+1} + \gamma \max_A Q\left(S_{t+1}^i, A; \theta_t\right) \tag{14}$$

By integrating a Deep Q-Network (DQN)—a multi-layer feedforward neural architecture—with both a target network y_t^{DQN} and experience replay, the learning algorithm achieves significantly improved performance and generalization capabilities, as demonstrated in [12]. The target network, illustrated in Eq. (15), serves as a static copy of the primary Q-network. To maintain training stability, the target network's parameters are synchronized with those of the main network every τ steps, following the rule $\theta^- = \theta_t$.

$$y_t^{DQN} \equiv R_{i+1} + \gamma \max_A Q\left(S_{t+1}^i, A; \theta_t^-\right) \quad (15)$$

4. DEEP Q-LEARNING APPROACH

4-1- Double deep Q-learning framework

Traditional DQN models are prone to overestimating action values due to random environmental noise and the inherent use of the are max function. To address these limitations and enhance stability, the Double Deep Q-Learning (DDQN) technique was introduced [13]. This method separates the processes of action selection and evaluation across two distinct networks. In the current research, the DDQN agent takes exploratory actions, and its input at each time step is a tuple containing sensor readings (x_t^i). The target DQN network generates corresponding Q-values based on this input and can be expressed as follows:

$$y_t^{Double\ DQN} \equiv R_{t+1} + \gamma Q\left(s_{t+1}, a^*; \theta_t^-\right) \quad (16)$$

$a^* = \arg\max Q(s_{t+1}, A, \theta_t)$. The parameters of the target network, denoted as (θ_t^-), are periodically updated by copying from the main Q-value network. The discounted Q-value, represented as ($y_t^{Double\ DQN}$), is computed using the target network with the copied weights (θ_t^-). In particular, while the evaluation of future rewards is performed using the target Q-network, the selection of actions is carried out by the main Q-network.

4-2- Prioritized experience replay

Within the Deep Reinforcement Learning (DRL) framework, the agent is expected to navigate reward constraints to identify an optimal decision point based on the dataset or system model. Since the likelihood of failure increases exponentially over time, equipment replacement is most likely to occur during the later stages of its operational life (refer to Figure 3). In this scenario, rewards are distributed in a sparse-

to-dense manner, and the chances of triggering either R_{Rpl} or R_{Rpa} actions become significantly higher as the equipment nears the end of its lifecycle. This observed behavior suggests that an ideal target value for the normalized health index of the equipment is approximately 0.2.

Although the traditional approach of combining a greedy policy with an Experience Replay (ER) buffer is widely used, it proves inadequate in scenarios involving sparse rewards—an issue frequently encountered in real-world environments. The ER buffer tends to oversample a limited number of high-reward experiences, leading to sampling bias. Additionally, due to the variation in Time-to-Failure (TTF) cycles across different equipment units, relying on a fixed value decay rate as a hyperparameter is not an effective strategy. To address these challenges, Prioritized Experience Replay (PER) [14] has been proposed as an enhancement to the standard DDQN framework.

As an enhancement to the standard ER method, Prioritized Experience Replay (PER) emphasizes experiences that exhibit large differences between the predicted and target Temporal Difference (TD) values. By leveraging the magnitude of TD error as a priority metric, the DRL agent can implicitly focus more on underexplored or rare states—those with higher learning potential—which may ultimately lead to improved reward outcomes.

The Temporal Difference (TD) error magnitude ($|\delta_i|$) can be incorporated into each replay buffer entry as an extended tuple: $(s_t, a_t, r_t, s_{t+1}, |\delta_i|)$. To address overfitting commonly observed in standard experience replay, stochastic prioritization [14] is employed to assign a selection probability $P(i)$ to each state-action pair in the buffer—see Eq. (17). Here, P_i denotes the priority score of the *i*-th experience, and α is a hyperparameter that controls the degree of randomness in the sampling process. When $\alpha = 0$, experiences are chosen uniformly at random, while $\alpha = 1$ heavily favors transitions with the highest priorities. The notation $\sum_k .$ implies that all priority values within the replay buffer have been normalized.

$$P(i) = \frac{P_i^{\alpha}}{\sum_k P_k^{\alpha}} \tag{17}$$

$$\left(\frac{1}{N}, \frac{1}{P(i)}\right)^b \tag{18}$$

During neural network training, prioritizing high-value experiences introduces a bias, as these samples may no longer reflect the true underlying distribution. To mitigate this effect, Importance Sampling (IS) Weights are applied to adjust the influence of frequently selected experiences within the replay buffer. A scaling parameter β is gradually increased (annealed) toward 1 throughout the training process, controlling how strongly IS impacts the learning updates. In this context, N represents the total number of experiences in the replay buffer, while $P(i)$ refers to the sampling probability defined in Eq. (17). For a deeper understanding of prioritized experience replay, readers are referred to [15].

4-3- Parameter noise integration

In sparse reward environments, the challenge of inefficient exploration—a well-known limitation in Reinforcement Learning (RL)—becomes even more pronounced. Traditional RL methods influence the agent's action policy at each time step, but they do not directly alter the internal parameters that shape the agent's decision-making process (i.e., the neural network policy). Evolutionary strategies, on the other hand, focus on adjusting these internal parameters while leaving the action selection mechanism unchanged during rollouts. Parameter Noise (PN) [16] offers a hybrid solution by injecting noise directly into the agent's decision parameters. This approach enhances exploration in both on-policy and off-policy settings by enabling more consistent and structured behavior. By adding controlled randomness to the agent's policy parameters, PN reduces erratic behavior and supports more effective exploration, especially in environments with sparse or unevenly distributed rewards. To address this challenge, our study incorporates PN into the RL framework, specifically targeting exploration efficiency. The proposed method, Prioritized DDQN with Parameter Noise (PDDQNPN), combines the benefits of Prioritized

10.22060/EEJ.2026.24815.5769

Experience Replay and Parameter Noise. A high-level outline of this approach is provided in *Approach 1*.

5. Experimental Study

5-1- Dataset description

To evaluate the proposed model, we utilized the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset developed by NASA [17], which is widely adopted in existing research. This dataset, generated through the turbofan engine degradation simulator, contains sensor measurements that mimic the degradation behavior of multiple turbofan engines operating under various conditions.

Algorithm 1: The initial method proposed is called Parameter Noise-Enhanced Prioritized Double Deep Q-Learning (PDDQ)

1: **The inputs**: include the action space A , the mini-batch size L_b , and the update frequency of the target network L^{-}

2: **Outcome:** The optimal policy π^{*} is determined

3: **Initialization:** Experience is stored in a prioritized replay buffer with capacity N . The action-value function Q is initialized with random weights, while the target action-value function $\hat{Q}$ uses weights $\theta^{-}=\theta$. The main network is represented by Q_{θ}, and the target network by $Q_{\theta^{'}}$.

4: Iterate through episodes from 1 up to E .

5: Within each episode, loop over timesteps from 1 to T .

6: At each timestep, observe the current state s_t and select an action a_t following the policy $a_t \sim \pi(a,s)$.

10.22060/EEJ.2026.24815.5769

7: With a certain probability ε, take a random action instead.

8: If not random, select the action a_t that maximizes the estimated Q-value: $a_t = \arg\max_a Q(s_t, a) Q(s, a, \theta)$.

9: Perform action a_t and obtain the reward r_t.

10: Observe the next state s'.

11: Insert the experience tuple (s_t, a_t, r, s') into the replay buffer D with the highest priority.

12: Draw a random mini-batch of transitions of size (L_b) from the prioritized replay buffer $D_{priority}$, based on their assigned priorities.

13: $$y_t^{DDQN} = \begin{cases} r, \text{ if timestep } +1 \text{ is when the episode ends} \\ r + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a', \theta^-\right), \text{elsw} \end{cases}$$

14: Apply gradient descent to minimize the loss $(y_t^{DQN} - Q(s_t, a_t))^2$, and adjust the priority values of the sampled transitions accordingly.

15: Every L^- steps, synchronize the target network by setting $\theta^- = \theta$.

16: Update the current state: $s_t \leftarrow s'$.

17: Increase the time step count by one.

Continue this process until the time step exceeds T, then end the procedure.

Repeat the cycle until the episode count surpasses E, at which point the process concludes.

The datasets contain fault scenarios and complex correlations with various sensor outputs across four similar subsets (FD001 to FD004). For simplicity, only FD001 and FD003 engine datasets were utilized. A summary of the dataset is presented in Table 1. Each file includes 26 attributes: the first two columns

10.22060/EEJ.2026.24815.5769

indicate the engine ID and the cycle number, columns 3 through 5 reflect operational parameters such as temperature, and the remaining columns (6 to 26) consist of raw sensor measurements.

Table 1. Evaluated C-MAPSS dataset [17].

Dataset	FD001	FD003
Training Paths for Datasets	100	100
Trajectories of Testing	100	100
Conditions of Operation	1	1
Conditions of Fault	1	2

5-2- Data Preparation

Each sensor's data (i.e., each column corresponding to a sensor) has been individually normalized to ensure consistency across varying measurement scales.

$$\hat{x} = \frac{x_i - \mu_i}{\sigma} \tag{19}$$

Here, μ_i and σ represent the mean and standard deviation for each specific sensor type, respectively. The variable x_i denotes the value of the *i*-th reading within its corresponding sensor dataset.

The normalized dataset is then augmented with the estimated Remaining Useful Life (RUL), and a linear regression model is subsequently applied. After determining the sensor parameters, all data exhibited a Gaussian distribution, leading to the creation of an RUL distribution plot. To reduce the dimensionality of the sensor measurements, principal component analysis (PCA) was applied to derive the equipment health indicator. According to [17], when actual equipment health labels are unavailable, the degradation trend can be modeled as an exponential decay function.

$$H(t) = 1 - d - e^{at^b} \tag{20}$$

Here, a and b represent weighting factors, while d denotes the initial non-zero degradation level. Figure 5 depicts a similar deterioration pattern observed in the turbofan health indicator dataset. It is important to note that only replacement actions are evaluated, and in the absence of ground truth for repair or replacement, the equipment is considered to be of medium criticality.

5-3- Findings

The PDDQN-PN model [17] was implemented utilizing deep learning libraries based on TensorFlow, with the Adam optimizer employed for training. Prior to the actual training phase, a warm-up period of 5000-time steps were conducted using a random policy to fill the agent's memory. To balance exploration and exploitation, the exploration rate was maintained at 80% throughout the training simulation steps.

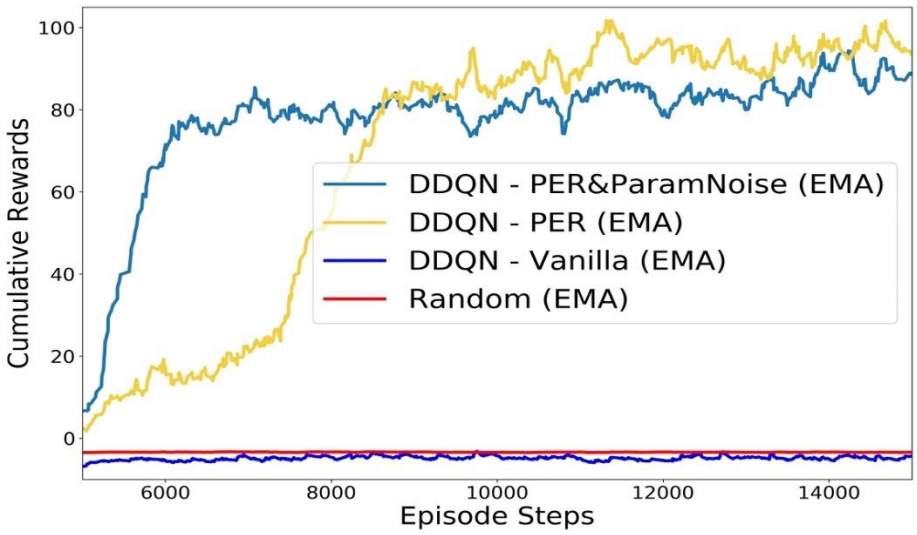


Fig. 4. Performance in learning: evaluating the proposed elements against a random strategy.

Separate tests were conducted using sensor data from engine #76 in the FD001 dataset to demonstrate the benefits of the proposed algorithm. This clearly highlights the contribution of each sub-component to the overall learning performance. The DRL agent reached an optimal policy within 12,000-time steps,

achieving an average cumulative reward of 95, largely due to the Prioritized Experience Replay (PER) module. When parameter action noise was incorporated, learning efficiency improved by approximately 50%, albeit with a 10% decrease in performance. Figure 4 indicates that with extended simulation time and reduced exploration duration, the average scores of both DDQN-PER and PDDQN-PN are expected to converge. Due to the challenge of sparse rewards, DDQN-Vanilla performed slightly worse than a random policy, as shown in Figure 4. To enhance clarity, the results were smoothed using an exponential moving average (EMA) with a smoothing factor of 0.18.

The decision-making policy learned by the DRL agent was evaluated using a model prediction method. Results presented in Table 2 revealed that the median values for DDQN-based algorithms were consistent, with PDDQN-PN exhibiting the largest standard deviation at 1.27×10^{-2}, attributed to the inclusion of action parameter noise. The inability of the random policy agent to acquire a suitable policy was clear, while DDQN-Vanilla showed prediction performance inversely related to that of the DDQN-PER variant.

Table 2. Average results of model predictions on engine #50.

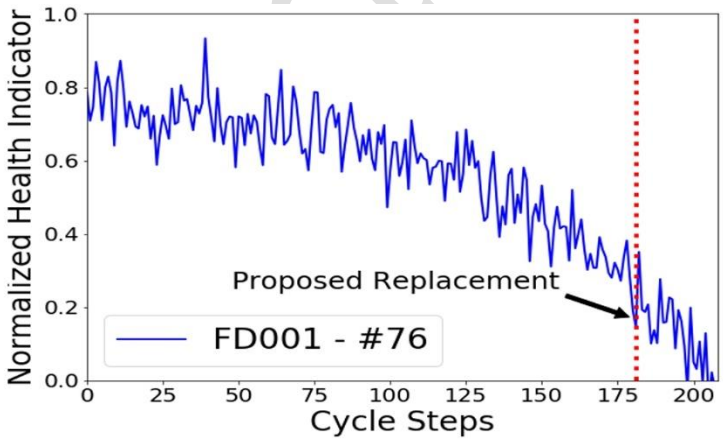
The method	The median statistic	The variability indicator
DDQN combined with PER and action noise	0.17	0.013
DDQN with PER mechanism	0.17	0.011
Standard DDQN	0.17	0.011
Baseline random agent	0.897	0

Engine health indicators were randomly selected from both training and test datasets as part of the validation process, with results displayed in Figure 5. Although the initial phase reveals a steep decline signaling imminent equipment failure, the DRL agent is still capable of recommending an effective replacement strategy (refer to Figure 5c). This is primarily due to the failure-to-failure penalty constraint,

10.22060/EEJ.2026.24815.5769

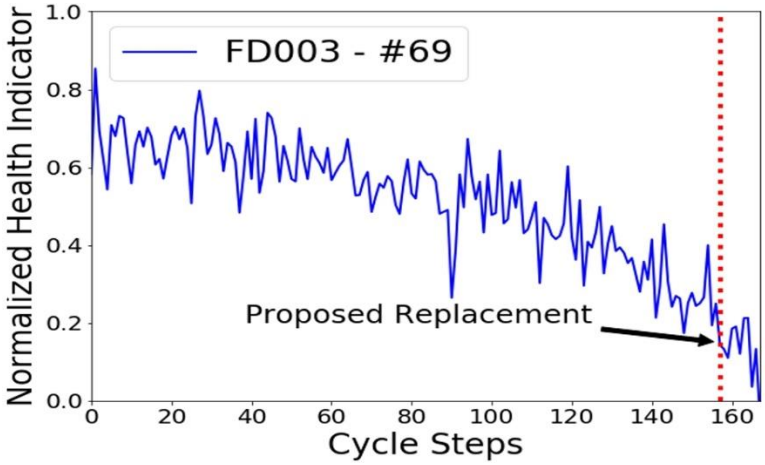
(R_{Pen}), defined in Eq. (11). Consistent suggested actions by the DRL agent were observed across different engines and datasets. Proposed replacement points are illustrated in Figure 5. The median cross-validation score on FD001 was 0.177 ± 0.02, while for FD003 it was 0.174 ± 0.02.

Remark 2: The proposed PDDQN-PN framework is designed for maintenance decision-making under budget constraints rather than pure Remaining Useful Life (RUL) prediction. Therefore, direct comparison with deep learning–based RUL regression models (e.g., LSTM- or CNN-based approaches) is not straightforward, as they address a fundamentally different problem. Such RUL predictors are better regarded as complementary modules that can be integrated into the state representation of a reinforcement learning agent. The selected baselines (random policy and DDQN) ensure a fair comparison within the same decision-making paradigm, allowing the impact of the proposed prioritization network and reward design to be clearly isolated.

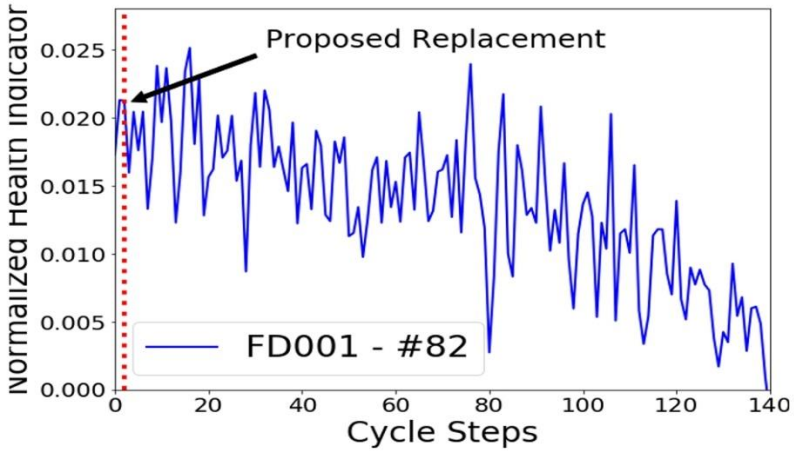


(a)

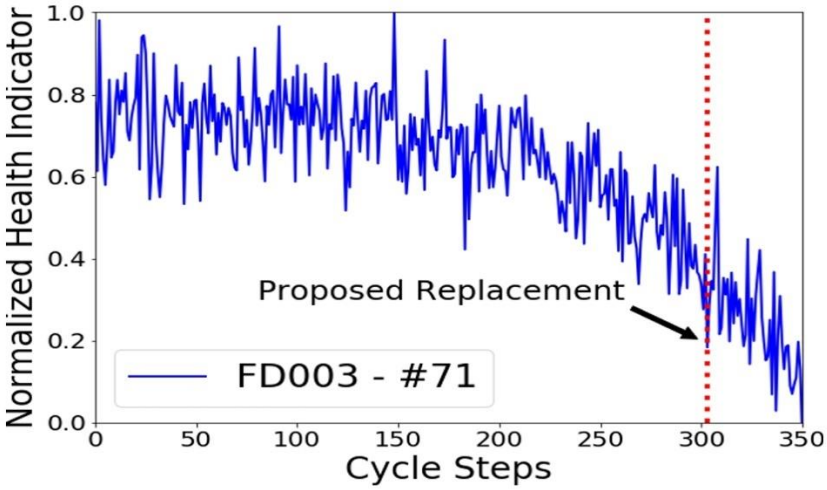
10.22060/EEJ.2026.24815.5769



(b)



(c)



(d)

Fig. 5. Illustration of replacement points suggested by the DRL agent.

Remark 3: The experimental evaluation in this study is conducted on the FD001 subset of the C-MAPSS dataset to provide a controlled validation of the proposed DRL-based maintenance decision framework. While FD002 and FD004 introduce higher operational variability and multiple fault modes, extending the evaluation to these subsets requires additional retraining and tuning and is therefore left as an important direction for future work.

Remark 4: Although the performance of the proposed framework is evaluated using cumulative reward, this metric directly reflects the core objectives of the study. Specifically, lower cumulative maintenance costs are achieved by penalizing unnecessary repair and replacement actions, while higher effective equipment uptime is promoted through health-aware decision-making that delays failure and avoids premature shutdowns. Thus, improvements in cumulative reward correspond to both reduced maintenance expenditure and enhanced operational availability.

6. Conclusion

As industries worldwide embrace the industry 4.0 paradigm to boost production, maintaining modern machinery has become increasingly challenging. Consequently, there is a growing demand from customers for accurate, interpretable, and actionable insights derived from predictive maintenance systems. In this study, we propose a strategy that delivers practical recommendations tailored to the condition of specific equipment. Our approach formulates a method to maximize equipment availability by leveraging multiple sensor inputs, using state-action pairs to effectively represent the corresponding health states of the machinery. Determining the optimal states is challenging with existing techniques, resulting in a sparse and scattered distribution. To address this, we propose a model-free deep reinforcement learning algorithm capable of rapidly learning an optimal maintenance policy—achieving this within approximately 2000-time steps. Despite differences in initial health conditions, experimental

results demonstrate consistent maintenance recommendations across similar machinery. Future work could involve benchmarking this approach against actual equipment maintenance schedules and extending it to other datasets involving equipment failures.

References

[1] H. Hesabi, M. Nourelfath, A. Hajji, A deep learning predictive model for selective maintenance optimization, Reliability Engineering & System Safety, 219 (2022) 108191.

[2] C. Riccio, M. Menanno, I. Zennaro, M.M. Savino, A new methodological framework for optimizing predictive maintenance using machine learning combined with product quality parameters, Machines, 12(7) (2024) 443.

[3] M.H. Abidi, M.K. Mohammed, H. Alkhalefah, Predictive maintenance planning for industry 4.0 using machine learning for sustainable manufacturing, Sustainability, 14(6) (2022) 3387.

[4] U. Dereci, G. Tuzkaya, An explainable artificial intelligence model for predictive maintenance and spare parts optimization, Supply Chain Analytics, 8 (2024) 100078.

[5] Z.M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, B. Safaei, Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0, Sustainability, 12(19) (2020) 8211.

[6] S. Arena, E. Florian, F. Sgarbossa, E. Sølvsberg, I. Zennaro, A conceptual framework for machine learning algorithm selection for predictive maintenance, Engineering Applications of Artificial Intelligence, 133 (2024) 108340.

[7] A. Aminzadeh, S. Sattarpanah Karganroudi, S. Majidi, C. Dabompre, K. Azaiez, C. Mitride, E. Sénéchal, A machine learning implementation to predictive maintenance and monitoring of industrial compressors, Sensors, 25(4) (2025) 1006.

[8] S. Choubey, R.G. Benton, T. Johnsten, A holistic end-to-end prescriptive maintenance framework, Data-Enabled Discovery and Applications, 4(1) (2020) 11.

[9] I.P.S. Ahuja, J.S. Khamba, Total productive maintenance: literature review and directions, International journal of quality & reliability management, 25(7) (2008) 709–756.

[10] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI conference on artificial intelligence, 2016.

[11] H. Li, X. Qian, W. Song, Prioritized experience replay based on dynamics priority, Scientific Reports, 14(1) (2024) 6014.

[12] A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (XAI), IEEE access, 6 (2018) 52138–52160.

[13] Y. Ren, Optimizing predictive maintenance with machine learning for reliability improvement, ASCE-asme journal of risk and uncertainty in engineering systems, part b: mechanical engineering, 7(3) (2021) 030801.

[14] K.-L. Zhou, D.-J. Cheng, H.-B. Zhang, Z.-t. Hu, C.-Y. Zhang, Deep learning-based intelligent multilevel predictive maintenance framework considering comprehensive cost, Reliability Engineering & System Safety, 237 (2023) 109357.

[15] B. KAMPMARK, Georey Hinton, AI, and Google's Ethics Problem, CounterPunch, (2023).

[16] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature machine intelligence, 1(5) (2019) 206–215.

[17] A. Kumar, R. Shankar, L.S. Thakur, A big data driven sustainable manufacturing framework for condition-based maintenance prediction, Journal of computational science, 27 (2018) 428–439.